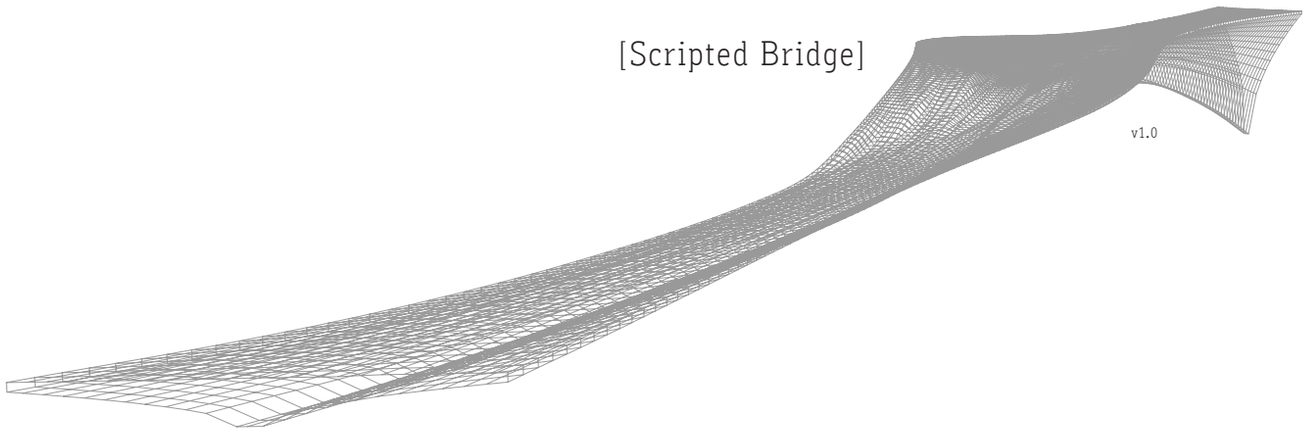
[Scripted Bridge]

v1.0



Scripted Bridge

DIPLOMARBEIT

zur Erlangung des akademischen Grades eines Diplom-Ingenieurs

Studienrichtung Architektur

Christian Pichlkastner

Technische Universität Graz
Erzherzog-Johann-Universität
Fakultät für Architektur

Betreuer
Ass.Prof. Dipl.-Ing. Dr.nat.techn. Andreas TRUMMER
Institut für Tragwerksentwurf

Mai/2010

EIDESSTATTLICHE ERLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Ort, Datum

Unterschrift

STATUTORY DECLARATION

I declare that I have authored this thesis undependently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

location, date

signature

Die vorliegende Arbeit spiegelt die Verarbeitung eines Themengebietes wider, welches mich während des Diplomstudiums immer wieder beschäftigt hatte und zu welchem ich abschließend eine Haltung gewinnen wollte.

Danke

Besonderer Dank gilt **Andreas Trummer**, der die Betreuung meiner Diplomarbeit übernommen hat und während der gesamten Zeit neben mir als einziger den Überblick über die Arbeit bewahren konnte. Dabei war Andreas Trummer auch abseits vereinbarter Termine immer zu ausführlichen Diskussionen bezüglich einzelner Problemstellungen bereit und ist dabei hilfreich gewesen, mich auf nützliche Literatur hinzuweisen und unterschiedlichste Kontakte zu vermitteln. Außerdem möchte ich mich bei Andreas dafür bedanken, dass er mir gegenüber viel Geduld entgegengebracht und während der langwierigen Dauer Nervenstärke bewahrt hat.

Herzlicher Dank gilt **Helmut Schober**, der mir bei Detailfragen zu einzelnen Berechnungen sowie deren Überprüfung auf Richtigkeit hilfreich zur Seite stand und sich dabei aufgrund seines knappen Zeitplanes auch am Wochende beziehungsweise in seiner Freizeit Zeit für mich nahm. Danke auch für die ambitionierte Zuammenarbeit bei dem Realisierungswettbewerb "Connecting Link" im Büro "Structural Design Olipitz".

In diesem Zusammenhang gilt mein Dank auch **Michael Olipitz**, der mir die Teilnahme an dem Realisierungswettbewerb "Connecting Link" und die Erfahrung bezüglich der Zusammenarbeit zwischen Architekten und Ingenieuren in der Praxis ermöglichte.

Danken möchte ich **Daniela Puffer**, die mit mir gemeinsam an dem Studentenwettbewerb "Concrete Student Trophy 2009" teilnahm und dabei im Team die Bauingieurseite vertrat.

Vielen Dank an **Florian Graf** und **Norbert Seyff**, die mich mit Informationen für den Abschnitt der "Architekt als Programmierer" versorgten.

Danken möchte ich auch **Georg Pircher** von der Firma "ABES Pircher & Partner GmbH", der in einem Gespräch detailierte Auskünfte bezüglich Informationen über Statiksoftware vermitteln konnte.

Ein großes Dankeschön an meine KollegInnen von "**vonhausaus**", die mich währed der gesamten Diplomarbeitsphase unterstützt haben. Danke an **Marion Winkler**, die meine Kennisse im Layoutbereich entscheidend erweitert hat, an **Johannes Schlattau** für Unterstützung bei Visualisierungen, an **Gernot Siegel** für seine Hilfe bei konstruktiven Detailüberlegungen und an **Christian Sturmaier** für das Mitfeilen an den Berechnungsüberlegungen.

Danken möchte ich auch meinen Freunden der **"Dienstagsrunde"**, die immer interessiert an meinem Diplomarbeitsthema gewirkt haben und meine oft längeren Monologe über die Arbeit mit viel Geduld ausgehalten haben.

Außerordentlicher Dank und ein dickes Bussi gebührt meiner Freundin **Bettina Schmalzmayer**, die diese Arbeit Korrektur gelesen und auch bei so manchen Formulierungen konstruktiv mitgewirkt hat.

Vor allem möchte ich mich hiermit bei meinen Eltern, **Johanna** und **Siegfried Pichlkastner**, bedanken, die meine etwas konservativen humanistischen Ansichten bezüglich des Studiums unterstützten, indem sie mir im Grunde eine unbeschwerte "Zeit" für das Studium ermöglichten.

Einleitend

Zunehmend hat die Diskussion rund um „Freiformdesign" in der Architektur in den letzten Jahren wieder an Bedeutung gewonnen. Neue computergestützte Entwurfsmethoden ermöglichen dem/-r Architekten/-in - über eine spielerische Herangehensweise - komplexe Formen zu konstruieren oder auch unter bestimmten Rahmenbedingungen generieren bzw. optimieren zu lassen. Durch einen hohen Grad an maschineller Fertigung können nun immer mehr Entwürfe nicht nur in Zeitschriften als gewagte visionäre Illustrationen abgebildet werden, sondern manifestieren sich auch im Alltag vor einem breiten Publikum in gebauter Form. Besonders die erste Welle an realisierten Projekten war oft dem Vorwurf von verschwenderischer Beliebigkeit, einem Mangel an Funktionalität oder auch dem Hinweis auf erhebliche Diskrepanz zwischen Form und Konstruktion ausgesetzt. Als Reaktion auf eine Zeit, in der die Frage nach der Ökonomie und Nachhaltigkeit von Konstruktionen und Begriffe wie jene der Vorfertigung oder Flexibilität große Aufmerksamkeit genießen, zeigen immer mehr Projekte einen weit rationaleren Umgang im Einbinden digitaler Werkzeuge in den Entwurfsprozess. Neben dem Architekten als „Plugin-Scout", der auf der Suche nach neuer Software in einem gigantischen virtuellen Experimentierlabor interaktiv seine Konzepte entwickeln kann, werden die zunehmend leistungsfähigeren Systeme für den Umgang mit komplexen Aufgabenstellungen verwendet. Letztere Ansätze werden zunehmend dem Begriffsfeld des „parametrischen" Entwurfes zugeordnet. Diese Entwürfe resultieren tendenziell aus einem mehr oder weniger komplexen Netzwerk an Parametern, die je nach Entwurfskonzept ausgewählt und untereinander verknüpft werden, wodurch sich in den Ergebnissen jeweils individuelle Optimierungsprozesse widerspiegeln. Manche Protagonisten sehen im „Parametrismus" den neuen internationalen Stil, wohingegen andere bereits auf dessen angeblichen "Tod" reagieren.

In diesem Kontext ist „Scripted Bridge" ein Beispielprojekt für die Einbindung digitaler Methoden in den architektonischen Entwurfsprozess. Unter architektonischen Konzeptvorgaben wird ein Brückengenerierungsprozess entworfen, welcher bei Veränderung von Parametern die Generierung von Entwurfsvarianten zulässt. Dabei wird der Frage nachgegangen, inwiefern dieser parametrische Entwurfsansatz den gesamten Entstehungsprozess beeinflusst und wie Zusammenhänge zwischen Material und Konstruktion einfließen können. Insgesamt wurde ein weiteres Beispiel für den aktuellen Diskurs geschaffen.

Inhaltsverzeichnis

Aufbau der Untersuchung

Generell ist das vorliegende Buch die schriftliche Dokumentation meiner Auseinandersetzung mit dem Einsatz digitaler Methoden in den architektonischen Entwurfsprozess. Die Arbeit wird in zwei Abschnitte unterteilt.

In einem allgemeinen Teil wird ein Überblick der Entwicklungen digitaler Methoden in technischem wie auch theoretischem Hinblick skizziert, um sichtbar zu machen, in welchem Kontext sich der Hauptteil dieser Arbeit bewegt. Weiters begründet sich die etwas breitgefächerte, allgemeine Auseinandersetzung aus dem Grundziel, im Zuge dieser Diplomarbeit abschließend eine Haltung zu einzelnen Tendenzen rund um digitale Methoden im architektonischen Entwurf und dem Brückenentwurf als Königsdisziplin der BauingenieurInnen entwickeln zu können.

Im zweiten Abschnitt folgt eine detaillierte Darlegung der praktischen Auseinandersetzung, die sich im Wesentlichen mit der Machbarkeit eines parametrischen Freiformbrückenentwurfs beschäftigt und den damit verbundenen Aufwand dokumentiert.

A: Allgemeiner Teil

Die Entwicklung von CAD

In diesem allgemeinen Kapitel wird zunächst versucht, über die historische Entwicklung von computergestütztem Entwerfen einen Bezug zwischen der aktuellen und der historischen Debatte herzustellen.

Die Entwicklung des Computers – eine Kurzeinführung

In diesem Kapitel soll lediglich eine Vorstellung vermittelt werden, welches Grundprinzip nach wie vor auch den heutigen Computern zu Grunde liegt. Wie ein Rechner rechnet, „denkt", in welcher Zeit dieser entwickelt wurde und mit welcher Geschwindigkeit die Entwicklung fortgetrieben wird.

Entwicklungen und Visionen als Voraussetzung für CAD

Am Beginn dieses Kapitels steht die vorwiegend theoretische Auseinandersetzung mit der Verwendung des Computers. Diese Überlegungen werden anhand der in den 1960er Jahren stattfindenden Forschungsarbeiten in den USA am MIT in Cambridge und in Deutschland an der Technischen Hochschule Stuttgart nähergebracht. In einem zweiten Abschnitt wird schließlich auf die daraus resultierende Entwicklung von CAD-Software eingegangen. Dieses Kapitel orientiert sich stark an der Aufarbeitung dieses Themas in dem Buch „Kulturtechnik Entwerfen".

Die Entwicklung von CAAD-Software zur Verwendung als Scriptingtool

Da heutzutage ein Großteil der digitalen Unterstützung in der Praxis über die Anwendung von unterschiedlichsten Softwareprogrammen stattfindet und sich die vorliegende Arbeit ebenfalls einer solchen Software bedient, wird in diesem Abschnitt die Entwicklung und aktuelle Verwendung von CAAD-Software behandelt. Zunächst wird auf deren Anfänge und Entwicklungsansätze verwiesen, um danach speziell auf die Entwicklung zum Skriptingwerkzeug einzugehen.

Parametrisches Entwerfen – Fragmente der aktuellen Debatte

Zu Beginn wird die aktuelle Diskussion rund um den „Parametrismus" und die digitalen Methoden anhand von einigen einzelnen Positionen und Beispielen behandelt. Dabei wird dargelegt, inwiefern die gegenwärtige Argumentation mit den im vorangehenden Abschnitt erläuterten geschichtlichen Entwicklungen verknüpft ist und wo neue Tendenzen in der digitalen Avantgarde Architektur sichtbar werden. Dabei wird auch versucht, zumindest die gängige bzw. populäre Kritik an den Entwicklungen durch bekannte Architekten und Theoretiker wie Peter Eisenman und Rem Koolhaas exemplarisch anzudeuten. Weiters wird darauf hingewiesen, dass dieses Kapitel, welches sich vor allem auf Beiträge in Zeitschriften, Berichte und Interviews aus dem Internet bezieht, dazu dienen soll, die vorliegende Arbeit tendenziell einordnen zu können.

Allgemeines zum Entwerfen von Brücken

Um einen unmittelbaren Bezug zum Entwurfsthema herzustellen, wird zunächst versucht die „Königsdisziplin" der BauingenieurInnen über die rein funktionelle Bedeutung als effektive Tragkonstruktion hinaus zu betrachten. Besonders aus dem Blickwinkel der Architektur ist es wichtig, das Wesen von Brücken als Überlagerung von Zeit, Technik und Kultur zu begreifen. In einem weiteren Abschnitt wird der Konflikt zwischen ArchitektInnen und BauingenieurInnen und deren Rolle beim Entwerfen von Brücken zu verdeutlichen versucht. Da in den Diskussionen rund um die Konstruktion und Ästhetik von Brücken immer wieder die zwingende Kenntnis der Grundprinzipen der Tragwerkslehre betont werden und ein wesentlicher Teil der Scripted Bridge ohne dieses Verständnis nicht verstanden werden kann, werden auch die Bemessungsschritte und Kernbegriffe in Grundzügen erläutert.
Um darzulegen wie Brücken im digitalen Zeitalter entworfen werden, wird in einem letzen Abschnitt auf die Entwurfssoftware eingegangen. Dabei wird untersucht, wie weit neue digitale Methoden wie Analyse-, Simulations-, Modellierungstechniken etc. genutzt und die Softwareprogramme in den Planungsprozess eingebunden werden.

B: Scripted Brigde

Aufgabenstellung

Aus dem allgemeinen Kontext ergibt sich, wie bereits erwähnt die Aufgabe der Gestaltungen eines Brückenentwurfprozesses. Zu Beginn dieses Abschnittes wird zunächst die Wahl des Entwurfsthemas begründet.

Um den Aufwand und einzelnen Probleme eines parametrisierten Entwurfes an einem praktischen Beispiel feststellen zu können, orientiert sich die Entwicklung des Konzepts für die Gestaltung des Entwurfsprozess' an einer konkreten Aufgabenstellung. Als Rahmenbedingung für den Entwurf diente die Aufgabenstellung des von der Vereinigung der Österreichischen Zementindustrie im März 2009 ausgeschriebenen Wettbewerbes „Concrete Student Trophy 2009". In späterer Folge wurde eine Überprüfung und Weiterentwicklung in der Zusammenarbeit mit dem Ingenieurbüro „Structural Design Olipitz" an dem von der Stadt Wien im Sommer 2009 ausgeschriebenen einstufigen Realisierungswettbewerb „Connecting Link" durchgeführt. Der wesentliche Unterschied in den zwei Aufgabenstellungen besteht darin, dass letzterer keine Materialeinschränkung beinhaltete.

Demzufolge werden an dieser Stelle die zwei annähernd identischen Ausschreibungen der Wettbewerbe „Concrete Student Trophy 2009" und „Connecting Link" zusammengefasst erläutert.

Brückengenerierungsprozess

In diesem Teil soll dargelegt werden, welche Motivationen und Zielsetzungen bei der Entwicklung des Brückengenerierungsprozess' im Vordergrund standen. Weiters wird die Entwurfsidee für die Automatisierung des Entwurfsprozesses und das damit verbundene Grundkonzept erläutert. Dabei wird beispielsweise festgelegt, welche Eigenschaften der Wettbewerbsausschreibungen bzw. des Themas „Brückenentwurf" in Form von Parametern berücksichtigt wurden. In einem weiteren Abschnitt wird der Ablauf und die Organisation bzw. Vernetzung der einzelnen Funktionsboxen bzw. Module des Prozesses beschrieben. Anschließend folgt die Erläuterung dieser einzelnen Funktionselemente. Die generelle chronologische Entwicklung dieser Module wird kurz anhand einer Skizze verdeutlicht. Am Schluss des Kapitels wird erklärt, inwiefern man dieses Entwurfstool für das Entwerfen einer Brücke verwenden kann.

Generieren einer "Scripted Bridge"

Um das Verwenden des entwickelten Entwurfstools dem Leser auch praktisch näher zu bringen, soll hier anhand von zwei Grundbeispielen schrittweise der Generierungsprozess erklärt werden.

Fallbeispiele: Brücken über den Wienfluss

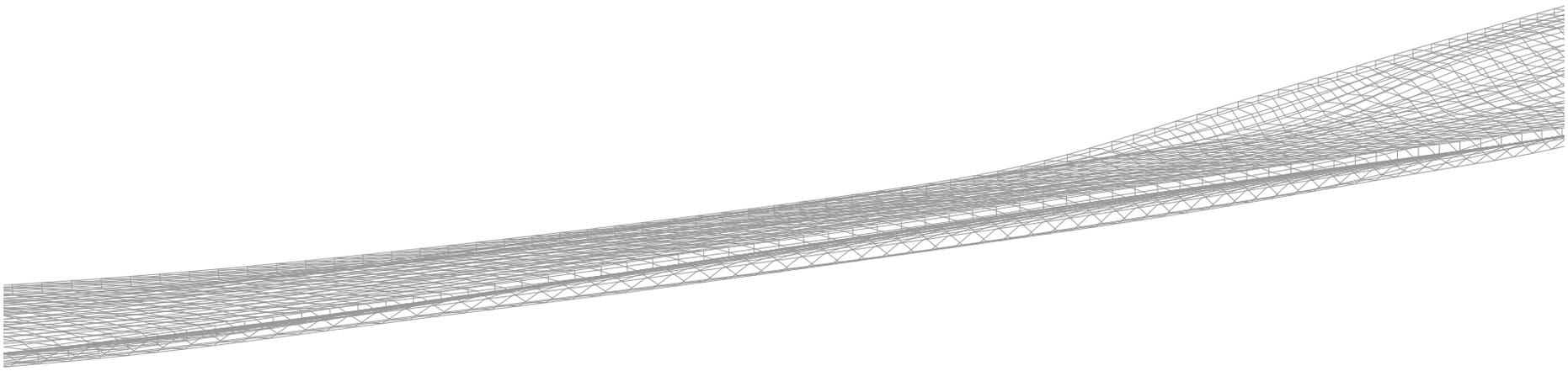
In diesem Abschnitt werden als praktische Beispiele die Ergebnisse der beiden Wettbewerbe „Concrete Student Trophy 2009" und „Connecting Link" dokumentiert. Da die Brücken mit diesem parametrisierten Prozess entworfen wurden, war die Teilnahme an den genannten Wettbewerben neben der Parameterfindung auch für die Überprüfung des Konzeptes wichtig.

Resümee

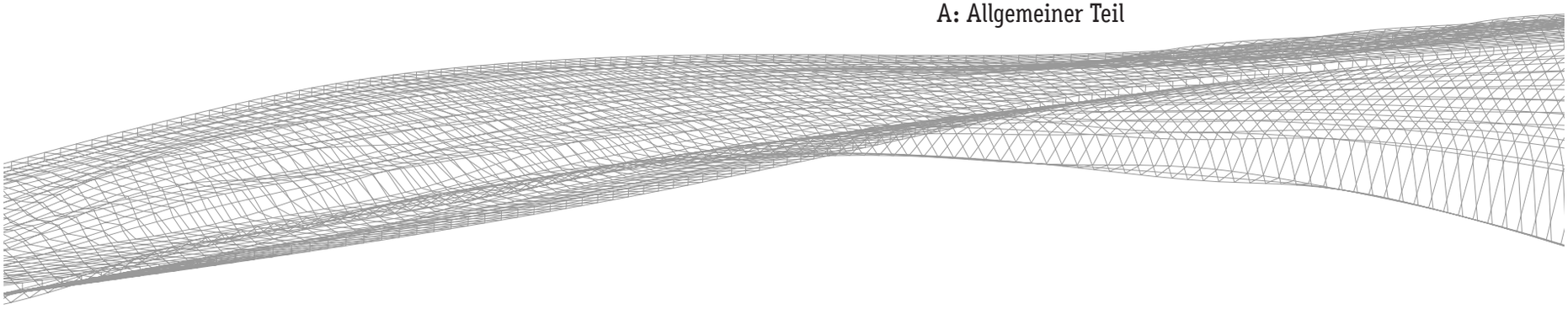
Abschließend werden die wichtigsten Erkenntnisse, die im Zuge dieser Arbeit gewonnen wurden, zusammengefasst und mögliche Perspektiven für die Zukunft vorgeschlagen.

Appendix

Zu Beginn wird auf den ersten Seiten collagenhaft ein kleiner Auszug aus den Skizzenbüchern gezeigt. Damit soll plakativ dargestellt werden, dass während des Entwickeln der digitalen Module bzw. Scripte eine Menge Ideen und Überlegungen vorerst analog in Handskizzen festgehalten wurden. Danach werden Auzüge gezeigt, die die Veränderungen des dynamischen Planungsprozess grafisch dokumentieren sollen. Abschließend werden die Texte der Scripte abgebildet. Dem Verfasser dieser Arbeit ist klar, dass üblicherweise die Quellcodes verschiedenster, mittels Scripting entstandener Entwürfe aus unterschiedlichsten Gründen wie etwa des möglicherweise enormen Platzbedarfes oder des Copyrights nicht bzw. wenn nur in kleinen Ausschnitten abgebildet werden. Auch in Informatikarbeiten ist es nicht üblich, den Quellcode in abgedruckter Form abzubilden. Da diese Diplomarbeit aber in den Architekturbereich fällt und dem Autor aufgrund der Recherche und der unzähligen Gesprächen mit Studien- und ArbeitskollegInnen ein extremer Mangel an Bewusstsein für den Aufwand für banalste Ergebnisse aufgefallen ist und der Zeitaufwand für die Erstellung der einzelnen Scripte in der vorliegenden Arbeit mehr als die Hälfte der gesamten „Scripted Bridge" eingenommen hat, wird dieser mit marginalen Kürzungen annähernd vollständig gezeigt. Eine genaue Beschreibung ist nicht möglich. Es wird jedoch in der Beschreibung der einzelnen Funktionsblöcke bzw. Module auf einzelne Abschnitte im Quelltext verwiesen.



A: Allgemeiner Teil



A: Allgemeiner Teil:

Allgemeines zur Entwicklung von CAD

Die Entwicklung des Computers – eine Kurzeinführung

Das binäre Zahlensystem

In einem digitalen Zeitalter, in dem digitale Methoden in verschiedensten Bereichen bereits die digitalisierte Normalität darstellen und auch Menschen zumindest zeitweise in digitalen Welten[1] mit digitalen Freunden leben, gilt es hier zunächst einmal der Frage nach der Herkunft und Bedeutung dieses Begriffes nachzugehen. Das Wort „digital" stammt vom lateinischem Wort „digitus" ab, bedeutet Finger und verweist auf ein Zahlensystem, mit dem in erster Linie gerechnet wird. Als Basis heutiger Computersysteme ist dieses spezielle Zahlensystem von enormer Wichtigkeit. Auch der Begriff „Computer" verweist auf den Umgang mit Zahlen und wurde anfangs jedoch noch nicht im Zusammenhang mit Maschinen verwendet. *"Schon im Mittelalter hießen in England mathematische Mitarbeiter, die langwierige und umfangreiche Berechnungen für Astronomen durchführten »Computer«, übersetzt »Rechner«."*[2] Dieses duale oder auch binäre Zahlensystem ist ein Stellenwertsystem, welches als Basis auf der Zahl „Zwei". Das bedeutet, dass für die Darstellung von verschiedensten Zahlen lediglich zwei Zeichen (im Binärsystem üblicherweise „0" und „1") zur Verfügung stehen, wobei diese je nach Positionierung in der Zeichenfolge (Stelle) unterschiedliche Wertigkeiten zugewiesen bekommen (siehe Abb. 1). Diese Zeichenfolgen werden in sogenannte Register geschrieben. Ein einzelnes Zeichen wird als Bit bezeichnet. Acht Zeichen ergeben ein Byte.[3]

Die einzelnen Zeichen werden mittels Aussagenlogik kombiniert, wodurch z.B. alle Grundrechenarten dargestellt werden können. Die Aussagenlogik dient auch als Grundprinzip beim Lösen von Problemstellungen mittels Programmiersprachen und wird deshalb im Kapitel "Parametergesteuertes Entwerfen mittels Skripten oder Node-Systemen" genauer erläutert. Wie das Rechnen mit Binärzahlen genau funktioniert, ist nicht Teil der Untersuchung und wird folglich auch nicht weiterbehandelt.[2]

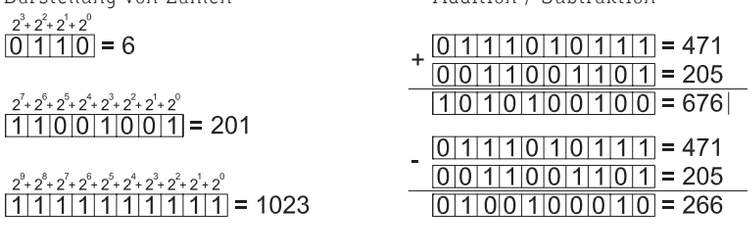


Abbildung 1: Binäres Zahlensystem: Mit dieser Methode kann man z.B. mit 10 Fingern bis 1023 zählen.

Bereits Gottfried Willhelm Leibniz (1646-1716) entdeckte die Möglichkeit, das System der Zahlen binär zu codieren und hatte auch die Idee eines Notationssystems, das später als „Begriffsschrift" bekannt wurde und schuf damit bereits die Grundlagen der Digitaltechnik. Leibniz entwickelte auch einen eigenen unvollkommenen mechanischen Rechner und stellte diesen 1673 der Öffentlichkeit vor. Für dessen Funktion hat Leibniz allerdings noch keine Binärcodierung verwendet.[4]

Vom ersten binären Rechner zur ersten EDV-Anlage

Nach diesem Prinzip - dem der Binärcodierung - entwickelte der Bauingenieur und Hobbyerfinder Konrad Zuse (1910-1995) im Jahr 1938 die erste universelle Rechenmaschine mit dem Namen „Z1". Diese sollte zum Lösen von einfachen statischen Berechnungen eingesetzt werden. "Z1" gilt als Vorläufer des modernen Computers, wobei das von Zuse entwickelte Modell noch nach elektromechanischen Prinzipien funktionierte. Im Gegensatz zu seinen Vorgängern verwendete Zuse bereits das duale Zahlensystem, bistabile Schaltelemente und nutzte für Berechnungen die Logik des Aussagenkalküls (UND, ODER und NEGATION)[5]. Die Dateneingabe erfolgte mittels Lochkarten, welche bei Zuse für die Abbildung einer binären Codierung eingesetzt wurden. Das Prinzip der Lochkartencodierung wurde von Hermann Hollerith (1860-1929) im Jahr 1886 für eine Zählmaschine entwickelt, wobei noch keine binäre Codierung verwendet wurde[6]. Obwohl die elektrische Telegrafie bereits um 1837 als *„die Muttertechnologie aller informatischen Maschinennetzwerke"*[7] als erste Anwendung von Elektrizität im industriellen Stil eingesetzt wurde, verging ein gutes Jahrhundert, bis die erste elektronische Datenverarbeitungsanlage „ENIAC" (Electronic Numerical Integrator and Computer) 1946 in Betrieb genommen werden konnte. "ENIAC" wurde in Pennsylvania im Auftrag des amerikanischen Verteidigungsministeriums zur Berechnung von Schießtabellen von John W. Mauchly (1907-1980) und J. Presper Eckert (1919-1995) entwickelt. Zum ersten Mal wurden Elektronenröhren als Schaltelemente eingesetzt, wodurch sich die Rechengeschwindigkeit gegenüber den vergleichbaren elektromagnetischen Relaisrechnern um das 2000-fache erhöhte.[8] An dieser Stelle wird nicht auf weitere Entwicklungen (wie z.B.: Transistorbauteile oder die Entwicklung eines flexiblen Programmspeichers durch John von Neumann), die für das Funktionieren eines modernen Computers notwendig waren, eingegangen.

Vom Großrechner zum Personal Computer

Mit "ENIAC" begann langsam auch die Serienproduktion von Großrechnern, welche für verschiedenste Berechnungen adaptiert wurden. Um die Maschinen nicht immer neu auf ihr Einsatzgebiet anpassen zu müssen, wurden Softwareschnittstellen entwickelt, um verschiedenen Firmen die Entwicklung von zusätzlichen individuellen Erweiterungen zu ermöglichen. In den 1960er Jahren begann eine rasante technische Entwicklung. Diese äußerste sich nicht alleine in den Leistungssteigerungen der Rechner, sondern auch in Erweiterungen und Verbesserungen der Peripherie.

Die Firma IBM entwickelte im Jahr 1964 das erste Betriebssystem namens „OS/360" und brachte am 12. August 1981 das sehr erfolgreiche Betriebsystem "DOS" auf den Markt.[9] Für viele beginnt die Geschichte der heutigen Homecomputer in den 1970er Jahren mit der enormen Entwicklung der Mikrocomputertechnologie.[10] Diese Homecomputer oder auch Personal Computer wurden am Beginn der 1980er Jahre zuerst in manchen Firmen einzelnen Mitarbeitern an ihren Arbeitsplatz gestellt, wobei parallel langsam auch immer mehr Privathaushalte mit den Miniaturausgaben der Großrechenanlagen ausgestattet wurden. Die Fortschrittsgeschwindigkeit bei der Entwicklung der Computer wird mit dem ökonomischen Gesetz der Mikroelektronik beschrieben. Dieses Gesetz wurde von Gordon Moore im Jahr 1965 zum ersten Mal formuliert, weshalb es auch nach ihm benannt wurde (Moorsches Gesetz). Es besagt, dass alle zwei Jahre eine Verdoppelung der auf einem Computerchip integrierten Transistorfunktionen zu erwarten ist.[11] Gegenwärtig prognostiziert Moore gegenwärtig ein Ende seines Gesetzes in den nächsten 10-20 Jahren.[12] Den aktuellen Fortschritt in der Computerchipindustrie beschreibt Moore aus der Sicht der Firma Intel als dessen Gründungsmitglied positivistisch wie folgt:

"NO EXPONENTIAL IS FOREVER . . . BUT WE CAN DELAY "FOREVER"."[13]

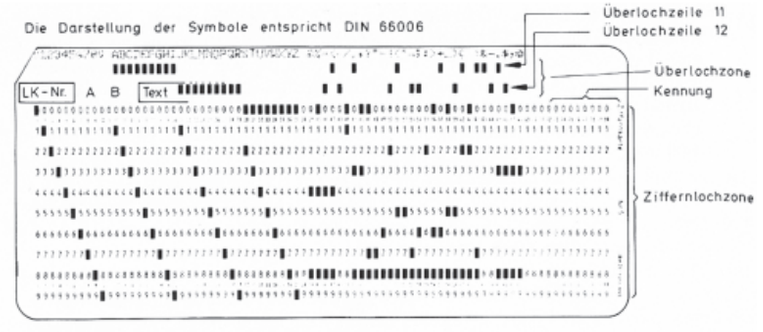


Abbildung 2: Modene Lochkarte nach DIN 66 018

1 Bemerkung: In dem Computerspiel namens „Second Life" von der Herstellerfirma Linden Lab kann man sich als virtuellen Charakter in einer virtuellen Welt vertreten lassen und durch diesen an der Gestaltung der immateriellen Umgebung mitwirken. Siehe http://secondlife.com/?v=1.1 (Stand 14.02.2010).

2 Gottfried Wolmeringer, "Coding for Fun", Galileo Press, 1. Auflage, 2009, S.24-27.

3 Vgl. Ebd. S.24-27.

4 Georg Franck, "Maschinelle Entwurfshilfen", Kulturtechnik Entwerfen, Hg. Gethmann u. Hauser, 2009 tanscript Verlag Bielefeld Deutschland, S.228 oder auch Edgar P. Vorndran, "Entwicklungsgeschichte des Computers", 1982, VDE-Verlag GmbH, S.40.

5 Vgl. Edgar P Vorndran, "Entwicklungsgeschichte des Computers", a. a. O., S.75.

6 Ebd. S.59.

7 Wolfgang Pircher, "Entwerfen zwischen Raum und Fläche", Kulturtechnik Entwerfen, Hg. Gethmann u. Hauser, 2009 tanscript Verlag Bielefeld, Deutschland, S.115.

8 Vgl Edgar P Vorndran, "Entwicklungsgeschichte des Computers", a. a. O., S.89.

8 Vgl. Gottfried Wolmeringer, "Coding for Fun", a. a. O., S.108.

9 Ebd. S.108.

10 Vgl Michael Friedewald, "Der Computer als Werkzeug und Medium", Verlag für Geschichte und Naturwissenschaften und der Technik, Berlin, Diepholz 1999, S.355.

11 Ebd., S.250.

12 Vgl Gordon Moor, http://www.intel.com/cd/corporate/pressroom/emea/deu /archive/2005/212674.htm (Stand 12.02.2010).

13 Gordon Moore, http://download.intel.com/research/silicon /Gordon_Moore_ISSCC_021003.pdf (Stand 16.02.2010).

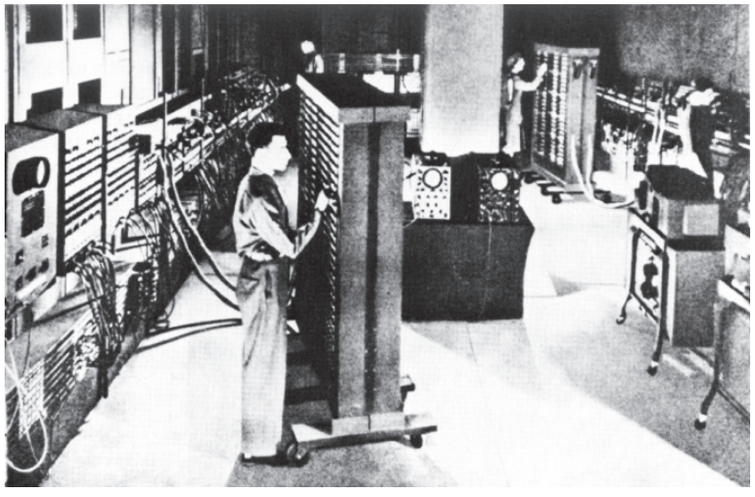


Abbildung 3: Die erste EDV-Anlage "ENIAC" in Pennsylvania USA.

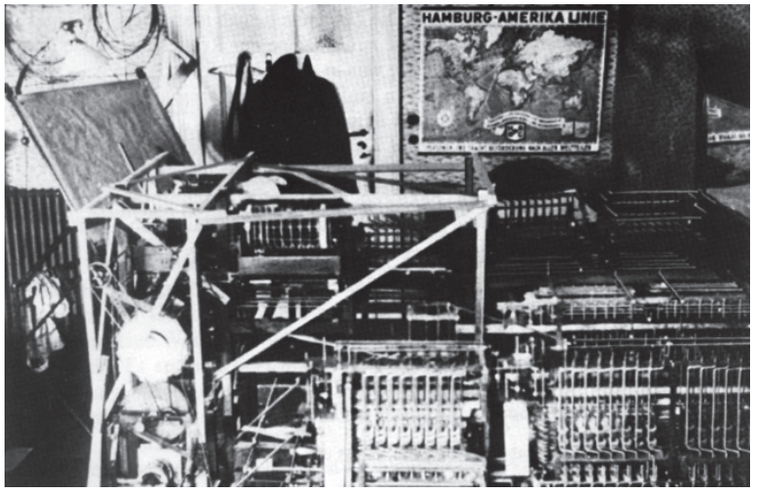


Abbildung 4: Aufbau der Rechenmaschine "ZUSE Z1" in der elterlichen Wohnung (Berlin 1936)

Visionen und Entwicklungen als Voraussetzung für CAD

CAD und Kybernetik

Die Abkürzung „CAD" kommt aus dem Englischen und steht für „Computer Aided Design", was üblicherweise als „computergestütztes Entwerfen" ins Deutsche übersetzt wird.[14] Der englische Begriff „Design" kann im Deutschen neben dem Begriff des „Entwerfens" auch als „Konstruieren" übersetzt werden, wodurch im englischen Sprachgebrauch eine allgemeinere Verwendung möglich ist.

Eine der zwingend notwendigen Grundvoraussetzungen für die Entwicklung des Rechners zum Entwurfs- bzw. Konstruktionswerkzeug war ein Umdenken bezüglich der Erweiterung des Einsatzgebietes des Computers. Als Beispiel dieser neuen Konzepte soll hier zunächst die Entwicklung in den USA am Massachusetts Institut of Technology (im Folgenden: „MIT") in Cambridge rund um Nicholas Negroponte und danach die Ansätze in Deutschland an der Technischen Hochschule Stuttgart rund um Max Bense anhand der Arbeiten des Architekten Manfred Kiemle in Grundzügen dargestellt werden.

Manche dieser Arbeiten werden oft dem Wissenschaftsgebiet der „Kybernetik" zugeordnet. Der Begriff „Kybernetik" geht auf Norbert Wiener (1894-1964) zurück, der im Jahr 1948 in seinem Buch „*Cybernetics or Control and Communication in the Animal and the Machine*" diesen vom Griechischen „*kybernétes*" für „*Steuermann*" abgeleitet hat. Norbert Wiener war später auch an dem, noch weiter unten im Text genannten, Projekt „Whirlwind" am MIT beteiligt. Die Kybernetik entwickelte sich aus dem Forschungsgebiet der Steuerung und Regelungstechnik. Kernprinzip der Kybernetik ist das Konzept der negativen Rückkopplung in Regelungskreisen, welches als ein universeller Ansatz begriffen über die bis dahin geltende linear-kausale Denkweise hinauswies. Dies bedeutet, dass „*die Wirkungen von Ursachen*" in einer Art von Kurzschlussschaltung nun "*selbst wieder zu einer Ursache von Wirkungen werden*" konnten. In weiterer Folge führten diese zu dem Gedanken, dass komplexe Systeme aus Elementen bestehen, die selbst kleine Organisationsgruppen darstellen, also eine Vernetzung von Systemen in Systemen. Besonders die Einbeziehung nichttechnischer Systeme wie etwa den Menschen, ist ein wesentlicher Aspekt der Kybernetik. Begriffen als Konzept für vernetztes Denken wurde dieses besonders in der Forschung rund um „künstliche Intelligenz" oft als Analogie zu den Denkprozessen im Gehirn gedeutet. Nach dem Zweiten Weltkrieg entwickelte sich die Kybernetik zu einer der umstrittensten Forschungsgebiete des 20. Jahrhunderts und wirkte sich auf die Entwicklung von verschiedensten Wissenschaftsdisziplinen aus.[15]

Als der Beginn von computergestütztem Konstruieren bzw. Entwerfen werden im Allgemeinen die Forschungsentwicklungen in den 1960er Jahren am MIT genannt. Dabei wird das von Ivan Sutherland im Jahr 1963 entwickelte Projekt namens „Sketchpad" meist als ein signifikanter Entwicklungsschritt in der Entstehung von CAD angesehen. In enger Verbindung dazu stehen auch die zukunftsweisenden Arbeiten der „Architecture Machine Group" (im Folgenden: „AMG"), die von Nicolas Negroponte im Jahr 1967 ebenfalls am MIT gegründet wurde. Deren visionäre Forschungsprojekte bewirkten einen Wandel in der Betrachtung des Computers als ein einfaches Rechenwerkzeug und öffneten den Horizont für die Entwicklung digitaler Methoden in verschiedensten Anwendungsbereichen.

Vannevar Bush und die Memex

Bis lange nach dem Zweiten Weltkrieg wurde der Computer nämlich als ein „Werkzeug" verstanden, das in einem sehr eingeschränkten Anwendungsgebiet dem/der IngenieurIn für unterschiedlichste Berechnungsaufgaben zur Seite gestellt werden konnte. Obwohl der Ingenieur Vannevar Bush (1890-1974) in seinem Aufsehen erregenden Aufsatz "As we may think" in den USA bereits 1945 eine fiktive Maschine namens „Memex" skizziert hatte, in welchem Bush schon damals eine universelle Verwendung des Computers auch für zivile Zwecke andachte, fanden sich Bush's innovative Überlegungen erst ca. 15 Jahre später als Denkanstoß für neue Entwicklungskonzepte in einigen Forschungsprojekten wieder. Bush, der bereits in den Jahren von 1919 bis 1938 ebenso wie später Sutherland und Negroponte am MIT beschäftigt gewesen war, dachte sich seine Maschine „Memex" (Memory Extender) als einen individuellen und universellen Wissensspeicher zur Erweiterung des persönlichen Gedächtnisses des Menschen. "Memex" sollte über verschiedenste Eingabe- und Ausgabegeräte gesteuert werden und Informationen wie Text-, Audio- und Bilddaten speichern können. Hiezu hatte er einige innovative Ansätze entwickelt. So sollte ein/e WissenschaftlerIn eine - später von Zeitungsmagazinen so genannte - Zyklopenkamera mit der Größe einer Walnuss auf der Stirn tragen können, womit er durch die Aktivierung eines auf dem Arm montierten Sensors farbige Bilder seiner Arbeit speichern können sollte. Dabei sollte ein beliebiger Bildausschnitt

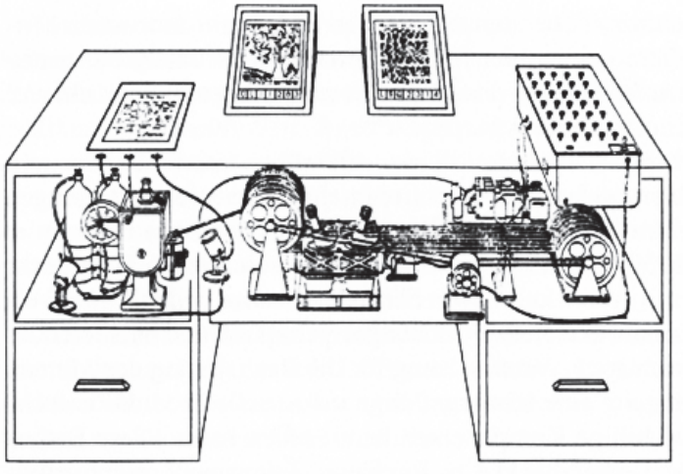


Abbildung 5: Skizze von Vannevar Bush's Gedächtniserweiterungsmaschine - Memex.

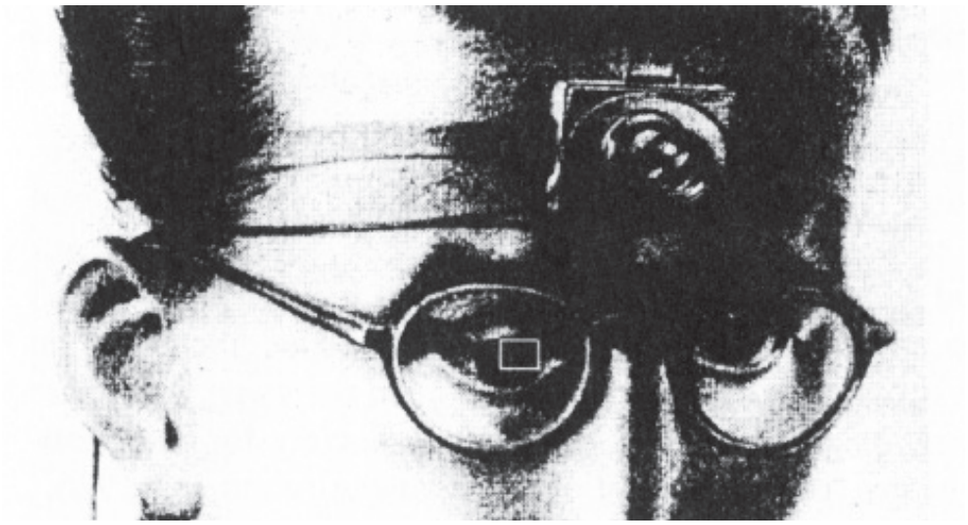


Abbildung 6: Vision der Zyklopenkamera als Unterstützung für wissenschaftliches Arbeiten.

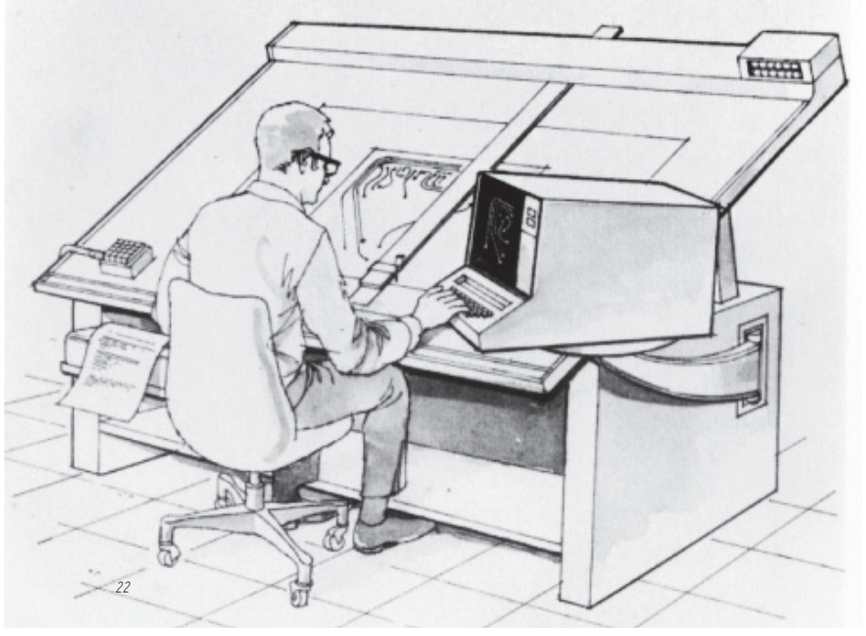
gewählt werden können, wobei das Suchfenster als kleines Rechteck mit feinen Linien auf die Brille des Benutzers projiziert werden sollte. Weiters regte Bush eine Spracheingabe mittels der bereits existierenden Technologie des „Vocoders" an, wobei durch diese eine natürlichsprachige Kommunikation zwischen "Memex" und Benutzer ermöglicht werden sollte. Das Vorgängergerät „Voder" konnte einen Text in Sprache akustisch wiedergeben und wurde von der „Bell Telephone Comp." 1939 auf der Weltausstellung in New York einer breiten Masse vorgestellt. Einige Jahre später entwickelte diese Firma schließlich den „Vocoder", welcher die mittels über ein Mikrofon aufgenommenen Sprachen in Textzeichen transformieren konnte. Weiters hatte Bush Ideen zum Einlesen von gedruckten Vorlagen. Dieses Konzept sollte mittels Fotozellen realisiert werden, welche zu diesem Zeitpunkt noch in ihrem Entwicklungsanfang steckten. Über eine Glasscheibe sollten Vorlagen eingescannt werden können, wobei zusätzlich zum Einlesen von Textvorlagen eine Art Texterkennung überlegt wurde. Alle diese Daten sollten auf einem optischen Speicher (Microfilm) mittels seines entwickelten „Rapid Selectors" über die "Memex" kompakt gespeichert werden. Zur Bewältigung der gigantischen Datenmenge, die im Gegensatz zum menschlichen Gehirn für die Ewigkeit archiviert werden sollte, entwickelte Bush eine Methode zur assoziativen Datenspeicherung. Dies bedeutet, dass am Beginn der jeweiligen Datensätze vermerkt wurde welche Information darin abgespeichert wurde. Durch diese Information – wie z.B.: der Informationstyp (Audioaufzeichnung, Bild, Text, etc.) oder die Länge bzw. Größe einer Aufzeichnung - wurde eine Unterscheidung zwischen den Daten und damit deren strukturierte Organisation möglich.[16]

Bush erhob keinen Anspruch auf die technische Realisierbarkeit seiner Maschine mit den zu seiner Zeit zur Verfügung stehenden Mitteln. Vielmehr lieferten seine kreativen Kombinationen von existierenden Technologien neue Lösungsvorschläge und Entwicklungsansätze. Wie bereits erwähnt wurde, fielen Teile seiner Visionen erst viel später auf fruchtbaren Boden. Bush selbst schrieb bezüglich der Realisierung und Zukunft seiner Maschine: *„that day is not yet here, but has come far closer..."*[17].

Ursprung von CAD am MIT

Zunächst wurde die Weiterentwicklung von Computersystemen häufig an Universitäten in Zusammenarbeit mit militärischen Einrichtungen vorangetrieben. So resultierten die unmittelbaren technischen Grundvoraussetzungen für die Erfindung von Ivan Sutherlands „Sketchpad", das meist als der erste Ansatz eines CAD-Programms angesehen wird, aus den intensiven Entwicklungsarbeiten an einem militärischen Flugsimulator, welche im Auftrag des amerikanischen Verteidigungsministeriums im Jahr 1944 am MIT unter dem Projektnamen „Whirlwind" begonnen wurden. Anfangs war die Entwicklung einer analogen Rechenanlage mit einer rückgekoppelten Steuerung geplant, die eine Echtzeiteingabe ermöglichen sollte. Inspiriert von dem in Pennsylvania gerade fertig gestellten elektronischen Digitalrechner „ENIAC" machte Jay Forrester – eine der Leitfiguren des Projektes „Whirlwind" – den Vorschlag, die Aufgabe mit digitalen Methoden zu lösen. Im Zuge der Arbeiten an „Whirlwind" entwickelte der Assistent Robert Everett eine erste Version der „Light Gun", mit deren Hilfe einzelne Punkte auf einem Kathodenstahlröhrenbildschirm (CRT-Screen) identifiziert werden konnten. Light Gun hatte die Form einer Pistole und war der Vorläufer des stiftförmigen Eingabegerätes namens „Light Pen", welches Ivan Sutherland später für die Steuerung von „Sketchpad" verwendete. Eines der wichtigsten Ergebnisse - neben der technologischen Verbesserung der Rechenleistungen von Computern - für die Entstehung von CAD-Anwendungen ist die im Zuge der Arbeiten an Whirlwind neu entstandene Echtzeiteingabe, mit deren Unterstützung nun eine interaktive Kommunikation zwischen Benutzer und Computer ermöglicht wurde. Das anschaulichste Ergebnis lieferte Ivan Sutherlands Projekt „Sketchpad". Sutherland griff bei der Gestaltung der grafischen Darstellung seiner Anwendung auf die theoretische Arbeit von Doug Ross zurück, welcher sich eine CAD-Software Struktur überlegte, die über eine vereinfachte

Abbildung 7: Erste Eingabegeräte: Linke Seite Light Gun, rechte Seite Light Pen



Eingabetechnik einem breiteren Benutzerkreis zugänglich gemacht werden sollte. Bei der Verwendung von Sketchpad konnte der Benutzer auf eine vom technischen Hintergrund stark abstrahierte grafische Eingabeoberfläche zurückgreifen, welche mittels Kathodenstahlröhrenbildschirm (in Folge "CRT-Bildschirm") dargestellt wurde. Es konnte mit vordefinierten grafischen Symbolen gearbeitet werden, die durch die Veränderung von Parametern manipuliert wurden. Sutherlands Terminal bestand aus einem CRT-Bildschirm, einer Light Pen, einem Panel mit mehreren Schaltknöpfen und vier Wähltasten, mit denen diese grafischen Elemente in Echtzeit zu komplexeren Darstellungen kombiniert werden konnten. Diese Zeichenelemente (wie z.B.: Linien, Kreise etc.) konnten - ähnlich wie später Blöcke oder Verbunde in CAD Softwareprogrammen – kopiert und an unterschiedlichste Stellen in der Zeichnung eingefügt werden, wobei Transformationen wie Skalieren, Verschieben und Verdrehen möglich waren. Um eine Linie von einem existierenden Punkt weiter zeichnen zu können, musste man mit der Light Pen am Bildschirm auf einen Punkt (wie z.B.: dem Endpunkt einer Linie oder dem Eckpunkt eines Quadrates) eines dargestellten Elementes tippen, wodurch ein neuer Startpunkt definiert wurde, danach die Taste „LINE" drücken und schließlich wieder mit der Light Pen kurz auf den gewünschten Endpunkt am Bildschirm zeigen. Egal welchen Pfad der Benutzer vorher am Bildschirm gezeichnet hatte, es wurde immer eine gerade Linie von dem eingegebenen Startpunkt bis zum zuletzt berührten Bildschirmpunkt gezeichnet.[18] Da der Endpunkt je nach zuletzt berührtem Bildschirmpunkt variieren kann, ähnelt die Linie durch ihre Längenveränderung einem Gummiband. Diese Gummibandmetapher wird als „Rubberbanding" bezeichnet.[19]

Damit war der Startschuss zur Entwicklung der ersten Generation von CAD-Anwendungen gefallen. Unter welchen zwei grundlegenden Ansätzen sich die CAAD-Software entwickelt hatte und in welchem Spektrum diese Software heute eingesetzt wird, ist Thema des nächsten Kapitels.

Negroponte und die AMG

Während langsam die erste Generation von CAD-Programmen vorwiegend unter der Motivation entwickelt wurde, den KonstrukteurInnen in verschiedensten Bereichen als flexible digitale Imitation von analogen Zeichenbrettern zu dienen, erkannte Nicholas Negroponte dagegen das Potential des Computers für den Architekturbereich. Mit "AMG" entwickelte Negroponte neue innovative Projekte und Visionen für digitale Medien für den Einsatz im Architekturentwurf und veröffentlichte diese 1970 in dem Buch „The Architecture Machine: Toward a More Human Environment" in der MIT-Press. Negroponte sah in der Verwendung von CAD die Chance für das Entstehen einer neuen humaneren Architektur. Negroponte's Fokus auf den architektonischen Entwurfsprozess rührt aus einer tiefen Unzufriedenheit, die er der Architektur seiner Zeit entgegenbringt. Er wirft den ArchitektInnen den Verlust der Fähigkeit

Abbildung 8-9: In oberen Abbildung demonstriert Ivan Sutherland sein Projekt "Sketchpas"; darunter ist eine Skizzen von zukunftsvisionen aus "The Architecture Machine" abgebildet.

zur Vermittlung zwischen individuellen Zielen und den Bedürfnissen der Masse besonders beim Entwerfen großer Projekte zugunsten von größeren räumlichen Strukturlösungen vor. So verkündet Negroponte, dass mit der Unterstützung durch die neuen digitalen Methoden schließlich „jeder Mensch Architekt sein kann". Jeder soll Architektur zuhause mit Hilfe digitaler Unterstützung entwerfen können, wobei die Rolle des ExpertInnen vom Computer übernommen werden soll. Bereits auf der ersten Seite seines Buches „The Soft Architecture Maschine" bemerkt er folgendes:

„This is a book about a new kind of architecture without architects"[20].

Negroponte begriff den Computer nicht mehr als ein einfaches Werkzeug, welches eine digitale Analogie zu physischen Gegenständen wie dem digitalem Zeichenbrett oder wie zuvor bei Bush eine persönliche Gedächtniserweiterung des Menschen darstellte, sondern vielmehr als ein leistungsfähiges Medium, welches dem/der EntwerferIn als *„Design Amplifier"*[21] ergänzend zur Seite gestellt werden kann. Während Bush seine Memex noch als *„gadget"*[22] - was ins Deutsche übersetzt einem technischen Schnickschnack gleichkommt – bezeichnet, begreift Negroponte den Computer als intelligentes Medium, das über verschiedenste Schnittstellen die Kommunikation zwischen EntwerferIn und Architekturmaschine ermöglicht und einen dialogischen Entwurfsprozess fördert. Einer der Schwerpunkte bei der Betrachtung des Computers als Vermittler zwischen der physischen und imaginären Welt liegt in der Entwicklung von Konzepten bezüglich der möglichen Schnittstellen zwischen Mensch und Maschine. Die Symbiose des/der ArchitektenIn mit der Maschine soll es über eine interaktive Steuerung ermöglichen, Raumkonzepte vor ihrer baulichen Umsetzung erlebbar und auch auf verschiedenste Aspekte hin prüfbar zu machen. Diese Interaktion kann über nahezu unendlich viele Methoden wie z.B.: Sprachsteuerung, Bewegungserkennung, verschiedenste Scanntechniken etc. stattfinden.

Als Beispiel für eine Schnittstelle, über welche die visuelle Erlebbarkeit von digitalen Räumen ermöglicht wird, führt Negroponte die Arbeiten Sutherlands und Raymond Goertzim's an. Sutherland und Goertzim forschten 1966 an einem ersten „Head Mounted Display" (in Folge: „HMD") und einem Datenhandschuh. Diese Arbeiten sind die Vorläufer, der heutigen „Cave-Systeme", in denen virtuelle Räume erleben werden können.[23]

Weiters soll Negroponte's Architekturmaschine neben der Funktion als Analyse- und Simulationsinstrument nicht einfach nur Skizzen bewerten können, sondern auch in der Automatisierung des Entwurfprozesses als intelligente Maschine künstliche Kreativität aufweisen und als Ideenlieferant zur Erzeugung von Varianten beitragen. Anhand des Projektes „HUNCH" zeigt Negroponte unter anderem eine Arbeit der AMG, durch deren Anwendung Grundrisse auf den Gehalt von Elementen wie Kreisen, Quadraten, Linien etc. analysiert werden konnten, wobei die eventuell verwackelten, kurvigen oder verzitterten Linien der Eingangsskizze nach der Analyse in einfachen

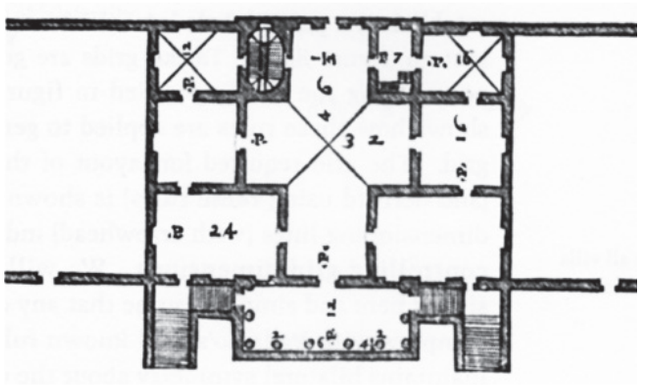


Abbildung 10: William Mitchell originaler Grundriss von Palladio

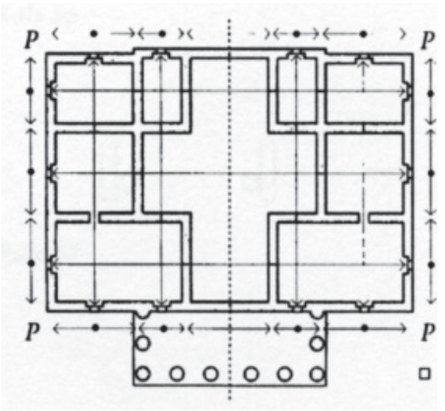


Abbildung 11: William Mitchell; computergenerierter Grundriss

Basisgeometrien transformiert wurden. Zeichnet der/die BenutzerIn ein Quadrat in schnellen Zügen, so interpretiert der Rechner das Quadrat mit einfachen geraden Linienzügen. Wird der/die EntwerferIn beim Zeichnen beispielsweise in den Eckbereichen langsamer, dann wird von der Software angenommen, es sei hier ein höherer Detaillierungsgrad gefordert, wodurch folglich die Ecken des Quadrats als Kreisbögen interpretiert werden. Weiters zeigt Negroponte Softwareprogramme, die in Grundrissen dargestellte Räume auf ihre topologischen Raumzusammenhänge hin analysieren.[24]

Als Beispiel für künstliche Kreativität soll hier eine Arbeit aus den 1990ern von William Mitchell angeführt werden. Im Buch *„The Logic of Architecture"*[25] zeigt Mitchell ein von ihm entwickeltes Softwareprogramm, welches sich des Formenvokabulars und der Gestaltungsgrammatik des Architekten Andrea Palladio bedient und nun selbständig Grundrissvarianten von palladianischen Villen generiert (siehe Abb. 10-11).

An dieser Stelle sei erwähnt, dass im Jahr 1977 der Architekt Christopher Alexander sein Buch „A Pattern Language: Towns Buildings Construction" veröffentlichte, in welchem er darzustellen versucht hatte, dass beim Architekturentwurf auf eine Art Grammatik zurückgegriffen wird, in dem immer wiederkehrende archetypische Muster verwendet werden.

Alexander versuchte verschiedenste Architekturthemen und Elemente in Symbole zu fassen – die Alexander Pattern (Muster) nennt -, wobei diese je nach zugewiesener Bedeutung und Wichtigkeit in hierarchisierten Ebenen organisiert werden. Die Beziehung einzelner Architekturelemente untereinander kann über Diagramme anschaulich dargestellt werden. Alexander nimmt Anleihen an Sutherlands Sketchpad, welches mit grafischen Symbolen arbeitet und versucht sich an einer Systemtheorie für die gesamte Architektur. Bei Sutherland dient diese Art von Oberfläche auch dazu, diese Software einem breiteren Benutzerkreis zugänglich zu machen. Ähnliche Auswirkungen haben auch die von Alexander entwickelten „Design Pattern" auf den in den 1970er Jahren stark forcierten Begriff der „Partizipation". Durch die streng gegliederte Sammlung einzelner gewerteter Architekturelemente mit den dazugehörigen detaillierten Erklärungen sollte auch bei Alexander einem größeren Kreis an Laien ein Überblick vermittelt und Vokabular zur Verfügung gestellt werden, um im architektonischen Entwurfsprozess mitbestimmen zu können.[26] Einleitend schreibt Alexander:
„Du kannst die Sprache verwenden um mit deinem Nachbarn an der Verbesserung deiner Stadt und Nachbarschaft zu arbeiten. Du kannst sie verwenden um mit deiner Familie dein eigenes Haus zu entwerfen und zu bauen, oder um mit anderen Leuten am Entwurf von Büros, Werkstätten oder öffentlichen Gebäuden, wie Schulen zu arbeiten."[27]

Hier verfolgt Alexander ähnlich wie Negroponte, die Architektur der Allgemeinheit zugänglich zu machen. Alexanders Herangehensweise hat großen Einfluss auf die Informatik und ist besonderes mit Bereich der Forschungen an der künstlichen Intelligenz von CAD-Systemen verknüpft.

Während sich Alexanders folgende Arbeiten mehr auf die digitale Übersetzung seiner „Pattern Language" und auf Programmierung von Architektur fokussieren bleibt Negroponte bei einem offeneren allgemeinen Ansatz. Hauptanliegen Negroponte's ist und bleibt die Entwicklung einer humanen Architektur durch die Unterstützung von CAD. Im Jahr 1985 wird die AMG (Architecture Machine Group) passend zu seiner Auffassung des Computers als Medium in MIT „Media Lab" umbenannt.

Auch für den aktuellen Architekturdiskurs sind Negroponte's Ideen und Prognosen nach wie vor zukunftsweisend. So erkennt man den generellen Anspruch, den Negroponte für die Verwendung digitaler Methoden in verschiedensten Lebensbereichen erhebt, – und diese gehen weit über die Verwendung von CAD hinaus -, in seinem im Jahr 1995 erschienen Buch „Being digital" in zahlreichen Aussagen wie den folgenden: *„Computing is not about Computers anymore. It's about living."* Als Beispiel dafür schreibt Negroponte eine Seite später:

"We will socialize in digital neighborhoods in which physical space will be irrelevant and time will play a different role. Twenty years from now, when you look outside your window, what you see maybe five thousand miles and six time zones away"[28].

Max Bense und die Verwissenschaftlichung von Architektur

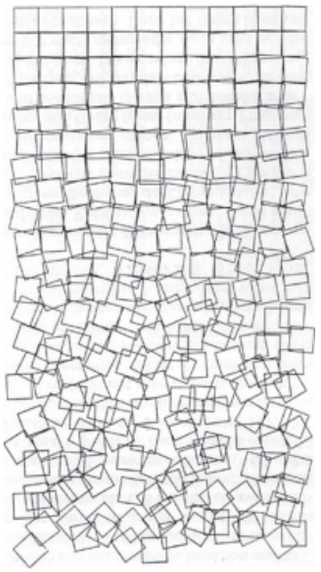
Parallel zu den Arbeiten um Negroponte entwickelte sich in Deutschland ausgehend von den theoretischen Arbeiten Max Bense's der Ansatz, mittels der technischen Errungenschaften eine Verwissenschaftlichung der Architektur vorzunehmen. Im Gegensatz zu Negroponte, der besonderes Augenmerk auf den Entwurfsprozess gerichtet hatte, lag die Aufmerksamkeit der „Stuttgarter Schule" rund um Max Bense in der quantitativen Bewertung der Ästhetik von fertig gestellten Architektur- und Kunstprojekten.

Ab dem Sommersemester 1949 lehrte der Philosoph Max Bense (1910-1990) an der Technischen Hochschule Stuttgart, welche als ein idealer Ort für die Entwicklung seiner Philosophie der „Technischen Existenz" diente. Bense versuchte eine Annäherung der technischen Wissenschaften an die Geisteswissenschaften und legte besonders mit seinen theoretischen Arbeiten über die Ästhetik den Grundstein für die von der „Stuttgarter Schule" unternommenen Versuche einer Verwissenschaftlichung der Architektur. Bense beschrieb das Wesen der realen Welt in seiner „Informationsästhetik" anhand von zwei symmetrisch gegenüberstehenden gegeneinander gerichteten Prozessen, die zusammen eine Einheit – Welt – bilden sollten.[29]

„So ging er davon aus, dass ein wirklicher und verwirklichender Prozess für die physikalischen und ästhetischen Erscheinungen der Welt sorgen. Während der physikalische Prozess auf einen Systemzustand maximaler Entropie (Unordnung) zuläuft, wirkt der nach maximaler Negentropie (Ordnung) strebenden ästhetischen Prozess diametral entgegengesetzt."[30]

Bense begreift die Perzeption von Kunst als das Auffinden von Ordnungsstrukturen in einem System von Unordnung, die sich seines Erachtens über den Quotienten der beiden Systeme in einem Messwert ergibt, welcher eine objektive Kritik von Ästhetik zulässt. Diese Ordnungsstrukturen werden durch das Auffinden von bekannten Zeichen und deren dynamische Beziehungen untereinander gekennzeichnet. Dabei versucht er in direkter Anlehnung an Claude Elwood Shannons (1916-2001) Ansätze zur Quantifizierung von Information, diese auch auf die Beuteilung des Schönen zu erweitern. Durch den allgemeinen Anspruch, den Bense in seinen theoretischen Arbeiten so wie in praktische Konzepten erhebt, welche unter anderem durch die mit Fritz Martini durchgeführte Gründung eines „Studium Generale" an der Stuttgarter Universität sichtbar werden, nimmt Bense direkten Einfluss auf verschiedenste Wissenschaftsbereiche.[31]

Meines Erachtens lassen sich Benses Intentionen schon recht gut alleine über die Titulierung seiner Schriften wie zum Beispiel *„Programmierung des Schönen"*, *„Semiotische Prozesse und Systeme in Wissenschaftstheorie und Design, Ästhetik und Mathematik. Semiotik vom höheren Standpunkt"* oder *„Die Unwahrscheinlichkeit des Ästhetischen und die semiotische Konzeption der Kunst"* ablesen.



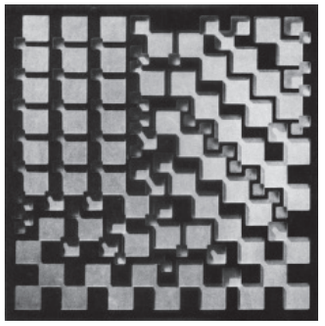


Abbildung 12: Georg Nees. Das Bild ist mit Fräse Aluminium geschnitten worden.

Abbildung 13: Georg Nees. Schotter

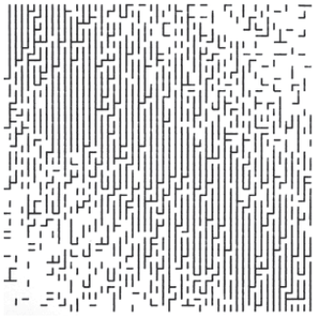


Abbildung 14: Frieder Nake. Walktrough

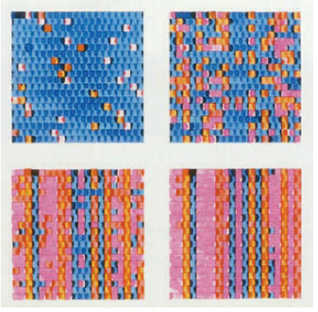


Abbildung 15: Frieder Nake. Plotter Zeichnungen

```
FUNCTION juliah(u,v)                                   (28)
  'Global f; LOCAL z,z1,s%; s%=0; z=(u,v)
  DO
    z1 = f(z)                                          (a)
    EXIT IF Cond(z1)                                   (b)
    EXIT IF ABS(nq(z1)-nq(z))<0.0001                   (c)
    z=z1; s%=s%+1
  LOOP
  RETURN s%                                            (d)
ENDFUNC
```

Abb. 8. Halleykarte. Darstellung der phlegmatischsten Punkte

Abbildung 16: Georg Nees. Das obere Bild zeigt das Ergebnis des kleinen Programmes darunter.

Als Experimente und Kunstwerke dieser Zeit lassen sich rund um Bense beispielsweise jene der Mathematiker und Computerkünstler Frieder Nake und Georg Nees anführen. Auch anderorts findet die Auseinandersetzung mit digitaler Kunst etwa in den Arbeiten von Herbert W. Franke oder Hajo Drott in München oder in den Werken von Bela Julesz und A. Michael Noll in New York statt. Die ersten Arbeiten wie die von Nake und Nees stützen sich auf Systemtheorien der Kybernetiker und beschäftigen sich besonders mit Pseudozufallsgeneratoren, der Nachahmung von Vorgängen in der Natur mittels Generativenprozessen und thematisieren auch das Problem der Autorenschaft dieser Kunstwerke. Einige dieser ersten Arbeiten findet man heute relativ einfach im Internet, wo man auch auf zahlreiche aktuelle Künstler der „Digital Art" stößt.[32]

Die unmittelbaren Auswirkungen und Reaktionen auf Benses Denkmodelle sollen hier zunächst anhand der Arbeit des Architekten Manfred Kiemle nähergebracht werden. Als ein Student und temporärer Mitarbeiter in Bense's Team am „Institut für Information" versuchte Kiemle mit Hilfe von CAD eine digitale Umsetzung der theoretischen Konzepte und veröffentlichte diese in seinem Buch „Ästhetische Probleme in der Architektur unter dem Aspekt der Informationsästhetik". In diesem Werk erreichte Kiemle eine kritische Erweiterung und teilweise Umsetzung von Bense's „Informationsästhetik" in eine algorithmische Logik. Über die ästhetische Kritik schreibt er wie folgt:

„Die ästhetische Kritik gibt kein Urteil über schön oder unschön ab, sondern sie macht Aussagen darüber, ob ein bestimmtes Kunstwerk in einem bestimmten soziokulturellen Raum Originalität, Semantik, usw. hat. Im Gegensatz zur ästhetischen Realisation ist die ästhetische Kritik im Prinzip programmierbar."[33]

Für die Erforschung des genannten soziokulturellen Kontextes welcher für die Interpretation der Information wesentlich ist schlägt er das Forschungsgebiet der „Soziokybernetik" vor. Ein Kunstwerk wird als eine Nachricht aufgefasst welche aus verschiedenen Zeichen besteht, die als Informationsträger an den Menschen gesendet wird, wobei dieser die Zeichen in ihrem Gesamtontext interpretiert. Überspitzt könnte man die Auffassung der Kybernetiker bezüglich des Menschen - besonders die Ansätze in Deutschland - folgend pointieren: „Die SMS „Kunst" wird über

Funkwellen an die Maschine „Mensch" gesendet, wobei das Signal über die Antenne „Fotorezeptoren" des Empfängermodems „Auge" aufgefangen wird und hier für den internen Datenbus „Nervenstrang" in Bitmuster „elektrochemische Impulse" codiert wird. Je nach Muster gelangen die codierten Zeichen über den Datenbus in verschiedenste untereinander vernetzte Postprocessinganlagen „Hirnareale" die sich schließlich je nach Interpretation für ein Handeln entscheiden und danach für eine Visualisierung „einen bewussten Gedanken" sorgen." Vielleicht wird nun auch verständlich, warum Forscher wie Kiemle etwas Angst vor dem Ersetzen des Menschen durch die Maschine hatten.

Kiemle und Ästhetik als Zahlenspiel

Trotzdem setzte Kiemle die Umsetzung der Theorie in der Praxis fort und versuchte zunächst die Kapazität des menschlichen Bewusstseins mittels mathematischer Formeln zu beschreiben. Dazu teilte Kiemle die Bewusstseinskapazität in eine Gegenwartsdauer „T", eine Zuflusskapazität „Ck" und eine Fassungskapazität „Kk" ein und stellte diese für die digitale Verarbeitung in Bits dar. Die Gegenwartsdauer ist die Zeit, in der ein Eindruck im Kurzspeicher „Bewusstsein" verweilt, wodurch dieser Reiz bewusst wahrgenommen wird. Die Wahrnehmungsdauer beträgt dabei 8 bis 12 Sekunden. Die menschliche Zeitauflösung beträgt eine 16tel Sekunde, was Rückschlüsse auf die maximale Zuflussgeschwindigkeit von Informationen zulässt. Dies bedeutet, dass Ereignisse die in diesem Zeitfenster stattfinden nicht in ihrer Reihenfolge unterschieden werden können und deshalb als Gleichzeitig wahrgenommen werden. Die Zuflusskapazität beschreibt wie viel unterschiedliche Zeichen der Mensch in einer bestimmten Zeit aufnehmen kann. Kiemle berechnet daraus eine Zuflusskapazität von 16 bit pro Sekunde. Multipliziert man die Zuflusskapazität (16 bit) mit der Gegenwartsdauer (10 Sek) bekommt man als Fassungskapazität des menschlichen Bewussteins 160 bit. Weitere Begriffe, wie der Überraschungswert, der Auffälligkeitswert, die Superzeichen oder die Entwicklung seiner vier notwendigen Kriterien für Schönheit, die für die Auffassung der „Architektur als Nachricht" unerlässlich sind, werden in dieser Untersuchung nicht weiter erläutert.[34]

Anhand eines praktischen Beispiels nahm Kiemle eine ästhetische Kritik an der Fassade der Technischen Hochschule Stuttgart vor, womit er seine Überlegungen zu verifizieren versuchte. Dabei demonstrierte Kiemle den wechselnden ästhetischen Informationsgehalt der regelmäßigen Fassade durch das Ein- und Ausschalten von Lichtern. Die gesamt Fassade enthält 216 Fensteröffnungen. Jedes Fenster enthält die Information von einem Bit, was für die gesamte Fassade einen maximalen Informationsgehalt von 216 bit ergibt. Kiemle versteht die Fassade als Bitmuster, welches er nun in einem Bereich von maximaler Ordnung und einem Grenzmaß an maximaler Unordnung untersucht. Dabei berechnet Kiemle auch die Redundanz die die Ordnung einer gegliederten Elementenmenge angibt und für die ästhetische Beurteilung wichtig ist. Ein großen Wert an Redundanz wird als langweilig interpretiert.[35]

„Wenn ein beträchtlicher Teil der Architektur einer Epoche zu informationsarm ist, dann scheint eine Entwicklung einzusetzen, die eine Informationssteigerung herbeiführt. So kann die informationsästhetische Wahrnehmungstheorie eine Erklärung für gewisse Erscheinungen in der neuesten Architektur liefern"[36].

Hingegen kann bei vielen verstreuten Elementen der Informationsgehalt größer als 160 bit sein, also größer als die Fassungskapazität des Bewussteins, dann muss eine superiert werden. Dies bedeutet, dass durch das Zusammenfassen von Einzelelementen zu größeren Elementen, die Redundanz erhöht wird, wodurch schließlich, der Informationsgehalt gesenkt wird. Weiters beschreibt Kiemle, dass ein Gebäude in verschiede Betrachtungsstufen und Betrachtungsschritte eingeteilt werden kann. Nach jedem Betrachtungsschritt wird das erkannte Muster in einem *„Superzeichen"* zusammengefasst, wobei nun im nächsten Schritt weitere Details am Gebäude bemerkt werden können. Je mehr solcher Ebenen ein Gebäude besitzt, desto öfter vollzieht sich der Wahrnehmungsprozess und desto länger bleibt dieses für den Betrachter spannend. Kiemle deutet eine hohe Anzahl von Betrachtungsebenen als einen Grundcharakter des Schönen.[37] Die Originalität bzw. den Stil eines Künstlers beschreibt Kiemle über eine hohe Informationsdichte auch in der Mirkostruktur. In diesem Zusammenhang deutet Kiemle in Anlehnung an Bense die Innovation als den zentralen Begriff für den ästhetischen Prozess.
„Die Originalität ist eine Funktion der Unwahrscheinlichkeit einer Zeichenstruktur, wobei sich Unwahrscheinlichkeit auf die subjektive Information einer bestimmten soziokulturellen Gruppe während eines bestimmten Zeitraumes bezieht. Je unwahrscheinlicher, unerwarteter und überraschender eine Zeichenstruktur für eine solche Gruppe ist, desto größer ist ihre Innovation."[38]

In einem weiteren Beispielprojekt, der Fassade des Rathauses in Kurashiki von Knezo Tange, kommt Kiemle zu dem Schluss, das diese mit einer Redundanz von ca. 70 bis 80 Prozent auch in der vierten Betrachtungsstufe „nicht der Eindruck der Langweiligkeit" wie etwa bei der Fassade der Technischen Hochschule in Stuttgart (mit 98,3 Prozent) aufkommt.[39]

Claus Pias pointiert diese Auseinandersetzung rund um Bense in Deutschland in seinem Aufsatz in dem Buch Kulturtechnik Entwerfen in Anlehnung an Friedrich Kittler wie folgt etwas spitz:

„Es handelt sich um die Austreibung des Geistes aus den Geisteswissenschaften"[40].

Das Bestreben zu einer Verwissenschaftlichung der Architektur führt Pias zum einen auf die nationalsozialistische Vergangenheit Deutschlands zurück, die in dem Reinheitsgebot der Ulmer Schule eine Verbannung des Gefälligen äußerte, sowie zum anderen auf die Hoffung, durch die Versprechen der Kybernetik eine Vereinheitlichung der vielen zerstreuten Wissenschaften zu erreichen.[41]

Zusammenfassend zu diesem Kapitel kann gesagt werden, dass sich rasch herausstellte, dass Bense's Forderungen nicht so einfach umsetzbar sind, da in vielen Fällen Unklarheit darüber die Möglichkeit einer mathematischen Problembeschreibung herrschte. *„Wie konnten analytische, perzeptive und kreative Prozesse in Algorithmen transponiert werden?"*[42] In vielen Fällen wurde auf die noch wenig ausgereifte Technik und die marginale Anzahl von Arbeiten in den verschiedensten Wissenschaftsdisziplinen verwiesen, auf deren Forschungsergebnisse bei der Entwicklung von komplexen Systemen zurückgegriffen werden muss. So werden diese Themen heute aufgrund der gigantischen Entwicklungen und Kapazitäten digitaler Medien unter dem Motto „Aufgeschoben ist nicht Aufgehoben!" mit wenigen Zusätzen neu diskutiert. Doch für die aktuelle Auseinandersetzung gilt unter anderem nach wie vor:

„Noch niemand hat nun allerdings die Funktion einer Wohnung oder eines Büros erschöpfend beschrieben."[43]

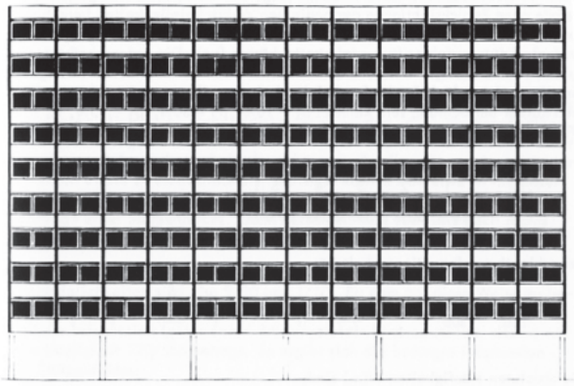


Abbildung 17: Manfred Kiemle. Fassade der Technischen Hochschule Stuttgart. Die Fassade sei informationsarm und deshalb langweilig.

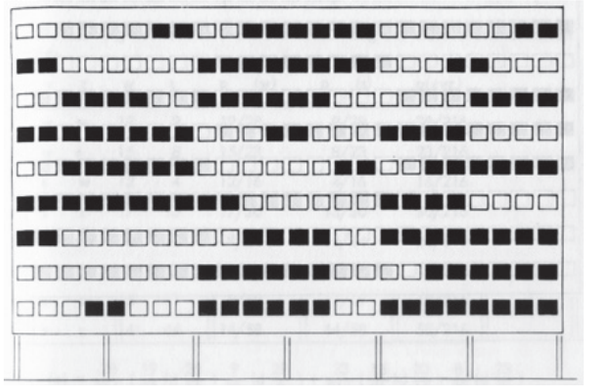


Abbildung 18: Jedes Fenster enthält die Information von einem Bit was für die gesamte Fassade einen maximalen Informationsgehalt von 216 bit ergibt.

14 Vgl. Marco Hemmerling, Anke Tiggemann, "Digitales Entwerfen", 2010, Wilhelm Fink GmbH & Co. Verlags-KG, S.12 – 15.

15 Michael Friedewald, "Der Computer als Werkzeug und Medium", Verlag für Geschichte und Naturwissenschaften und der Technik, Berlin, Diepholz 1999, S.74-78.

16 Vgl. ebd. S.51-69.

17 Ebd. S.69.

18 Vgl. David E. Weisberg, "The Engeneering Design Revolution", e-Book 2008, http://www.cadhistory.net (28.02.2010), Abschnitt 3: Computer-Aided Design Strong Roots at MIT, S.1-25.

19 Claus Pias, "Jenseits des Werkzeugs", Kulturtechnik Entwerfen., a. a. O., S.279. oder http://www.encyclopedia.com/doc/1O11-rubberbanding.html (Stand 14.02.2010)

20 Claus Pias, "Jenseits des Werkzeugs", Kulturtechnik Entwerfen, a. a. O., S.282.

21 Ebd. S.278.

22 Michael Friedewald, "Der Computer als Werkzeug und Medium", Verlag für Geschichte und Naturwissenschaften und der Technik, Berlin, Diepholz 1999, S.54

23 Vgl. Nicolas "The Archtecture Machine", Nicholas Negroponte, The MIT Press Cambridge, Massachusetts, 1970, S.6. und http://de.wikipedia.org/wiki/Head-Mounted_Display (Stand 17.02.2010).

24 Claus Pias, "Jenseits des Werkzeugs", Kulturtechnik Entwerfen, a. a. O., S.278-299.

25 William J. Mitchell, "The Logic of Architecture: Design Computation and Cognition", MIT Press, 1990.

26 Vgl. Christian Kühn, "Erste Schritte zu einer Theorie des Ganzen", Kulturtechnik Entwerfen, a. a. O., S.161-177.

27 Vgl. Christopher Alexander, "A Pattern Language", Oxford University Press, New York 1977. Zitiert nach der im Seminar für Planungsmethoden entstandenen Übersetzung, Klaus W. Gartler, Technische Universität Graz 1979, Einleitung.

28 Nicholas Negroponte, "Being digital", Vintage Books a division of Random House, 1995 USA, S.6-7.

29 Vgl. Ingeborg M. Rocker, "Berechneter Zufall", Kulturtechnik Entwerfen, a. a. O., S.245-259.

30 Ebd. S.252.

31 Vgl. ebd., S.245-259.

32 Vgl. ebd. S.246., http://www.biologie.uni-muenchen.de/~franke (Stand 19.02.2010), http://www.computerkunst-drott.de (Stand 19.02.2010), http://hyperallergic.com/5105/early-computer-art, (Stand 19.02.2010).

33 Manfred Kiemle, "Ästhetische Probleme der Architektur unter dem Aspekt der Informationsästhetik", Verlag Schnelle Quickborn, Deutschland, 1967, S.13.

34 Claus Pias, "Jenseits des Werkzeugs", Kulturtechnik Entwerfen, a. a. O., S.256-258.

35 Ebd.

36 Ebd. S.99.

37 Vgl. ebd.

38 Ebd., S.49.

39 Ebd., S.108.

40 Orignalquelle,Friedrich Kittler, "Austreibung des Geistes aus den Geisteswissenschaften. Programme des Poststrukturalismus." Paderborn: Schöningh 1980. Originalquelle wurde in Anlehnung an Claus Pias, "Jenseits des Werkzeugs", Kulturtechnik Entwerfen, a. a. O., S.271 zitiert.

41 Claus Pias, "Jenseits des Werkzeugs", a. a. O., S.277.

42 Ingeborg M. Rocker, "Berechneter Zufall", Kulturtechnik Entwerfen, a. a. O., S.253.

43 Georg Frank, "Maschinelle Entwurfshilfen", Kulturtechnik Entwerfen, a. a. O., S.238.

Die Entwicklung von CAAD-Software zur Verwendung als Scriptingtool

Der Einfluss der Fertigungsindustrie auf die Entwicklung von CAD-Software

Im Jahr 1949 erhielt Professor Gordon Brown, der Gründer des „Servomechanism Laboratory" am MIT einen Anruf von John T. Parsons. In diesem Telefonat bekundet Parsons, von der Firma „Parsons Corporation's Aircraft Division" in Michigan (ein Zulieferunternehmen für Flugzeugteile) ein Interesse an der Entwicklung einer computergesteuerten maschinellen Fertigung von Flugzeugbauteilen. Zur selben Zeit entwickelte das US-amerikanische Militär eine neue Generation von Kampfflugzeugen, bei welcher eine hohe Genauigkeit bei der Herstellung der Metallteile gefordert war.
So startete am MIT, parallel zu den Arbeiten an dem weiter oben im Text genannten Projekt „Whirlwind", die Forschung an der Herstellung von numerisch gesteuerten Fräsmaschinen, in enger Zusammenarbeit mit der Firma „Parsons Corporation's Aircraft Division" mittels Finanzierung durch die amerikanische Luftwaffe „Air Force". Aus diesen NC-Maschinen („NC" steht für „Numerical Control") entwickelte sich im Jahr 1972 die erste CNC-Maschine („CNC" steht für „Computerized Numerical Control" im Deutschen „computerisierte numerische Steuerung").

Damit war eine von zwei wichtigen Voraussetzungen gegeben, um das Interesse großer Fertigungsindustrien an der Forschung von CAD und den Einsatz dieser Technologie zu wecken. Der zweite Schritt bestand in der Erweiterung von Sutherlands „Sketchpad", um eine dritte Dimension durch Tim Johnson, welche noch im Jahr 1963, dem Veröffentlichungsjahr von Sutherlands Projekt, erfolgte. Ab diesem Zeitpunkt erkannten große Fertigungsfirmen langsam das Potential, das in den Möglichkeiten von CAD in der Kombination mit CAM („Computer Aided Manufacturing") für den Auto- oder Flugzeugbau steckt. Natürlich ist am MIT nach der Entwicklung von Sutherlands „Sketchpad" weiter an den Möglichkeiten eines praktischen Einsatzes von CAD geforscht worden, doch bestand der nächste große Fortschritt, welcher für den Umgang mit Freiformen und den Praxisbezug wesentlich war, in der Auseinandersetzung der großen Fertigungsindustrien mit „CAD" bzw. „CAM".

Während am MIT an der Entwicklung der ersten CAD-Software, welche beispielsweise für das Zeichnen von elektronischen Schaltungen gedacht war, gearbeitet wurde, beschäftigten sich Flugzeug- und Autohersteller wie General Motors, Renault, Citroen oder Lockheed, mit der mathematischen Beschreibung von komplexen Formen und deren computergestützten automatisierten Herstellung.

In der Anfangsphase lassen sich zwei unterschiedliche Herangehensweisen der Firmen ausmachen, die sich allerdings spätestens in den 1980er Jahren in einzelnen Produkten wieder vereinigen. David Weisberg schreibt dazu:

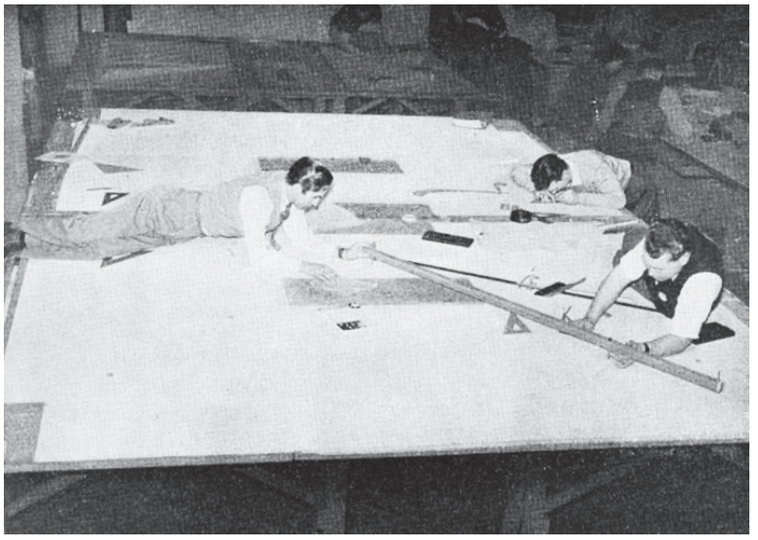


Abbildung 19: Einen Masterlayoutplan für ein Flugzeug zu erstellen war ein großer Aufwand und benötigte viel Personal.

"This early work fell into two categories. On one hand, automotive companies such as Renault and Ford focused on the mathematical definition of complex surfaces while other companies, such as Lockheed California focused on improving drafting productivity."[44]

Beispielsweise begann der Autohersteller General Motors (in Folge: „GM") Ende der 1950er Jahre in den USA an dem Projekt „DAC" (Design Augmented by Computers) zu arbeiten. Im Gegensatz zu dem Konzept von Lockheed war DAC schon von Anfang an auf eine dreidimensionale Anwendung ausgerichtet. Als Sutherlands Sketchpad publik wurde, hatte GM mit DAC bereits eine ähnliche Oberfläche entwickelt und konnte um 1964 bereits komplexe dreidimensionale Geometrien konstruieren und darstellen. Um 1970 wurde auch eine kommerzielle Software mit dem Namen „CGS" (Corporate Graphic System) entwickelt. CGS war allerdings weniger erfolgreich und wurde bereits in den 1990er Jahren von anderen Softwareprogrammen vom Markt verdrängt und findet heute nur noch vereinzelt Verwendung.

Eine der einflussreichsten Entwicklungen für die Entwicklung von CAAD-Software war das Projekt CADAM (Computer-graphics Augmented Design and Manufacturing), welches von dem Flugzeughersteller Lockheed entwickelt und später über IBM bereits in den 1970 Jahren an mehrere Firmen auch in Europa verkauft wurde. Trotz des großen Erfolges Ende der 1980er Jahre wurde CADAM im Jahr 1991 von Lockheed nach finanziellen Schwierigkeiten an Dassault Systèmes verkauft. Für Lockheed war CADAM nicht mehr maßgebend, da ein Großteil der Planungen ohnehin bereits mit der Software "CATIA" von Dassault Systèmes, einem Unternehmen aus Frankreich, bearbeitet wurde.

Die Entwicklung von „CATIA" (Computer-Aided Three-Dimensional Interactive Application) geht auf die Arbeiten von Pierre Étienne Bézier (1910-1999) zurück. In den 1960er Jahren beschäftigt sich Bézier als Angestellter des Autoherstellers Renault in Frankreich mit der mathematischen Beschreibung von freien Kurvenlinien. Er entwickelte Formeln mit denen diese geschwungenen Linien anhand von anliegenden Tangenten und wenigen Punkten (auch „Kontrollpunkte" genannt) interpoliert werden konnten. Die Kurvenlinien werden segmentweise in eine Kette von Polynomen unterteilt, womit die Kurvengeometrie mathematisch angenähert werden kann.

Diese Theorie steckt heute in nahezu jedem CAD-Softwareprogramm, welches diese für die Berechnung von beliebig gekrümmten Linien oder in einer Weiterentwicklung für die Darstellung von Flächen im Raum verwendet. Die Linien werden gängigerweise als „B-Splines" (Bézierkurven) und die Flächen als „NURBS" oder auch „Non-Uniform Rational B-Splines" bezeichnet. Das Wort „Spline" kommt aus dem Englischen und wird mit *„Kurvenlineal, Schiebekeil oder Fugenbrett"*[45] ins Deutsche übersetzt. Im Kontext von CAD-Anwendungen nimmt der Begriff einen Bezug zu den Kurvenlinealen, die in der jahrhundertelangen Tradition des Schiffbaus eingesetzt wurden, um dynamisch geschwungene Formen konstruieren zu können. Je nach dem, welchen Ordnungsgrad die verwendete Polynomkette aufweist wird zwischen unterschiedlichen Typen unterschieden. Ein Splinetyp „erster Ordnung" würde beispielsweise einen Polygonzug mit einer geraden Linienführung zwischen den Kontrollpunkten ergeben und daher als „lineare Spline" bezeichnet werden. Weiters werden je nach Ordnungstyp auch Namen wie „quadratische, kubische oder auch B-Spline" vergeben.[46]

Nach dem Kauf von CADAM durch Dassault Systèmes wurden einige Elemente in die Software CATIA eingebunden. In der Kombination der Zeichentechniken von CADAM und der verbesserten 3d-Modellierung von CATIA wurde CATIA zu einem der heute vor allem im Maschinenbau populärsten Softwareprogramme.[47]

Spezialisierung von CAD und die ersten CAAD-Anwendungen

In den 1980er Jahren setzte Frank O. Gehry als einer der ersten Architekten die CAD-Software „CATIA" in der Architektur ein. Zu diesem Zeitpunkt setzte langsam eine nutzerspezifische Spezialisierung durch einzelne Programmaufsätze ein, wodurch sich eine „zweite Generation" von CAD-Programmen entwickelte. Bis dahin verwendeten beispielsweise MaschinenbauerInnen und ArchitektInnen dieselben Softwareprogramme.[48]

Aus diesem Vorgang lassen sich zwei Tendenzen ablesen:
Zum Ersten zielen Fertigungsindustrien wie Flugzeughersteller, Automobilhersteller oder auch die Maschinen-, Werkzeug-, Formen-, Modellbauer wie auch Elektronikhersteller verstärkt auf die automatisierte Herstellung ihrer Produkte mittels NC-Verfahren, wobei die Herstellung komplexer Formen und auch die Simulation und Analyse von dynamischen Prozessen eine große Rolle spielt. An dieser Stelle wird nicht weiter auf die Entstehungsanfänge ähnlicher Softwareprogramme wie Bravo Sketcher, Unigraphics, Solid Edge, SolidWorks, Pro/ENGINEER, Inventor, etc. eingegangen, da hier das Augenmerk auf die Entwicklung von CAAD-Software gerichtet sein soll.

Wie bereits im vorherigen Kapitel erläutert, steht die Abkürzung „CAD" für „Computer Aided Design", wobei das zweite „A" in CAAD für die speziellere Anwendung im Architekturbereich steht.[49] So setzt sich nun die zweite Richtung tendenziell mit den Bedürfnissen rund um die Planung von Architektur auseinander. Auch Mike Riddle war im Jahr 1982 als einer der zahlreichen Mitbegründer der Firma „Autodesk", welche heute eines der erfolgreichsten Unternehmen in der CAAD- und auch CAD Branche darstellt, in den Anfängen zeitweise für das Architekturbüro „Frank Lloyd Wright Foundation" in Arizona tätig. Bei Produkten wie „AutoCAD" (CAAD-Software von Autodesk) wird der Computer als cleveres Zeichenbrett verstanden, wobei eine dreidimensionale Darstellung von Gebäuden vorläufig sekundär war. Ehemals transparente, übereinander gelegte Papierblätter wurden nun als eine Anzahl von „Layern" oder auch „Ebenen" interpretiert, sodass diesen verschiedene Stifte bzw. Linieneigenschaften zugewiesen werden konnten. Zudem waren für ArchitektInnen neben der Erstellung von Plänen auch Werkzeuge für Grundrissflächenberechnungen oder auch integrierte Material- bzw. Zeichnungsbibliotheken für eine bessere Kosten- und Planungsübersicht von Bedeutung.[50]
Dies führte zur Entwicklung der heute weitverbreiteten CAAD-Programme wie beispielsweise Allplan (Nemetschek), ABiS (ABIS Softwareentwicklungs Ges.m.b.H.), AutoCAD (Autodesk), ArchiCAD (Graphisoft), EliteCAD (Roland Messerli AG INFORMATIK), MicroStation (Bentley Systems) oder Revit (Autodesk).[51]

Mittels dieser CAAD-Programme konnten Pläne schnell und einfach vervielfacht werden und mit jedem gewünschten Detaillierungsgrad zwischen den unterschiedlichsten an der Planung beteiligten Gewerken ausgetauscht werden. Durch den Einsatz dieser neuen Planungsmethoden wurde an Personal und Zeit gespart. Durch Verwendung von vorgefertigten parametrisierten Objekten wie Wände, Fenster und Treppen konnten Gebäude rasch gezeichnet und durch fertige Bibliotheksobjekte wie Betten, Tische oder Stühle ergänzt werden, wobei etwaige Veränderungen mit einem relativ geringen Arbeitsaufwand verbunden waren. Somit wurde auch eine neue Art des Entwerfens gefördert. Das Arbeiten mit Kopien erleichterte zudem das Erstellen unterschiedlicher Varianten. Dies hat umgekehrt den Auftraggeber zur Tendenz verleitet, vom Planer immer mehr Varianten und Entwurfsversionen je Projekt zu verlangen. Daher ist in manchen Fällen die Zeit- und Arbeitsersparnis auf ein gesamtes Projekt gesehen möglicherweise marginal. CAAD-Software wird heute als Standard für die Umsetzung von Architekturkonzepten verwendet und weltweit in nahezu allen Architekturbüros in den Planungsablauf integriert.[52]

Die Entwicklung von Animationssoftware

Gegen Ende des 20. Jahrhunderts trat die Visualisierung als eine bis dahin sekundäre Komponente von CAD in den Vordergrund und führte zum Aufkommen einer neuen Generation von Softwareprogrammen. Nachdem bereits um 1960 sowohl durch die Arbeiten von insbesondere dem Franzosen Pierre Èttienne bei Renault als auch durch jene von Paul de Casteljau bei Citroën eine mathematische Beschreibung beliebiger Freiformobjekte ermöglicht worden war, konnte nunmehr an einer verbesserten Darstellung der Objekte gearbeitet werden. In einem der ersten Schritte wurden die bisher verwendeten Drahtgitterdarstellungen durch Vollkörperdarstellungen ersetzt. Dies konnte mit Hilfe des im Jahr 1974 von Edwin Catmull erfundenen „Z-Buffer" - Algorithmus umgesetzt werden. Unter Verwendung des sogenannten „Z-Buffers" konnte die Tiefe im dreidimensionalen Raum berücksichtigt werden, wodurch ermittelt werden konnte, ob Elemente von anderen Objekten verdeckt werden und folglich sichtbar oder unsichtbar zu zeichnen sind. Weiters veröffentlichte Bui-Toung Phong im Jahr 1975 seine Erfindung des „Phong shading". Ein Jahr darauf präsentierte auch James F. Blinn seine Forschungsergebnisse über das „Reflection / Enviroment Mapping". Mithilfe dieser Techniken war bereits eine sehr plastische Darstellung von beliebigen dreidimensionalen Körpern möglich.

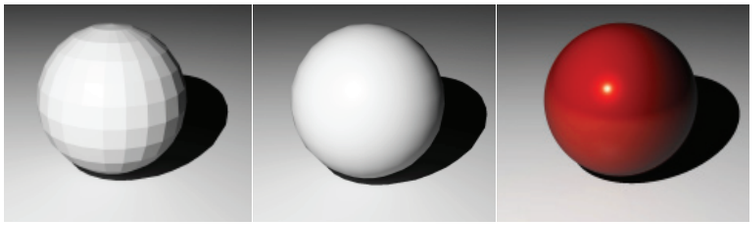


Abbildung 20: Kugelobjekt ohne Phong Shader, mit Phong-Shader und schließlich mit Farbe, Phong-Shader und Reflexionen.

Zunehmend entdeckte die Filmindustrie diese grafischen Errungenschaften für ihr Gebiet und regte über eine starke Wechselbeziehung die Entwicklung voneigenständigen Visualisierungs- und Animationsprogrammen a n. Bald wurden einige der heute bedeutendsten Animationsstudios wie Lucas Films (1979), Pixar (1985) oder DreamWorks SKG (1994) gegründet, wobei parallel ab Anfang der 1980er Jahre die ersten digitalen Animationssoftwareprogramme wie Wavefront Advanced (1984, Wavefront Technologies) oder Softimage 3d (1988, Softimage Co.) präsentiert wurden. Die erste kommerzielle Animationssoftware wurde ab dem Jahr 1984 von der Firma „Wavefront Technologies" vertrieben. „Wavefront Technologies" fusionierte im Jahr 1995 mit der Firma „Alias" und brachte mit der Software „Alias Wavefront Maya" im Jahr 1998 eines der heute erfolgreichsten 3d-Animationspakte auf den damaligen Markt.

Der erste Film, in dem bereits 15 Minuten ausschließlich digital generiert worden waren, wurde von Disney mit dem Titel „TRON" im Jahr 1982 präsentiert. Der erste komplett animierte Film war „Toy Story", welcher im Jahr 1995 nach einer vier Jahre langen Zusammenarbeit zwischen „Pixar" und „Disney" hervorging. Für die Weiterentwicklung von „Computer Graphic Software" (in Folge: „CGS") waren Animationen wie etwa die Kurzfilme „The Adventures of André and Wally B" (1984, Pixar) oder „Luxor Jr." (1986, Pixar), animierte Szenen aus „The Abyss" (1989), „Terminator II" (1991), oder „Jurassic Park" (1993) sowie Spezialeffekte in Fernsehserien wie „Babylon 5" (1993-1998) oder „SeaQuest DSV" (1993-1996) maßgebend. Ab den 1990er Jahren erschienen zahlreiche Animationsprogramme, von denen heute noch einige in weiterentwickelten Versionen erhältlich sind.[53]

Grundsätzlich sind die meisten dieser Programme heute sehr ähnlich aufgebaut.[54] Wie allgemein bei CAD-Software im 3d Bereich wird der Hauptarbeitsbereich je nach Wunsch meist aus einem oder mehreren Bearbeitungsfenstern zusammengesetzt, wodurch ein Element meist aus mehreren Blickwinkeln gleichzeitig betrachtet bzw. editiert werden kann. Je nachdem, ob gerade an der Modellierung, der Visualisierung oder der Animation der Objekte gearbeitet wird, stehen vorwiegend ähnliche Werkzeuge zur Verfügung. Für die Modellierung stehen üblicherweise einige einfache Grundobjekte, sogenannte „Primitives" (z.B.: Würfel, Kugeln, Pyramiden, Zylinder, Ringe, etc.) zur Verfügung, die mittels Grundfunktionen (z.B.: Bool'sche funktionen) miteinander kombiniert werden können. Bei komplexeren Modellierungsaufgaben gibt es unterschiedlichste Spline- und NURBS-Objekte, die mittels unterschiedlichsten Modifikatoren (z.B.: Extrusionsmodifier, LoftNURBS, SweepNurps, LatheNurbs, RailSplineObject) die Erstellung von komplizierten Geometrien ermöglichen. Bei der Visualisierung kann meist auf eine Materialbibliothek und unterschiedlichste Lichtquellen zurückgegriffen werden. Natürlich können auch eigene Materialien mit speziellen Farb-, Reflexions-, Transparenzeigenschaften etc. kreiert werden. Um die Objekte entsprechend zu animieren, können beispielsweise unterschiedlichste Partikelsysteme, Physik Engines, Skriptsprachen und zahlreiche Eingabemöglichkeiten eingesetzt werden.

Von Beginn an lag der Fokus bei der Entwicklung dieser Anwendungen auf Themen wie der einfachen Modellierung von einzelnen Geometrien mitsamt ihren Umgebungen, der photorealistischen Darstellung von Szenen und den Animationstechniken, die jeweils auch physikalische Eigenschaften der Objekte berücksichtigen können, um bei Bedarf eine möglichst wirklichkeitsgetreue Simulation der realen Gegebenheit erreichen zu können. Um komplexere Formen bzw. Bewegungsabläufe nicht manuell mittels Maus und Tastatur modellieren zu müssen, wurde ständig an der Verbesserung der Eingabesysteme gearbeitet. So konnten beispielsweise die Geometrien von Fabelwesen mit der Unterstützung von digitalisierten Handskizzen konstruiert oder durch Abtasten eines physisch realen Vorlagemodells mittels 3d-Laserscanners automatisch generiert werden. Um möglichst reale Gesichtsmimiken oder Körpergesten für die virtuellen Charaktere zu erhalten, wurden Sensorsysteme wie beispielsweise „Motion Tracking" - bzw. „Motion Capturing" - Systeme entwickelt, die die Bewegungen von Schauspielern aus Fleisch und Blut digitalisieren können.

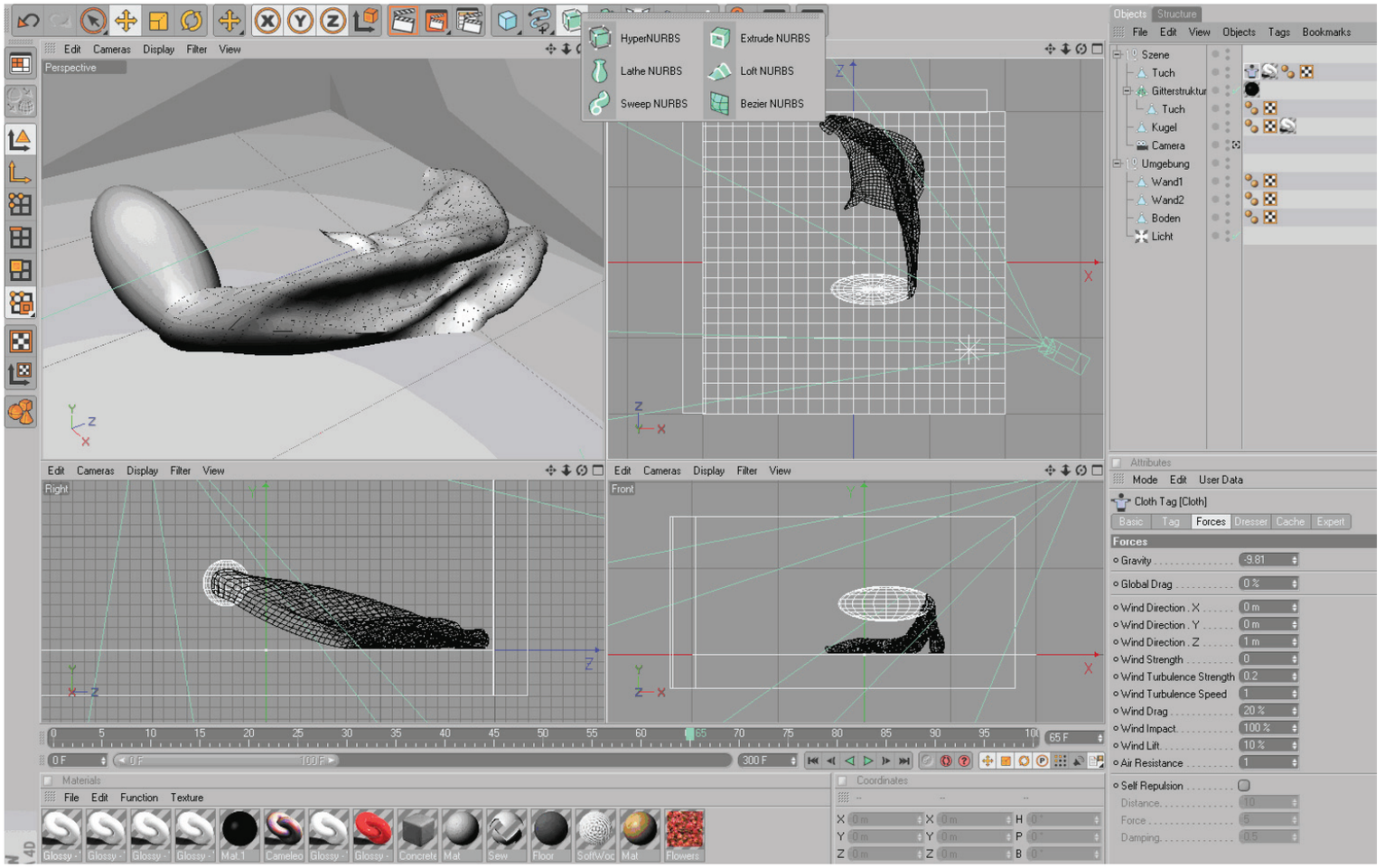


Abbildung 21: Benutzeroberfläche einer Animationssoftware mit vier Bearbeitungsfenstern, Materialien (unten) , Modellierungswerkzeugen (oben Mitte), Objektverwaltung (oben Links), etc.

In diesem Zusammenhang wurden auch Gelenkssysteme entwickelt, womit die 3d-Objekte gelenkig untereinander verbunden und damit auch als virtuelle Marionetten ähnlich wie reale Puppen verwendet werden konnten. Bereits im Jahr 1983 entwickelte William T. Reevers im Rahmen des Films „Star Trek - Der Zorn des Kahn" die ersten „Partikelsysteme". Diese werden für die gleichzeitige Animation von vielen Objekten verwendet, wobei die einzelnen Elemente zusätzlich untereinander in Abhängigkeiten gesetzt werden können. So können mit diesen Systemen Naturphänomene wie Feuer, Nebel, Rauch, Flüssigkeiten oder auch das intelligente Verhalten von Vogel- oder Fischschwärmen simuliert werden. Damit Ereignisse wie Kollisionen und deren Folgen ebenfalls mit geringem Aufwand nachgestellt werden konnten, wurden sogenannte „Physik Engines" entwickelt, um diese Vorgänge nun realitätsnahe simulieren zu können. Somit konnte die virtuelle Umgebungswelt durch Unterstützung der „Physik Engines" mit Gravitation oder Wind ausgestattet und den Szeneobjekten Masse-, Reibungs-, Elastizitätseigenschaften etc. zugewiesen werden. Um individuelle Objekts- bzw. Geometriezusammenhänge berücksichtigen zu können, wurden einfache Programmiersprachen - „Skriptsprachen" oder unterschiedliche Vernetzungsplugins - „Node-Systeme" erfunden.

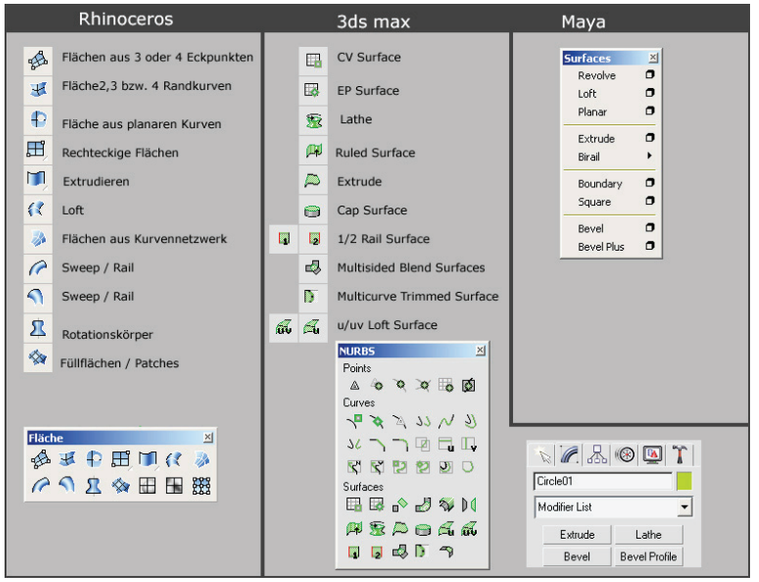


Abbildung 22: Standardwerkzeuge in drei unterschiedlichen Animationsprogrammen für die Modellierung von Grundobjekten.

Mit den genannten Hilfsmitteln konnte auf vordefinierte Eigenschaften von Objekten zugegriffen werden, wobei diese folglich über kleine Algorithmen oder über ein logisches System wie intelligente Knoten, „Nodes", untereinander verknüpft werden konnten. Damit musste beispielsweise der Fuß eines Radfahrers nicht mehr extra animiert werden, da die Fußobjektsposition in Abhängigkeit zur Position des Fahrradpedals gesetzt werden konnte. Hier sei kurz erwähnt, dass all diese vielschichtigen Entwicklungen auch für die Umsetzung von Computerspielen interessant waren. Aus diesem Grund waren Animationsprogramme schon von Anfang an mit der Computerspielherstellung verknüpft, sodass einige Hersteller etwa auch Konzepte für die Navigation und Echtzeitdarstellung von virtuellen Welten erarbeiteten. Heute gibt es eine nahezu unendliche Anzahl von verschiedenen „Plugin's", womit diese Programme erweitert werden können. Somit wurde die Animationssoftware zu einem leistungsstarken universellen Experimentierlabor, in dem noch nie dagewesene Situationen simuliert, analysiert und visuell erlebbar gemacht werden konnten.[66]

Auszug einiger Visualisierungs- bzw. Animationsprogramme:

- Autodesk 3d Studio Max - Autodesk
  1990 3d Studio DOS von Antic Software
  ab 1996 3d Studio Max
  http://usa.autodesk.com/

- Autodesk Maya - Autodesk
  Ursprünglich 1998 Alias Wavefront Maya,
  hervorgegangen aus der Kombination zweier Programme:
  1. Wavefront Advanced (1984 von Wavefront Technologies),
  2. Alias PowerAnimator (1991, Alias)
  http://usa.autodesk.com/

- Autodesk Softimage - Autodesk
  1988 Softimage 3d von Softimage, Co
  http://usa.autodesk.com/

- Blender - NeoGeo
  seit 1989
  Blender ist eine extrem leistungsfähige Opensource Software
  http://www.blender.org/

- Electric Image - Electric Image, Inc
  http://www.eitechnologygroup.com/

- Cinema 4D - Maxon
  1991 FastRay
  seit 1993 Cinema 4D
  seit 2000 70% der Gesellschaftsanteile
  von Maxon gehören der Nemetschek AG
  http://www.maxon.net/

- LightWave 3D - New Tec Inc
  1990 LightWave 3d
  http://www.newtek.com/lightwave/

- Rhinoceros - McNeel
  1993 Skultura von Applied Geometry
  seit 1994 Rhinoceros
  http://www.rhino3d.com/

- Truespace - Microsoft
  1986 Anfangs von Octree
  später von Caligari
  Integriert in Microsoft Virtual Earth
  http://www.caligari.com/

Abbildung 23-24: Animation einer Flüssigkeit, die auf einen Kubus auftrift und dabei einen Zylinder wegschießt.

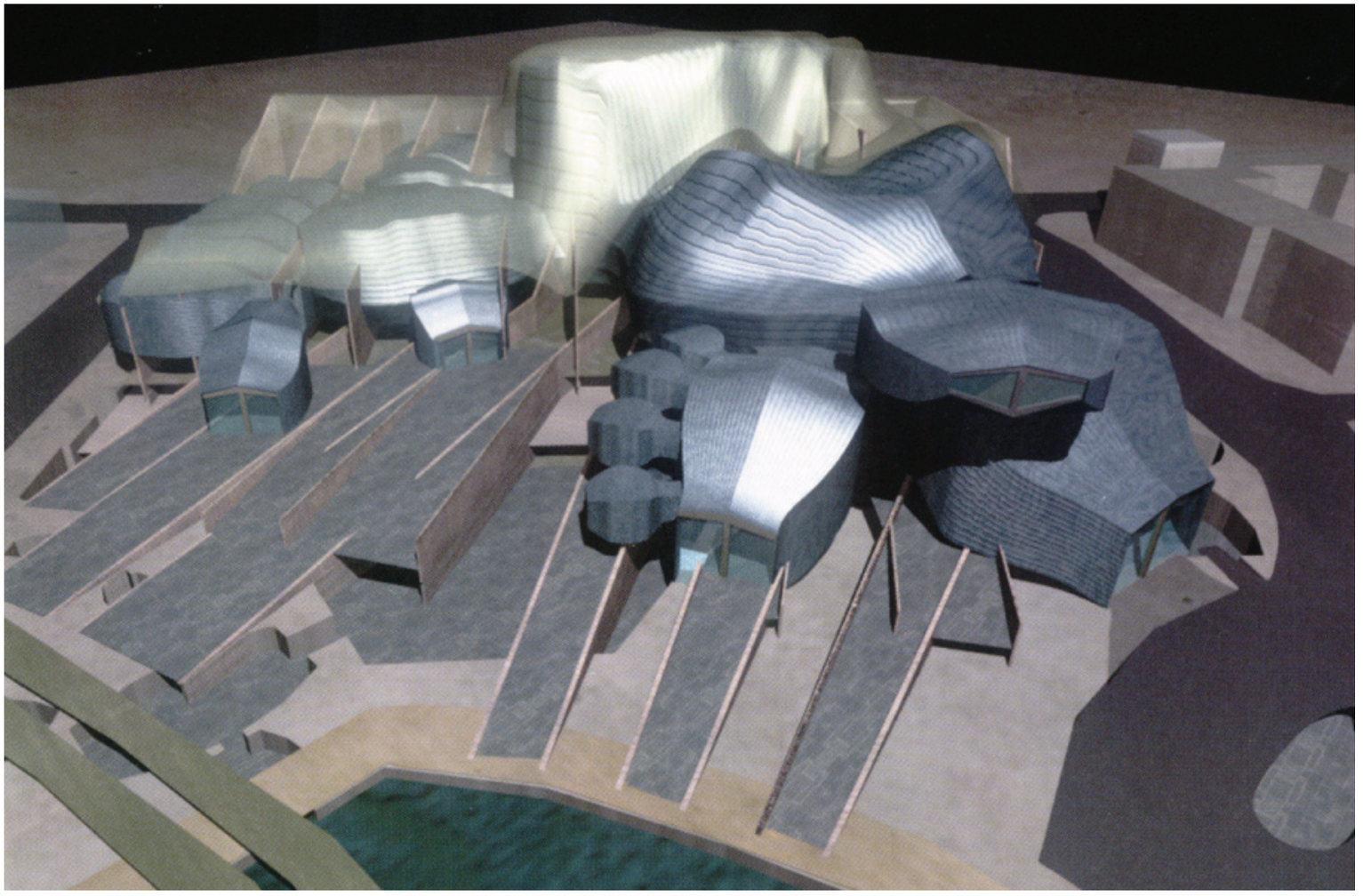


Abbildung 25: Greg Lynn. Modell des Cardiff Bay Opera House aus „Animate Form" – 1999.

CGS Software und der Einzug in CAAD

Vor allem die rasanten Fortschritte im Grafikbereich führten am Anfang des 21. Jahrhunderts dazu, dass die im vorherigen Abschnitt erwähnten Anwendungen mehr und mehr in den Architekturbereich Einzug nahmen. Einer der ersten Architekten, der mit diesen neuen Techniken experimentiert hatte und damit auch heute noch nach wie vor Formfindung betreibt, war Greg Lynn. Bereits 1999 veröffentlichte er sein Buch "Animate Form", in dem er seine Visionen von bewegten, flüssigen, verdrehten, verzerrten Formen, die mittels Splines und Hyperflächen modelliert werden sollten, beschrieb und einige seiner Projekte vorstellte. Greg Lynn hatte auch wesentlich zur Verbreitung des aus der EDV kommenden Begriffs „Blob" (Binary Large Objekt) beigetragen, mit dem diese neuen Formen und Objekte heute häufig bezeichnet werden. Weitere Architekten, die sich relativ früh mit diesen Programmen auseinandergesetzt hatten, sind beispielsweise Frank O. Gehry, Ben van Berkel, Rem Koolhaas, Peter Eisenmann, Norman Foster oder Zaha Hadid.

Im Allgemeinen wurden vorerst insbesondere die neuen Visualisierungstechniken in Form von Erweiterungsaufsätzen in CAAD-Programmen adaptiert, da es zunächst tendenziell um die verbesserte Darstellung während der Modellierungsphase von Objekten und danach um die Präsentation von fertigen Projekten ging. So war zum Beispiel die Entwicklungsfirma der Animationssoftware „3d Studio Max" von Anfang an eng mit dem CAAD-Softwarehersteller „Autodesk" in Verbindung, da die „Yost Group" für „Autodesk" einzelne Grafikanwendungen entwickelte.[55] Durch die vermehrte Verwendung dieser Werkzeuge begann langsam eine neue Ära in der Architekturvisualisierung. Da in den künstlichen Computerbildern „Renderings" inzwischen ganze Gebäudekomplexe und einzelne Räume mitsamt ihren Oberflächenmaterialien dargestellt werden konnten, wurden diese Visualisierungen auch zur Überprüfung von Architekturkonzepten verwendet. Weiters konnten Architekten mit diesen neuen Methoden über eine je nach Konzept mehr oder weniger realitätsnahe Projektdarstellung,

in der auch Umgebungsobjekte wie Menschen, Bäume, Autos enthalten sein konnten, ihre Idee somit dem Laien kommunizieren. Die ständig steigende Qualität der „Renderings" führte zu einem neuen Darstellungsstandard in der Architekturpräsentation.[56]

Neben ihrer Funktion zur Visualisierung nutzten Architekten die Animationssoftware in einem weiteren Schritt als Modellierungsmöglichkeiten zur Konstruktion von „Blobs". Bei der Modellierung stellt sich nach wie vor oft heraus, dass diese Programme zwar gut für die Erstellung von komplizierten Formen geeignet sind, es ihnen aber häufig an Genauigkeit mangelt und diese Formen nicht einfach bemaßt und direkt in ernstzunehmende Pläne umgesetzt werden können. Hinzu kommt, dass bei der Übergabe der Geometriedaten an Fertigungsmaschinen wie CNC-Fräsen zwar unzählige Austauschformate, auch „Export-Formate" genannt, unterstützt werden, aufgrund von Ungenauigkeiten oder Konvertierungsfehlern bei der Umwandlung zwischen den Formaten aber fehlerhafte Objektgeometrien entstehen. Nicht zuletzt aus diesen Gründen wird beim Konstruieren von komplizierten Geometrien häufig auf CAD-Softwareprogramme wie „CATIA" zurückgegriffen, die, da sie aus der Fertigungsbranche kommen, mehr Nähe zur praktischen Herstellung von Formen aufweisen.

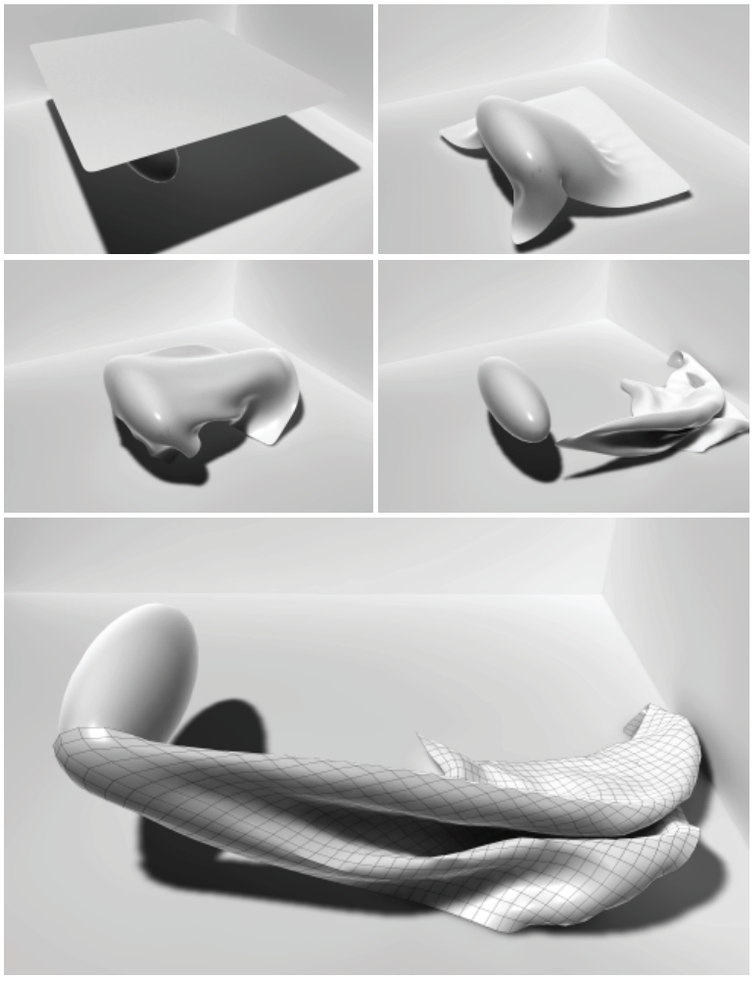


Abbildung 26: So kann die Formfindung mittels der Animationssoftware und ihren unterschiedlichsten Plugins durch Experimentieren mit physikalischen Eigenschaften stattfinden. In diesem Beispiel wurde versucht, eine Form zu generieren, die einem fallendem Tuch unter Windeinfluss enspricht.

Trotzdem zeigen aktuelle Tendenzen, dass Animationsprogramme wie Autodesk 3d Studio, Autodesk Maya, Cinema 4d oder Rhinoceros immer stärker in den Entwurfsprozess eingebunden werden. Wie auch Greg Lynn beispielhaft zeigt, wird in aktuellen Entwürfen der Computer oft als virtuelles Experimentierlabor verstanden und vermehrt als Formenfindungstool genützt. Hierbei werden alle möglichen Effekte und Erweiterungen ausprobiert und miteinander verknüpft. Dazu sei bemerkt, dass viele dieser „Plugins" wie beispielsweise Wassersimulationsprogramme in der Regel für die Filmbranche entwickelt wurden, wobei es vorwiegend um das Erzeugen des optischen Eindrucks geht und es keine Rolle spielt, ob sich diese Flüssigkeit nun wirklich wie in etwa Wasser verhält.

Zum Beispiel ist die „Fluid Dynamics Software" (auch „Flüssigkeiten - Simulations – Programm) von der Firma „Next Limit Technologies" in zwei Varianten erhältlich. Das Produkt „Real Flow" wurde für die Filmbranche entwickelt und führt in relativ wenigen Schritten zu einem optisch scheinbar richtigen Verhalten von Flüssigkeiten. Für wissenschaftliche Forschungsarbeiten bietet Next Limit Technologies ein professionelleres Produkt mit dem Namen „XFlow" an. Dieses Programm wird unter anderem im Flugzeug- bzw. Autobau für Aerodynamikstudien, im Schiffsbau für Wasserströmungsuntersuchungen, in der Medizin im Bereich des Bioengineering zur virtuellen Messung von Blutströmen in Gefäßen oder auch in der Architektur und Bauforschung für Windstrommessungsstudien von unterschiedlichen Gebäudeanordnungen oder auch zur Simulation von Überschwemmungssituationen verwendet.[57]

So gibt es zurzeit, wie bereits erwähnt, eine große Auswahl an Erweiterungen, anhand derer Animationsprogramme zu virtuellen Experimentierlaboren werden und den ArchitektInnen zur Analyse von bestimmten Objektkonstellationen sowie zur Konstruktion, Generierung oder Inspiration von neuen Formen dienen. Abschließend sei bemerkt, dass Lynn die eigentliche Revolution nicht in der bewegten Form an sich sieht, sondern in der Verwendung der Infinitesimalrechnung (engl. Calculus), die eine Interpolation zwischen Punkten im Raum in unendlich kleinen Schritten beschreibbar macht und somit fließende Formen und Bewegungen erst ermöglicht. So bemerkt Lynn:

„I initially, in Animate FORM, focused my thinking on the revolution in motion and only later realized that real revolution was in the use of a 300-year-old invention, calculus." Aus diesem Grund bezeichnet Lynn diese neuen digitalen Formen als „Calculus Forms"[58].

```
println("Haven't found yet");
    }
  } // Ende ARRAY ERZEUGEN FOR SCHLEIFE
return BRIDGE_POINTS_ARRAY;
}

//XZ-Abstand zwischen 2 Punkten berechnen
XYZ_Dist_2_Points_Calc(p1,p2)
{
  var DistPoint_X,DistPoint_Y,DistPoint_Z;

  DistPoint_X=p2.x-p1.x;
  DistPoint_Y=p2.y-p1.y;
  DistPoint_Z=p2.z-p1.z;
  Dist_current_Points=sqrt((DistPoint_X*DistPoint_X)+(DistPoint_Y*DistPoint_Y)+(DistPoi

  return Dist_current_Points;
}


//MAKE_SPHERE
MAKE_SPHERE(Scale,Name,Position,Parent)
{
  var Sphere;

  if(!doc->FindObject(Name))
  {
      Sphere = new(SphereObject);
     Sphere->SetName(Name);
     Sphere#PRIM_SPHERE_RAD=Scale;
     Sphere->SetPosition(Position);

     doc->InsertObject(Sphere,Parent, null);
  }
  else
  {
    Sphere = doc->FindObject(Name);
     Sphere->SetPosition(Position);
    Sphere#PRIM_SPHERE_RAD=Scale;
  }
  Sphere->Message(MSG_UPDATE);
```

Abbildung 27: „Coffee" Script in Cinema 4d.

```
float $xpos = 1;
float $ypos1 = 1;
int $zpos = 0;
int $Wuerfel = 50;
int $Reihen = 6;            //Anzahl der Reihen nebeneinander
int $FARBE =0;
int $Winkel=0;              //Sinus Startwinkel
int $Schichten = 3;         //Anzahl der übereinander liegenden Schichten
$Distanz= 2;                  //Abstand zwischen den Schichten
int $Abstand= 5;
int $sinus=2;               //Amplitude der Sinuskurve

//-------------Schichten Übereinander-----------------------------------------------------
for ($c; $c <=$Schichten; $c++)
{
 $zpos = 0;
 $Abstand= $Abstand + $Distanz;
 $a=1;
//---------Reihe in die Breite-------------------------------------------------------------
for ($a; $a <=$Reihen; $a++)
{
  $zpos=$zpos+3;
  $i=1;
  $xpos = 1;
  $Winkel=0;//Winkel ein/aus
//-------------------Wuerfel erzeugen-----------------------------------------------------
for ($i; $i <= $Wuerfel; $i++)
{
 print ("Z-Position: " + $zpos + " \n");
 $xpos =$xpos +1;
 $ypos1 =(sin(deg_to_rad($Winkel)))*$sinus + $Abstand;
 //$zpos =$zpos+1;
  $Winkel=$Winkel+15;
print ($i+"Y-Position: " + $ypos1 + " \n");
polyCube -w 1 -h 1 -d 1 -sx 1 -sy 1 -sz 1 -ax 0 1 0 -tx 1 -ch 1;
move -r $xpos $ypos1 $zpos;

viewFit -all; //Alle vorhandenen Objekte im Bild anzeigen;
refresh; //Neuzeichnen
}
```

Abbildung 28: „MEL" Script editiert in Notepad ++.

Parametergesteuertes Entwerfen mittels Scripting- oder Mashup-Systemen

In aktuellen Architekturdebatten wird rund um das stark präsente Thema des parametrischen Entwerfens neben der Konstruktion von intelligenten Formen durch den Einsatz von generativen Algorithmen als neues Thema im Entwurfsprozess auch der Umgang mit Komplexität betont.[59] Da Entwürfe, die in diesem Kontext diskutiert werden, in der Regel in Animationssoftware mittels „Scripting" und intelligenten "Mashup-" bzw. „Node-Systemen" entworfen wurden und auch in der vorliegenden Arbeit der „Brückengenerierungsprozess" mittels solcher Systeme realisiert wurde, sollen diese Methoden hier kurz näher gebracht werden.

Zur Erstellung von parametergesteuerten Entwürfen oder zur Programmierung von generischen Algorithmen mittels Animationssoftware sind Kenntnisse im sogenannten „Scripten" notwendig. Mit Hilfe von Scripten kann auf vordefinierte Softwarefunktionen wie Kopieren, Verschieben, Verdrehen, Skalieren, Punkte auswählen, Extrudieren etc. zugegriffen werden, wobei diese Befehle in Befehlsketten organisiert werden können. Dadurch kann die Objektbearbeitung automatisiert und von verschieden Faktoren abhängig gemacht werden.[60] Die Werkzeugbearbeitungsbefehle können ähnlich wie in Programmiersprachen beispielsweise in sogenannten „For" -oder „Do-While" - Schleifen wiederholt werden, bis eine bestimmte Abbruchbedingung erfüllt oder - in anderen Worten gesagt - auf „True" gesetzt wurde. Weiters können auch Objekteigenschaften wie Radius, Position, Verdrehung, Material etc. abgefragt werden und beispielsweise durch „If-Then-Else Bedingungen" mit anderen Parametern verknüpft oder über Formeln verändert werden. Ein Script wird in einem Texteditor geschrieben, wobei der Text aus Befehlen besteht und gewissen Grammatikregeln, der sogenannten „Syntax", unterliegt. Damit ähneln Scriptsprachen den sogenannten modernen „höheren Programmiersprachen" wie „C++" oder „Java". Allerdings bauen Scriptsprachen auf den vorher genannten höheren Programmiersprachen auf, sind bei weitem nicht so umfangreich, nur in starken Grenzen je nach Animationssoftware verwendbar und viel langsamer als die anderen Hochsprachen. Aus Übersichtsgründen verfügen die meisten Texteditoren über eine Text- „Highlighting"- Funktion, womit einzelne Befehle, Zeichen etc. je nach Bedeutung Farben zugewiesen bekommen. Gängige Scriptsprachen sind zum Beispiel „Coffee" (Maxon Cinema 4d), "Form*Z Script" (AutoDesSys Form*Z), „Max Script" (Autodesk 3d Studio Max), „MEL" (Autodesk Maya) oder „RhinoScript" (McNeel Rhinoceros).

An dieser Stelle sei erwähnt, dass auch andere CAD-Programme wie etwa „AutoCAD" mit „LISP" ebenfalls Scriptsprachen unterstützen.[61] AutoCAD war das erste am Personal Computer verwendete CAD-Programm, das das Aufzeichnen von „Command-Line-Befehlen" erlaubte und damit eine sequenzielle Organisation von Zeichenoperationen ermöglichte.[62] Viele Softwarehersteller bieten neben ihren internen Scriptsprachen Plugins für weitere Sprachen an, wobei „Python" eine der beliebtesten Scriptsprachen ist. Mit Python kann zum Beispiel in „Blender", „Cinema 4d", „Autodesk Maya" und „Rhinoceros" „gescripted" werden.[63] Scripten werden auch in Form von Plugins als Befehlserweiterungen von Programmen zur Lösung von gängigen Problemstellungen im Internet in zahlreichen Foren privat oder kommerziell angeboten.

Als Alternativen zum „Scripting" bieten einige Hersteller speziell für DesignerInnen und ArchitektInnen auch „Node-Systeme" bzw. *„Mashup-Tools"*[64] an, die für das Realisieren von parametergesteuerten Formen praktisch keine Programmiergrundkenntnisse voraussetzen. In der Informatik werden diese - in dieser Arbeit als „Node-Systeme" bezeichneten - Werkzeuge als „Mashup-Tools" bezeichnet, die das Mixen verschiedener Daten ermöglichen sollen. Diese Systeme stellen eine Menge von unterschiedlichsten Funktionsboxen bereit, die als „Black Boxes" jeweils Ein- und Ausgänge besitzen und über diese miteinander vernetzt werden können. So können auch Objekte als Eigenschaften/Funktionsbox dargestellt werden, wobei in den einfachsten Fällen etwa die X-Position eines Würfelobjekts mit dem Radius eines Kugelobjekts verbunden werden kann (siehe Abb. 29).

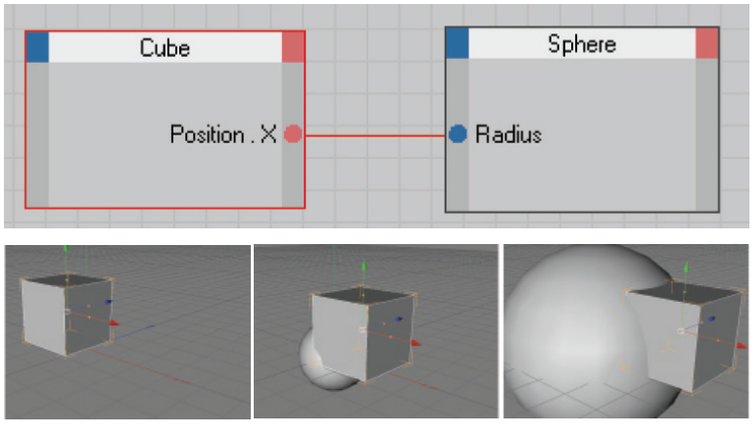


Abbildung 29-30: Die obere Abbildung zeigt „XPresso"- Schaltung in Cinema 4d. Hier wurde die X-Position des Würfels mit dem Radius der Kugel verbunden. In der Abbildung darunter erkennt man, dass die Kugel unsichtbar wird, wenn der Würfel sich im negativen Koordinatenbereich bewegt, da es keinen negativen Radius gibt.

In ihrer grafischen Erscheinung orientieren sich die Node-Systeme an der Optik von elektrischen Schaltungen. Da solche „Node-Schaltungen" bei komplexeren individuellen Aufgaben schnell unübersichtlich werden können, kann ein solcher „Node" auch ein ganzes Script enthalten, womit eine Kombination zwischen Script- und Node-Systemen möglich wird. Weiters werden Bedienoberflächen unterstützt, über die Eigenschaften und Werte mittels Schiebereglern, Textfeldern oder ähnlichen Möglichkeiten eingegeben werden können. Über spezielle Knoten lassen sich auch externe Daten wie Textdateien oder Audiosignale einlesen. Beliebte Node-Systeme sind zum Beispiel „Grasshopper" in Rhino oder „XPresso" in Cinema 4d.

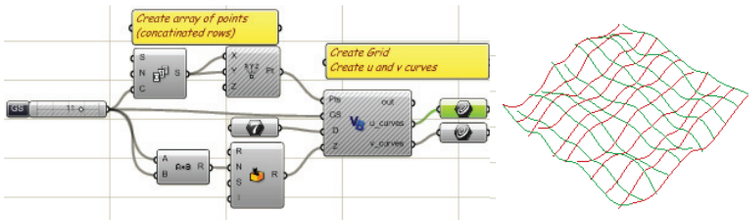


Abbildung 31: „Grasshopper" – Schaltungen in Rhinoceros.

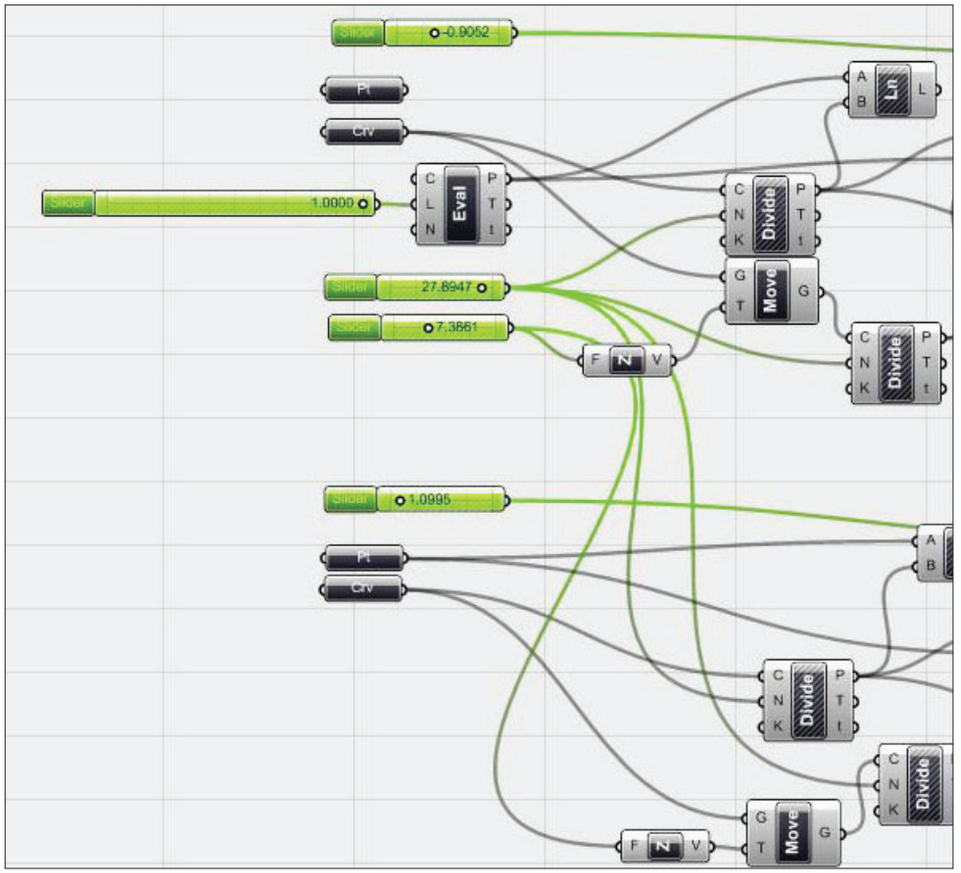


Abbildung 32: „Grasshopper" – Schaltungen in Rhinoceros. Achtung: kein Objekt der restlichen Abbildung wurde mit dieser Schaltung generiert.

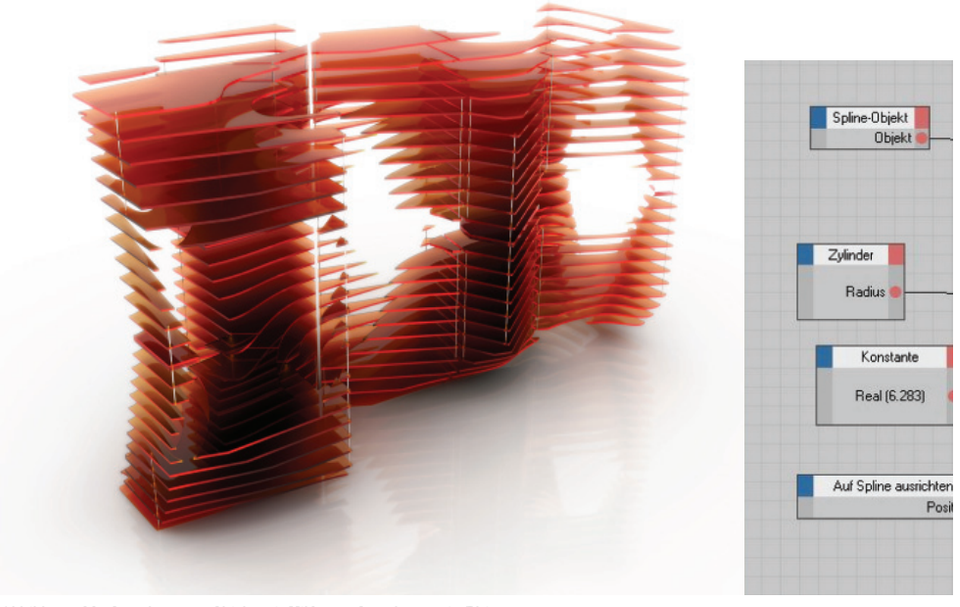
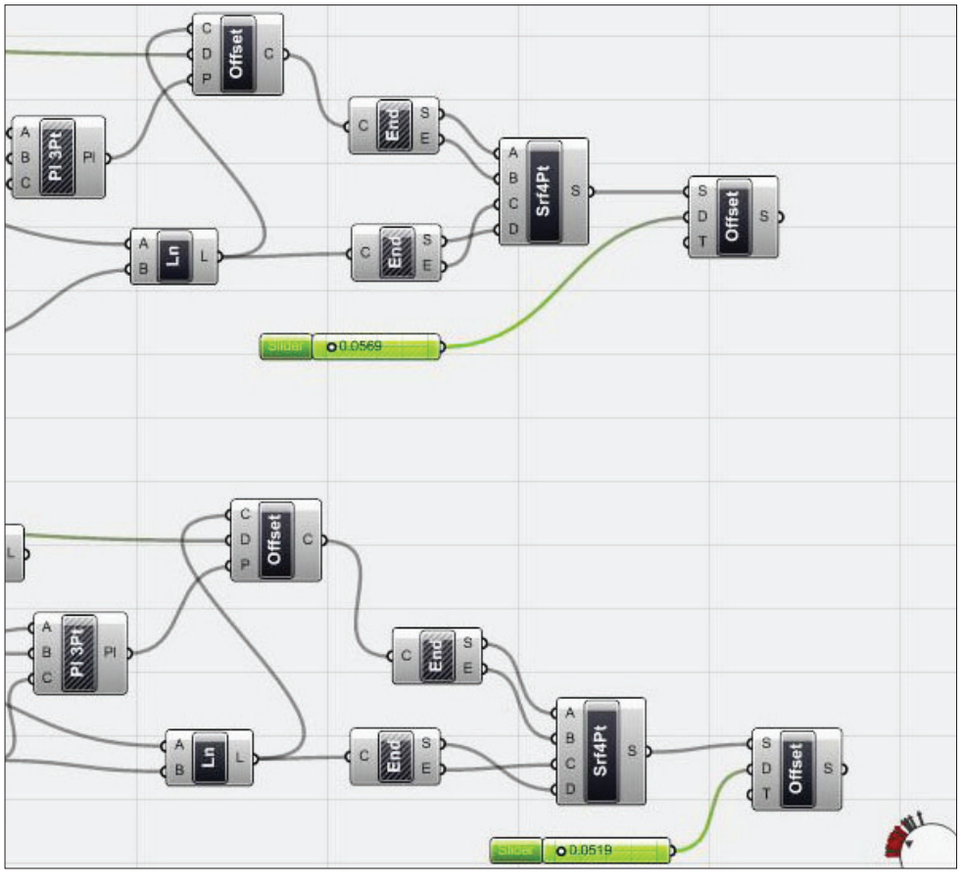
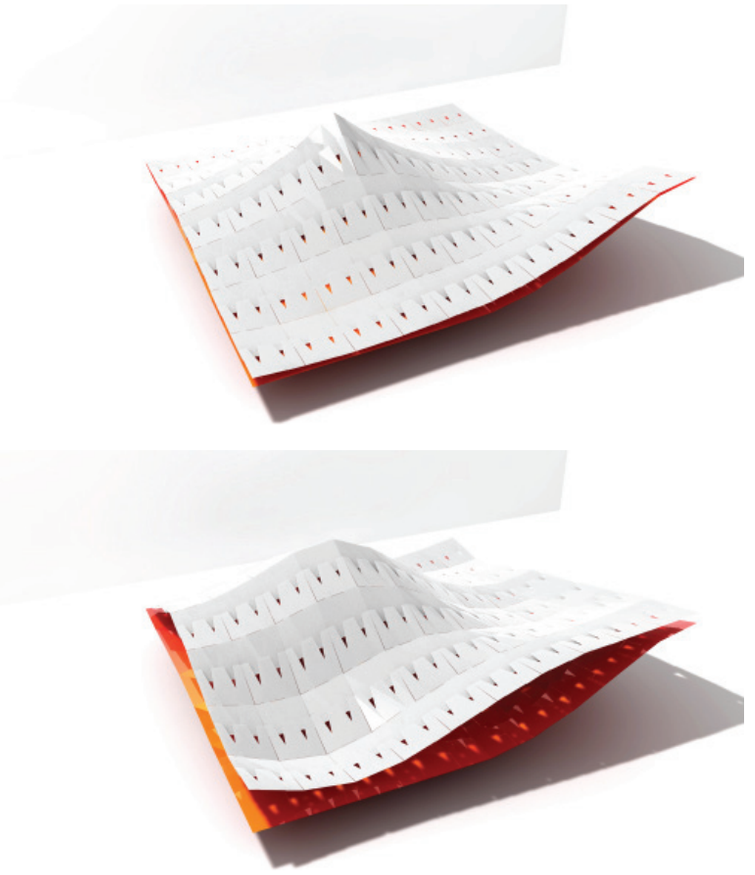


Abbildung 33: Grasshopper - Objekt mit Hilfe von Grasshopper in Rhinoceros.





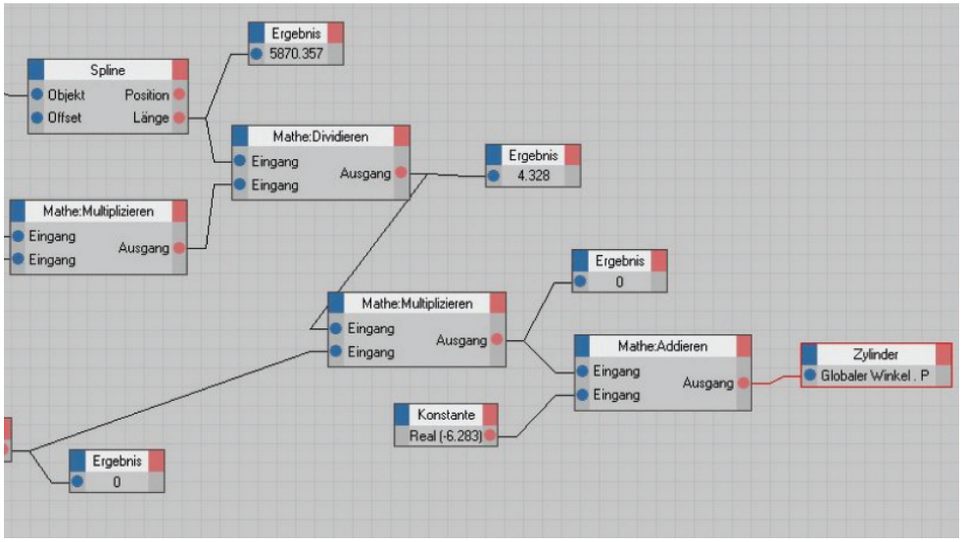


Abbildung 34: "XPresso"-Schaltung in Cinema 4d.

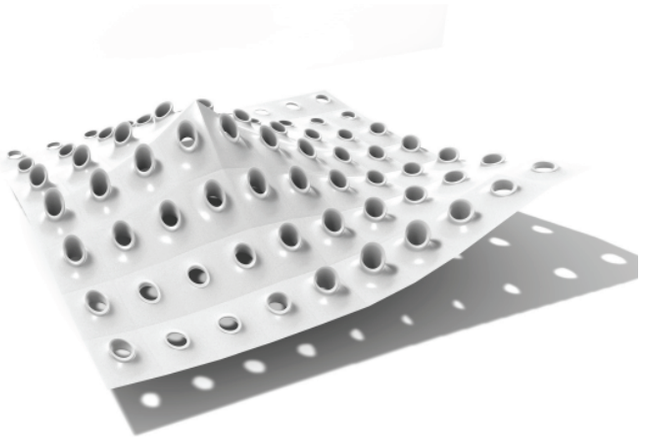


Abbildung 35: Diese drei Objekte sind mittels eines sogenannten Paneling-Werkzeugs generiert worden. Dieses Werkzeug wurde mit Hilfe von „Coffee" und „XPresso" in Cinema 4d selbst entworfen. Mit Hilfe dieses Werkzeuges können Objekte (Panele) auf beliebigen Obflächen postitioniert und ausgerichtet werden, dabei kann das Panel zusätzlich zwischen zwei Oberflächen verzerrt werden. In Rhinoceros ist das sogennate Panneling Tool bereits integriert. Die zwei oberesten Bilder zeigen Plattenelemente, die zwischen zwei Ebenen positioniert werden. Das untere Bild zeigt ein dreidimensionales Objekt, das ebenfalls vervielfacht und an die zwei Ebenen angepasst wird.

Zusammenfassend kann festgehalten werden, dass bei der Entwicklung von CAD-Programmen versucht wird, eine möglichst geschlossene Kette vom Entwurfsstadium bis zur Ausführungs- und Fertigungsphase zu erreichen. Immer mehr Programme versuchen, ihr Funktionsspektrum zu erweitern und streben eine „All in One Lösung" in Form einer flexiblen Plattform an. Dies funktioniert in der Regel durch Kooperation mit anderen Herstellern, wobei versucht wird, die unterschiedlichsten Produkte in Form von Plugins in das eigene Programm zu integrieren. Werden CAD-Programme als Werkzeuge verstanden, die einerseits zur bewussten Konstruktion und andererseits auch im Entwurfsprozess zur Formfindung eingesetzt werden, dann ist in vielen Fällen nicht mehr ganz klar, was nun eine CAD-Software ist und was nicht. So können beispielsweise etwa mit Open Source Software wie „Processing"[65], einer Java basierten objektorientierten Scriptsprache, welche für den Grafik- und Simulationsbereich entwickelt wurde, oder auch mit Multi-Agentenprogrammiersprachen wie „NetLogo"[66] generative Prozesse simuliert werden. Die Ergebnisse können dabei zum Beispiel in Form von „Dxf-Dateien" ausgegeben werden. So versucht man nach Negropontes Erfolgsrezept „Moving bits instead of atoms"[67] analoge Prozesse weitest möglichst zu digitalisieren, wobei über das Schlagwort „File to Factory" auch der Fertigungsprozess direkt eingebunden und dadurch die Virtualität mit der physischen Realität verknüpft werden soll.

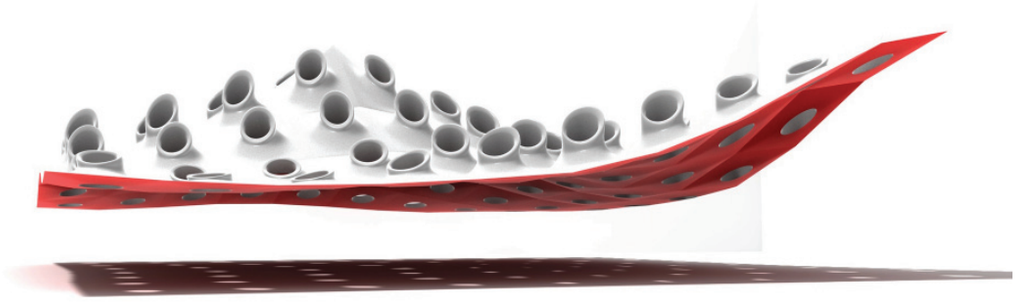


Abbildung 36:
Mittels "Mashup-Tool" und "Scripting" generierte Form.

44 David E. Weisberg, "The Engeneering Design Revolution", e-Book 2008, http://www. cadhistory.net (28.02.2010), Abschnitt 2: A Brief Overview of the History of CAD, S.8.

45 http://dict.leo.org, Online Wörterbuch (Stand 08.03.2010).

46 Vgl. Oliver Fritz, "CAM of Freeforms in Architecture", Hg. Institut für Raumplanung der Hochschule Liechtenstein, Michael Imhof Verlag GmbH & Co. KG, S.29–30.

47 Vgl. dieses gesamte Kapitel "Der Einfluss der Fertigungsindustrie auf die Entwicklung von CAD-Software" mit David E. Weisberg, The Engeneering Design Revolution, e-Book 2008.

48 Vgl. Marco Hemmerling, Anke Tiggemann, "Digitales Entwerfen", 2010 Wilhelm Fink GmbH & Co. Verlags-KG, S.15.

49 Vgl. ebd., S.12–15. a

50 Vgl. ebd., S.14.

51 Eine Liste von einigen CAD-Softwareprogrammen: http://de.wikipedia.org/wiki/Liste_von_CAD-Programmen http://www.experiencefestival.com/a/List_of_CAD_companies_-_Commercial_CAD_Software/id/1590580 http://crunkish.com/top-ten-cad-software

52 Vgl. Marco Hemmerling, Anke Tiggemann, "Digitales Entwerfen", a. a. O.,, S.15.

53 Vgl. http://www.dma.ufg.ac.at/app/link/app (Stand 15.03.2010).

54 Vgl. ebd., ab S.16.

55 Vgl. http://www.maxunderground.com/the_history_of_3d_studio, (Stand 13.03.2010).

56 Vgl. Marco Hemmerling, Anke Tiggemann, "Digitales Entwerfen", a. a. O., S.87-89.

57 Vgl. http://www.nextlimit.com/techno_fluid.php (Stand 15.03.2010).

58 Greg Lynn, Interview mit Ingeborg Rocker, Architectural Design, Ausgabe July/August 2006, S.90.

59 Vgl. GAM06, Grazer Architektur Magazin, Herausgeber Architektur Fakultät Technische Universität Graz 2010.

60 Vgl. Marco Hemmerling, Anke Tiggemann, "Digitales Entwerfen", a. a. O., S.200.

61 http://usa.autodesk.com/adsk/servlet/item?siteID=123112&id=2309147, (16.03.2010).

62 Vgl. Malcolm McCullough, Article: 20 Years of Scripted Space, Architectural Design, July/August 2006, EditorHelen Castle, Wiley-Academy, S.13.

63 Vgl. http://www.py4d.com, (16.03.2010); http://usa.autodesk.com/adsk/servlet/index?siteID=123112&id=9469002, (16.03.2010), http://blog.rhino3d.com/2010/03/python-scripting-in-rhino-os-x.html, (16.03.2010), http://wiki.blender.org/index.php/Doc:Manual/Extensions/Python, (16.03.2010), http://blog.rhino3d.com/2010/03/python-scripting-in-rhino-os-x.html; (16.03.2010.)

64 http://technologydriven.wordpress.com/2007/02/09/yahoo-pipes-visual-mashup-tool (Stand 01.04.2010).

65 Siehe Beispiele auf http://processing.org, (Stand 17.03.2010).

66 Siehe Beispiele auf http://ccl.northwestern.edu/netlogo, (Stand 17.03.2010).

67 Vgl. Nicholas Negroponte, Being digital, Vintage Books a division of Random House, 1995 USA, S.11-20.

Parametrisches Entwerfen – Fragmente der aktuellen Debatte

Parametrismus als der neue Internationale Stil

Im November 2009 schrieb Patrik Schumacher von „Zaha Hadid Architects" in der Zeitschrift „Arch+" in seinem Beitrag *„PARAMETRISMUS – Der neue International Style"*[68] der aktuellen Avantgarde-Architektur die Fähigkeit zur Bildung eines neuen eigenständigen Stils zu. Nach einer langen Inkubationszeit könne nun nach der „Moderne" erstmals endlich wieder ein neuer internationaler Stil ausgerufen werden. Dabei betrachtete Schumacher Entwicklungen wie etwa die „Postmoderne" und den „Dekonstruktivismus" als unausgereifte Stile, die als bloße Vorboten des nun vollkommenen „Parametrismus" zu verstehen seien. Doch hätten diese Wegbereiter des neuen Stils wesentlich zur Forschung und Entwicklung der neuen Techniken beigetragen. So schrieb Schumacher dazu u.a. folgendes:

„Die Verwendung von Animation, Simulation und Formfindungstechniken ebenso wie das parametrische Modellieren und Scripten hat eine neue Architekturrichtung mit radikal neuen Zielen und Werten inspiriert."[69]

In den gegenwärtig zahlreich zur Verfügung stehenden digitalen Methoden sieht Schumacher das Pendant zur heutigen heterogenen pluralistischen Gesellschaft, welche durch eine zunehmende Differenzierung eine immer komplexere Dynamik beschreibt. Der Begriff der „kontinuierlichen Differenzierung", welcher in den 1990er Jahren von Greg Lynn und Jeff Kipnis geprägt wurde, markiert den Zusammenhang zwischen der vermehrt individualisierten Massengesellschaft und dem Umgang mit der steigenden Komplexität mittels digitaler Techniken und soll als Schlüsselbegriff der aktuellen Entwicklung verstanden werden. Bereits im Jahr 2002 betonte Schumacher in dem Text *„The Autopoesis of Architecture"*[70] die Möglichkeit der Schaffung einer neuen automatisierten oder sich selbstgenerierenden Poesie durch digitale Medien, wobei er in Anlehnung an Niklas Luhmanns (1927-1998) allgemeiner Theorie in der fortschreitenden Zerstreuung der Wissenschaften untereinander eine damit verbundene Autonomisierung ausmachte, in der er eine geniale evolutionäre Strategie sah. So schrieb Luhmann: *„Im Kontext dieser Evolutionstheorie und bei entsprechender Anreicherung der Begriffe Selektion und Anpassung gelangt man zu einer Neueinschätzung der (zunächst technischen) Erfindungen interaktionsfreier gesellschaftlicher Kommunikation; und ferner zur Neueinschätzung von Formen gesellschaftlicher Komplexität (zum Beispiel Systemdifferenzierung, die nicht mehr durch Interaktion gefährdet werden können)."*[71]

Kurz: Aufgrund der vielen unabhängigen Einzelsysteme besteht eine geringe Gefahr eines Totalversagens des Gesamtsystems. Schumacher pointiert den Unterschied des Reaktionsverhaltens der seines Erachtens nach erfolgreichen Chaossysteme gegenüber jenen der Kausalität in dem Absatz *„Irritation versus Determination"* wie folgt: *„It's the difference between kicking a dog versus kicking a ball."*[72]

Zu den aktuellen Schwerpunkten und allgemeinen Tendenzen in der Gegenwartsarchitektur bemerkte Schumacher im Jahr 2002 zudem folgendes:

„Current experimental work focuses on issues of organizational complexity (layering, interpenetration of domains), the production of diversity (iteration vs. repetition), the spatial recognition of fuzzy social logics (smooth vs. striated space), ways of coping with uncertainty (virtuality vs. actuality), and engagement with new production technologies (file to factory), etc."[73]

Im Umgang mit komplexen Systemen - unterstützt durch neue Techniken wie Scripten und parametrische Modellierung - sieht Schumacher das Potential, Innovationen systematisch mit wissenschaftlichen Methoden anreichern zu können. In diesem Sinn beschrieb Schumacher „Stile" als Forschungsprogramme, die durch ihre unterschiedlichen Zielvorstellungen und Werte zu unterschiedlichen Praktiken und damit verschiedenen architektonischen Innovationen führen. Bei einem Stilwechsel sei die Aufgabe der Avantgarde-Architektur das Produzieren von Manifesten und paradigmatischen Illustrationen, wodurch das Potential der neuen Strömung demonstriert werden soll, um somit den epistemischen Kern des neuen Stils zu festigen. Hierbei soll das Entwerfen von funktionierenden Gebäuden keine Rolle spielen, denn über die ersten Werke sollen eben vorerst neue Werte und Forschungsziele etabliert werden. In seiner Euphorie stellte Schumacher anhand von städtebaulichen Beispielen in Ahnlehnung an Frei Otto und Le Corbusier (1887-1965) fünf Punkte zur *„Definition einer Heuristik und Zielsetzung"*[74] des Parametrismus auf. Vorab werden in diesem Text die Verwendung von strengen geometrischen Formen wie Rechtecken, Dreiecken und Kreisen etc., die einfache Wiederholung sowie die Folge von unverbundenen Elementen als Tabus definiert. Im Wesentlichen lassen sich die fünf Punkte auf den mehrfach erwähnten vielschichtigen Umgang mit „Komplexität" in allen Ebenen reduzieren, woraus laut Schumacher eine Ästhetik - bestimmt durch elegante fluide Formen - resultiert.

Abbildung 37: Städtebauliches Projekt von Zaha Hadid Architects.

Als Beispiel vergleicht Schumacher diese neuen Formen mehrmals mit der Dynamik von Schwärmen, denen die Grundprinzipien der neuen Ästhetik als immanent scheinen. So sollen beispielsweise Gebäude oder Fassadenelemente wie Fische in einem Schwarm organisiert werden. Meines Erachtens wird in Schumachers Argumentation deutlich, dass die Selbstverständlichkeit der Existenz und Begründung des „Parametrismus" vor allem über die mehrfache Betonung einer offensichtlichen Verwandtschaft zu natürlichen Systemen zu erreichen versucht wird. Generell lassen sich die mE nur wenig neue Ansätze erkennbaren Ausführungen Schumachers auf die Tradition der Kybernetiker in den 1950er Jahren zurückführen, wobei in diesem Zusammenhang insbesondere die Arbeiten rund um Max Bense oder Joseph C. R. Licklider hervorzuheben sind.

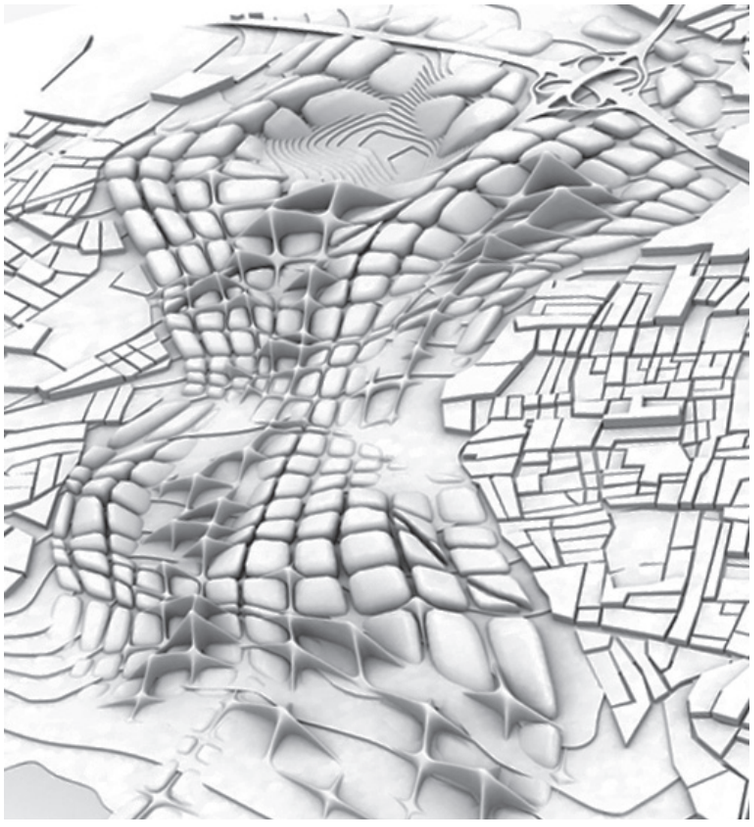


Abbildung 38: Städtebauliches Projekt von Zaha Hadid Architects.

Vitruv und der Parametrismus

Im Gegensatz zu Schumacher versteht Bernard Cache den Parametrismus nicht als Neuerscheinung am Himmel der Architekturgeschichte, sondern begreift das parameterabhängige Modellieren und Beschreiben von Architektur als einen schon seit jeher bekannten integralen Bestandteil des architektonischen Entwerfens. Etwas sarkastisch bemerkt Cache in seinem Artikel „After Parametrics"[75] , dass er den von einigen Architekturschulen bereits als überholt bezeichneten „Parametrismus" mit Freude zu Grabe tragen helfen würde, aber er doch sehr verwundert darüber sei, welche Neuerungen denn nun folgen sollten. Versteht man das parametrische Entwerfen als eine Methode zur Formfindung, in der unterschiedlichste Eigenschaften aus dem Entwurfskontext abstrahiert und in Form von Geometrie bestimmenden Parametern untereinander verknüpft werden, dann finden sich laut den Ausführungen von Cache bereits in der ersten Architekturbeschreibung um 30 v. Chr. Hinweise auf diese Praxis, woraus sich eine mindestens etwa 2040 Jahre lange Tradition für diese Technik ergibt. So stellt sich Cache in seinem Text gegen Anfang folgende Fragen:

„Wann hat jemand zuerst begonnen, die verschiedenen Bestandteile eines Gebäudes durch Zahlenverhältnisse zu bestimmen? Wann wurden Architektur und Mechanik miteinander verknüpft? Und seit wann haben wir Maschinen verwendet, um diese Verhältnisse mittels komplexer Funktionen zu berechnen?"[76]

Dabei demonstriert Cache beispielhaft anhand der von Marcus Vitruvius Pollio um 30 v. Chr. verfassten „De Achitectura libri decem", wie der Parametrismus in Rom bereits zu Cäsars Zeiten bei der Erstellung von Gebäuden und Kriegsmaschinen exerziert wurde. So stehen bei der Auswahl des geeigneten Entwurfsortes und der Ausrichtung des Gebäudes bis zu den Entscheidungshilfen für die richtige Material- oder Säulenwahl eine Vielzahl von Methoden und Regeln zur Verfügung, die für das Entwerfen einer einwandfreien Architektur zur Anwendung kommen sollten. Zur Herleitung von bestimmten Zahlenverhältnissen werden als Ausgangspunkt einzelne „Module" definiert. Als Beispiel schreibt Cache:
„Im Tempel wird dieses Modul der Durchmesser oder der untere Umfang der Säule sein, abhängig davon, ob man die ionische, korinthische oder dorische Säulenordnung verwendet. D.h., die Größe dieses Moduls ist im Wesentlichen das Ergebnis der Teilung der Fassadenbreite, die sich wiederum aus der Ordnung, der Anzahl der Teilungen und des Rhythmus, die zur Anwendung kommen, generiert."[77]

Eine weitere interessante Beobachtung macht Cache bezüglich der offensichtlichen Distanz zur kulturellen bzw. funktionalen Bedeutung von Gebäuden, die Vitruv in seinen Ausführungen einnimmt. So könne man bei Vitruv als jemanden, der zwei Bücher über den Tempelbau geschrieben hat, praktisch nichts über Gebäudefunktionen oder Abläufe wie Opferrituale, Prozessionen etc. erfahren. Vitruv hätte obendrein mit spöttischen Anmerkungen über Religion nicht gegeizt, wie Cache mit folgendem Beispiel darlegt: *„Man sollte daher die Tempel der heilenden Götter an gesunden Orten errichten, empfiehlt der Architekt ironisch. Wenn sich dann die Gesundheit der Kranken verbesserte, sobald sie diese Orte aufsuchten, könnte man die Heilung immer auf Aesculap und seine Brüder zurückführen."*[78]

Mit diesen Erläuterungen relativiert Cache mE Schumachers Ansage, den Parametrismus als neuen Stil etablieren zu wollen, indem er darlegt, dass eine parametrische Konzeptionierung bereits seit Jahrtausenden integraler Bestandteil des architektonischen Entwerfens sei.

Das neue Ornament - Wendepunkt in der Architektur

„Parametrismus" als Stil oder Nichtstil - laut Jörg H. Gleiter indiziert die aktuell aufflammende Diskussion über das „neue Ornament" einen Wendepunkt in der Architektur. So behauptet Gleiter in Anlehnung an Gérard Raulet und Burghart Schmidt, dass das Ornament keine autonome gestalterische Praxis sei, sondern über dessen „diskursiv-architektonische Funktion" in Kontext zum dynamischen „allgemein-kulturellen Kräftefeld" steht. Nach Gleiter *„ist das Wiederaufleben des Ornaments in der zeitgenössischen Architektur heute, am Übergang vom mechanischen zum digitalen Zeitalter, weniger erstaunlich als dies oft suggeriert wird; es ist Zeichen eines tief greifenden Wandels der Architektur"*[79].

Dementsprechend unterliegt das Ornament dem Wandel und Schwerpunkten der jeweiligen Zeit und ist hingegen der weit verbreiteten Ansicht im letzten Jahrhundert nicht überwunden worden, sondern war stets in unterschiedlichsten Formen äußerst präsent. Als Beispiele und zentrale Themen nennt Gleiter das *„strukturale Ornament"* in Verbindung mit den Debatten des „Strukturalismus" Mitte des Jahrhunderts sowie das *„ironisch-allegorische Ornament"*, hervorgerufen durch die Zeichenhaftigkeit des Poststrukturalismus in der Postmoderne um die 1970er Jahre und das *„performativ-kritische Ornament"* gegen Ende des 20. Jahrhunderts. So erlebte etwa das *„strukturale Ornament"* zu Beginn der Entwicklung des Computers ihre Blütezeit, wobei man hiezu Arbeiten wie die Computergrafiken Frieder Nakes oder Georg Nees' (siehe Abb. 12-15) oder Konstruktionen wie die geodätischen Dome von Buckminster Fuller oder das Waisenhaus in Amsterdam von Aldo van Eyck diesem Typus zuordnen kann. Was ist nun als Ornament zu verstehen und worin liegen die Unterschiede der vorhergehenden Ornamente zum „neuen Ornament"? Dazu führt Gleiter als grundlegende Eigenschaft den „doppelten Verweis" eines Ornaments ein, der einerseits auf anthropologische Bedeutungen und anderseits auf die konstruktive Gemachtheit – die Umstände des Werdens - eines Objekts hinweist und damit einen Zusammenhang zur jeweiligen Zeit herstellt. Je nach Epoche schwanken die jeweiligen Betonungen. So liegt der Schwerpunkt des „klassischen" Ornaments auf der Betonung der anthropologischen Komponente, indem das Artefakt über das Ornament auf die zusätzliche Präsenz von etwas Abwesendem deutet. *„Nach Sempers Bekleidungstheorie verweisen Ornamente unmittelbar darauf, dass die Wand ihren Ursprung in textilen Wandbehängen besitze."*[80]

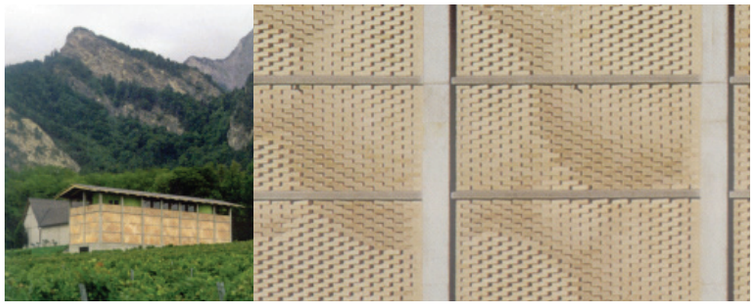


Abbildung 39: Die Maueren des Weingutes Gantenbein wurden mittels Industrieroboter aufgebaut.

Im Gegensatz dazu pendelt laut Gleiter im maschinellen Zeitalter die Betonung mehr in Richtung einer Gemachtheit und den Verfahrenscharakter einer Konstruktion.

Als bekanntes Beispiel nennt Gleiter Otto Wagners (1841-1918) Fassade der Postsparkasse in Wien. Die dort verwendeten Aluminiumanker verweisen auf den Herstellungsprozess. Die Anker dienten zur Positionierung der Fassadenplatten und wurden erst nach dem Aushärten des Mörtels überflüssig. An dieser Stelle sollen weitere Differenzierungen und komplexe Theorien, wie sie Adolf Loos (1870-1933) in seinem Text *„Ornament und Verbrechen"* oder Michael Müller in seinem Buch *„Die Verdrängung des Ornaments – Zum Verhältnis von Architektur und Lebenspraxis"*[81] entwickelt haben, zwar erwähnt, aber nicht weiter behandelt werden. Zwar verschiebt sich laut Gleiter mit den neuen digitalen Technologien, der Mass Customization und der Scripting-Software der Fokus des „neuen Ornaments" in Richtung der konstruktiven Seite, doch kommt es aufgrund des starken Bezugs zum digitalen Entstehungsprozess zu keiner Neuauflage des analogen Maschinenornaments.

Nach diesen Kriterien bewertet Gleiter die Fassade des Weinguts Gantenbein (siehe Abbildung 39) als „Muster" und nicht als „Ornament", was nichts über die Qualität des Projektes aussagen soll. Der Bildeffekt der Fassade weise, wenn dies überhaupt in der klassischen Tradition des Ornaments zu sehen sei, durch die vielschichtigen Effekte keine Beziehung zu den digitalen Techniken, die sie erzeugt haben, auf. Die mittels Roboterarm aufgebaute Ziegelmauer ist zwar innovativ und komplex organisiert, könnte aber dennoch von Menschenhand gefertigt sein. Als Beispiel für ein „neues Ornament" nennt Gleiter den Entwurf des Bahnhofs von Florenz von Arata Isozaki (siehe Abb. 47-49). Die baumartig verzweigte fließende Gestalt des Tragwerks deutet auf den computerbasierten Entstehungsprozess hin, der mittels eines Algorithmus die optimale Form für eine durch die angreifenden Kräfte verursachte gleichmäßige Spannungsverteilung errechnet hat. Demgemäß liegen die Bedingungen für das neue Ornament *„in der interaktiven Verknüpfung zwischen Design- und Konstruktionsverfahren."*[82]

Bemerkung:

Hiermit sei bemerkt, dass Michael Müller eine sehr ausführliche und interessante Analyse des Ornaments darbietet. Insbesondere analysiert Müller Abhandlungen und Texte von Autoren wie Walter Benjamin (1892-1940), Ernst Bloch (1885-1977), Theodor Adorno (1903-1969), Max Horkheimer (1895-1973) und Siegfried Kracauer (1889-1966). So betont beispielsweise Bloch in Bezug auf die Gründerzeit die Abwaschbarkeit der neuen Architektur und die Angst vor Utopien, Kracauer das Ornament der Masse, das sich in der Dynamik, Bewegung, Choreografie und Organisation der neuen urbanen Kultur wiederspiegelt oder Benjamin den Verlust der Aura und die Wertminderung des Originals im Zeitalter der technischen Reproduzierbarkeit von Produkten. In seiner Theorie weist Müller darauf hin, dass selbst bei Loos nie wirklich eine Abwesenheit des Ornaments feststellbar ist. Nach der scheinbaren Verbannung und Brandmarkung des Ornaments als Spiegel niedriger sexueller Triebe findet sich das Ornament in den von Loos verwendeten Materialien wieder. Jörg H. Gleiter verweist in dem hier behandelten Text darauf, dass der Cipollinmarmor, den Loos üppig in Innenräumen einsetzt, in seiner Texturierung auf den prähistorischen Entstehungsprozess hinweist.

File to Factory, Mass Customization und Nonstandard Architecture

"Hier ist Geld, warum bauen sie das nicht?". Mit dieser Frage versuchten die Autoren gegen Ende ihres Textes mit dem Titel „Computeranimismus" im Jahr 1995 auf einen ihres Erachtens wunden Punkt der digitalen Avantgardearchitektur hinzuweisen. Damit waren beispielweise Projekte von Frank O. Gehry oder Greg Lynn gemeint, die die Autoren unter den Computer-Utopismus der Neuzigerjahre einordneten. So lautete das Resümee in dem genannten Text wie folgt: *„Diese Projekte sind ... überaus provozierend, aber mindestens ein Teil der fragilen Gefährdung dieser Architektur liegt in ihrem eigenen Wunsch, wie gebaut zu wirken, und auch dies gehört zum Ethos des Computers"*[83].

Dementsprechend fokussiert etwa 15 Jahre später in der aktuellen Diskussion ein Teil der Fragestellungen auf die neuen digitalen Fertigungstechnologien und der effektiven Herstellung von komplexen Objekten und den damit verbundenen Chancen und Wegen, die sich daraus für zukünftige Architektur ergeben könnten. Unter dem Slogan „File to Factory" versucht man nun, die gesamte Kette des Architekturproduktionsprozesses - parallel zur Ideen- bzw. Konzeptfindung mittels CAD-Software an dem einen Ende - bis hin zur vollständig automatisierten baulichen Umsetzung durch CAM-Systeme an dem anderen Ende, zu erweitern. Dabei wird auf Fertigungstechnologien wie Industrieroboter oder CNC-Fräsen zurückgegriffen, die im Gegensatz zum Menschen eine große Menge an Daten kontrollieren und präzise verarbeiten können und bei mehrmaliger Wiederholung der Arbeitsschritte am Endprodukt keine Qualitätsschwankungen erzeugen, da sie keinen Gemütsänderungen oder ähnlichem unterliegen. Wenn gewünscht, können diese Schwankungen gezielt maschinell über eine parametrisierte Modellierung produziert werden, wodurch an jeder neuen Objektkopie bewusst Adaptionen bzw. Individualisierungen vorgenommen werden können. Dadurch kann eine Vielzahl von Varianten hergestellt werden, wobei allerdings keine einzige eine „Ausnahme" darstellt, denn sie alle folgen demselben Algorithmus.[84]

Unter dem Schlagwort „Mass Customization" (zu Deutsch „Massenindividualisierung") wird versucht, über mehr oder weniger stark modifizierte Reproduktionen eines Objekts auf die Vorlieben der Kunden einzugehen und damit schlussendlich „Unikate" anzufertigen. Der Begriff „Mass Customization" steht als Oxymoron einerseits für die industrielle Massenproduktion (mass) und andererseits für die individuelle Anpassung (customization)[85].
„Die Mass Customization grenzt sich von der Variantenfertigung ab, kann jedoch in Baukastensystemen organisiert sein,..."[86]
So kann man heute über den Einsatz von parametrisierten Objekten und digitalen Steuerungen etwa individuelle Hemden, Hosen, Schuhe, Kinderwagen, Sportwagen und auch Einfamilienhäuser über das Internet konfigurieren bzw. herstellen lassen[87]. Zu letzterem Beispiel sei bemerkt, dass eines der wesentlichen Merkmale von Architektur die Komplexität eines Gebäudes darstellt, weshalb sich in diesem Fall die Konfigurationsmöglichkeiten stark auf die Gestaltung von Oberflächenmerkmalen oder das Hinzufügen von Accessoires beschränken. Um diese Produkte kostengünstig produzieren zu können, stecken Hersteller viel Geld in die Entwicklung und Einrichtung von neuen Fertigungssystemen. In diesem Kontext traut man diesen industrialisierten Fertigungstechniken zunehmend die „Standardisierung" von *„Nonstandard"* Produkten zu, was schlussendlich zur vermehrten Verbreitung von Unikaten führen sollte und damit zur sogenannten Massenindividualisierung beitragen würde. Hier warnt Fabian Scheurer von der Firma „Design to Production" davor, Etiketten wie *„Mass Customization"* oder *„Nonstandard"* nicht einfach nur als *„Buzz Words"* zu verwenden, sondern verlangt, um eine ernsthafte Diskussion zu ermöglichen, diese Begriffe jeweils sauber in ihrem Kontext zu definieren.

„Bezieht sich beispielsweise das „Nonstandard" auf die Form des Gebäudes oder auf die Form und den Herstellungsprozess einzelner Komponenten?"[88]

Weiters zweifelt Scheurer auch an dem Zauberwort „File to Factory" und weist darauf hin, dass es sich vom Modellbaumaßstab nicht einfach zur 1:1 Produktion hochskalieren lässt. ArchitektInnen würden zwar immer häufiger Laserschneider und 3d-Drucker im Modellbau einsetzen, womit relativ einfach komplexe dreidimensionale Volumina hergestellt werden können, doch verleite dies zum Vergessen, dass bei einer 1:1 Umsetzung eine Maschine beispielweise Monate an der Fertigung von Teilen arbeiten und dabei verschiedenste Detailprobleme auftreten würden, deren Lösung Expertenwissen verlange. Im Realisierungsprozess müsse eine hohe Verantwortung übernommen und Entscheidungen gefällt werden, die nur sehr begrenzt an CAD/CAM-Software delegiert werden könne. *„Eine*

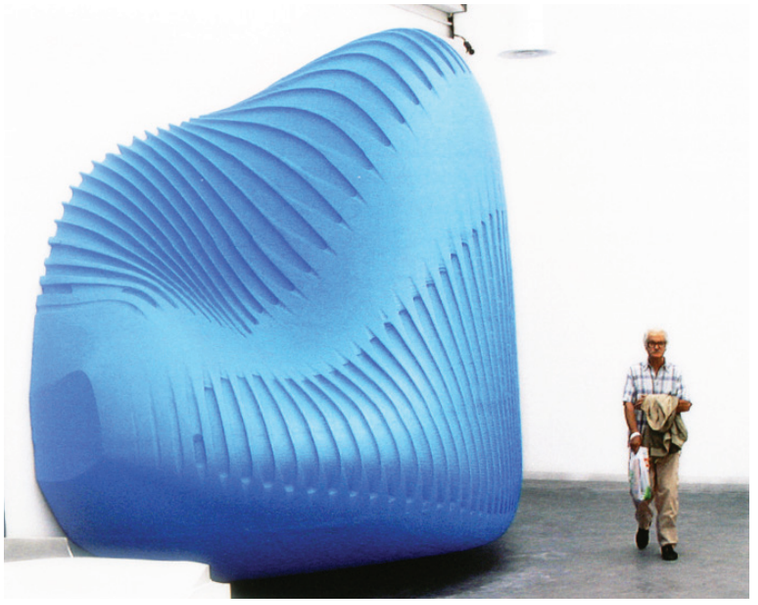


Abbildung 40: Greg Lynn. Nonstandard - Houses on Demand. Die Embriological Houses können je nach Bedürfnissen der Nutzer am Computer generiert werden und maschinell gefertigt werden.

5-Achs-Fräse kann denselben Punkt im Raum aus beliebig vielen Richtungen anfahren und irgendwer muss dann entscheiden, welche davon am besten geeignet ist, um ein optimales Ergebnis zu erzielen. Wenn man dann noch mit einem anisotropen Material wie Holz zu tun hat, das sich nicht in allen drei Dimensionen homogen verhält, weil es eine bestimmte Faserrichtung hat, dann ist plötzlich eine ganze Menge Spezialwissen gefragt"[89]. Das größte Potential für die digitale Architekturproduktion und die Entwerfer sieht Scheurer in der Entwicklung von parametrisierbaren Regelsystemen, die an allen Stellen eines Gebäudes funktionieren, da seiner Ansicht nach kein Computer diese Abstraktionsleistung schaffen könne.

Da generische Fertigungsmaschinen wie Industrieroboter nicht für eine bestimmte Tätigkeit spezialisiert sind und durch verschiedenste Aufsätze universell eingesetzt werden können, sind genaue Handlungsanweisungen und auch Kenntnisse über die Arbeitsweise der Maschinen notwendig. Dadurch tut sich ein weiteres Feld für die Architektur in der Entwicklung neuer Konstruktionen und Bauteile auf. Durch die bewusste Nutzung spezifischer Eigenschaften einzelner Fertigungsmaschinen ergibt sich die Möglichkeit, die Gestaltung von Fertigungsprozessen neu zu überdenken und diese Auseinandersetzung zum Teil der architektonischen Entwurfsplanung zu machen.
Beispiele für aus der Auseinandersetzung mit dem Fertigungsprozess resultierende innovative Lösungen bringen die Autoren des Textes „Digitales Handwerk" in Projekten wie dem „Weingut Gantenbein" und „Die sequenzielle Wand" näher. In diesen Projekten wurden Ziegelwände mittels eines Industrieroboters aufgebaut. So stellt die individuelle Rotation und Position einzelner Ziegelsteine für den Roboter einen geringen Aufwand im Vergleich zu jenem für den Menschen dar, wohingegen andererseits für die maschinelle Fertigung für den Zusammenhalt der Ziegel ein Kleber gewählt werden musste, da das gleichmäßige und vollflächige Auftragen von Mörtel, die dabei exakte waagrechte Positionierung des Ziegelsteins in dem Mörtelbett und das Abstreifen des überschüssigen Materials für den Roboter im Vergleich zum Menschen einen unvertretbaren Aufwand bedeuten würde.[90]

Neben der Entwicklung neuer Konstruktionen, die aus der Auseinandersetzung mit dem Fertigungsprozess resultieren, können dem Bauteil durch die präzise Bearbeitungsgenauigkeit und die Handhabung großer Datenmengen zusätzliche Qualitäten und eine noch nie dagewesene Detaillierung zugeschrieben werden. *„Anstatt Bauteile gemäß dem industriellen Paradigma in funktional getrennten Schichten aufzubauen, ermöglicht das digitale Handwerk, durch Informierung einfacher Materialien leistungsfähige und homogene Bauteile zu erstellen."*[91]

„Digitale Materialität" beschränkt sich nicht auf physische Materialeigenschaften oder Gesetze wie der Gravitation, sondern wurzelt in der digitalen Welt und bedient sich daher der Regeln der Virtualität. Somit kann ein physischer Bauteil mit virtuellen Eigenschaften versehen werden, wobei dieser beispielsweise über die Repräsentation von organisierter

Abbildung 41: Die ETH Zürch kann mittels ihrers Industrieroboters Ziegelwände voll autmatisiert aufbauen lassen.

Komplexität um verschiedenste Informationsebenen erweitert werden. Demzufolge können Materialien durch „Informierung" unterschiedlichste Wahrnehmungsebenen affizieren.

Dies bedeutet, dass durch den intelligenten Einsatz der neuen digitalen Fertigungsmethoden durch eine bewusste Planung bzw. Kontrolle des Fertigungsprozesses und der digitalen „Informierung" von Bauteilen Materialien, die bereits seit Jahrtausenden bekannt sind, in neuen Formen und Einsatzgebieten erscheinen können.[92]

Algorithmische Architektur - Der Architekt als ProgrammiererIn

Der Begriff „Algorithmus" geht auf den Namen des arabischen Mathematikers „Al-Khwarizmi" zurück. Ein „Algorithmus" ist eine Prozedur, um Probleme über eine festgelegte Anzahl von Schritten lösen zu können, wobei logische Operationen verwendet werden.[93] Diese Operationen sind in Programmiersprachen als „Befehle" in einem strengen Regelwerk organisiert. Dieses Regelwerk wird als „Code" bzw. „Kode" in Form von sogenannten „Quelltexten" dargestellt und unterliegt einer Syntax - ähnlich der Grammatik einer Sprache. Der Begriff „Code" geht auf das lateinische Wort „Codex" zurück, welches die hölzernen Tafeln bezeichnet, auf denen die Römer ihre Prinzipien und Regeln für das römische Imperium schriftlich festgehalten haben. In diesem Sinn wird auch das Regelkonstrukt, welches die Information über die Rahmenbedingungen für die Ausführungen eines „Algorithmus" definiert, als „Code" bezeichnet. Ingeborg Rocker bemerkt in diesem Zusammenhang, dass Architektur schon immer in allen Planungsebenen einem strengen Regelwerk unterlag und damit seit jeher „kodiert" war. Architektur könne nicht „nichtkodiert" sein.[94] „Architecture is always already coded." Weiters bemerkt Claus Dreyer, dass durch den "Code-Begriff " eine seit den 1960er Jahren zunehmende Problematisierung des überkommenen „Stil-Begriffes" stattfände und diese Entwicklung noch nicht abgeschlossen sei.[95]

Ein wichtiger Punkt in der Diskussion rund um parametrische bzw. algorithmische Architektur ist die Transformation des Entwurfansatzes vom *„architecture programming"* zu *„programming architecture"*[96]. In diesem Kontext bemerkt Antoine Picon:
„Im Unterschied zu den traditionellen Werkzeugen des Architekten schlagen Grafikprogramme dem Nutzer implizit bestimmte Arten von geometrischen Lösungen vor. Um zu verhindern, dass man an ein System von Regeln gebunden wird, die der Architekturpraxis fremd sind, wird es unerlässlich, der Maschine und der Software selbst gewisse Regeln vorzugeben."[97]

So kann Architektur mit Hilfe neuer Techniken wie dem Scripten nun beispielsweise geschrieben bzw. programmiert oder auch generiert werden, womit der/die ArchitektIn vom reinen „Sotwareanwender" zumindest teilweise zum „Entwickler" wird. Über die Verwendung von veränderlichen Parametern oder etwa dem Einsatz von generischen Algorithmen liegt der Fokus nun auf einer dynamischen Gestaltung des Entwurfsprozesses. Der/Die Architektin besitzt die Möglichkeit, über den Einsatz digitaler Informationstechnologien sein Entwurfswerkzeug selbst weiter zu entwickeln und den Entwurfsprozess zu kodieren, womit der/die ArchitektIn zum/zur „Architektur-ProgrammiererIn" wird. Im Gegensatz dazu sei bemerkt, dass in der Informatik bei der Entwicklung von Software vom „Entwerfen" von Programmen bzw. in diesem Kontext auch von „Softwarearchitekturen" gesprochen wird, sodass der/die InformatikerIn zum/zur „Software-ArchitektIn" wird.
Um jedoch den Entwurfsprozess über digitale Codierung individualisieren bzw. dynamisieren zu können, muss sich der Architekt bzw. Designer digitalen Regeln unterwerfen. Denn für die digitale Übersetzung analoger Problemstellungen ist trotz der immer benutzfreundlicheren Bedienungsoberflächen nach einem Verständnis der Logik auch eine Kenntnis der Sprache des Entwurfmediums nötig. In der Informatik versucht man seit längerem unter dem Schlagwort *„End-User-Programming"*[98] es Nutzern von Software zu ermöglichen, diese zu erweitern, anzupassen oder selbst eigene Programme zu erstellen.
Derzeit wird unter anderem versucht, Software oder Softwarebauteile als Services anzubieten. In einer sogenannten Service-orientierten sollen Endbenutzer dann intuitiv ohne tiefgreifende Programmierkenntnisse Anwendungen an ihre individuellen Ansprüche anpassen, Software erweitern oder Programme selbst erstellen können. Die in dieser Arbeit bereits erwähnten *„Mashup-Tools"*[99] stellen Werkzeuge für diese Art der Softwareentwicklung dar. Wie schon vorangehend in der vorliegenden Arbeit (Seiten 31-33) erwähnt, sollen solche Werkzeuge dem Anwender ohne große Kenntnisse der Scripting- bzw. Programmiersprachen die Gestaltung von individuelle Softwarelösungen über die Vernetzung von fertigen Funktionsboxen ermöglichen. Allerdings befindet sich diese Entwicklung derzeit noch in einer frühen Phase und die meisten „Mashup-Tools" können gegenwärtig nur von EndbenutzerInnen mit zumindest geringer programmatischer Kenntnis eingesetzt werden. Neue Ansätze erlauben den EndbenutzerInnen selbst Anforderungen zu dokumentieren, die die Grundlage für die Entwicklung von an die Wünsche der Benutzer angepasster Software darstellen.
Ein Beispiel für einen derartigen Ansatz ist das *„iRequire"*[100] Werkzeug, welches eine benutzerfreundliche Softwareoberfläche und Struktur bereitstellt, mit der ein individuelles Problem geordnet, erfasst, kategorisiert und via Internet einer breiten Gemeinschaft zur Bearbeitung zur Verfügung gestellt werden kann. Ziel von „iRequire" ist es, Endbenutzern zukünftig zu ermöglichen, einen Entwicklungswunsch in allgemein verständlicher Form über eine Plattform im Internet einer breiten Masse zugänglich machen. Die Rückmeldung auf so einen Wunsch könnte eine fertige Softwarelösung sein oder Lösungsangebote, Vorschläge und Hinweise in einer forumähnlichen Plattform. „I.Require" zielt derzeit darauf ab, Softwareanwendungen zur Lösung alltäglicher Probleme, die zum Beispiel im Verkehr oder beim Shopping auftreten, zu entwickeln, doch lasse sich diese Art von Problemerfassung nach Ansicht der Entwickler auch auf architektonische Problemstellungen anwenden.

Meines Erachtens ist ein spannender Punkt an „iRequire" eine möglichst klar verständliche Erfassung und Differenzierung des Problems. Denn über eine klare Darstellung der Problemstellung kann eine möglichst direkte Zuordnung zu bereits fertigen „Design Patterns" oder Themengebieten erfolgen, wodurch das Problem beispielsweise in Zukunft mit einem „Mashup-Tool" schnell und einfach gelöst werden könnte. So könnte der/die ArchitektIn oder DesignerIn mit marginaler programmatischer Kenntnis und geringem Bezug zu Softwareentwicklung zukünftig individuelle parametrische Systeme entwickeln, die seinen/ihren Bedürfnissen entsprechen.[101]

Zudem erreicht mit der Transformation analoger Gegebenheiten in digitale Codes die Verwissenschaftlichung der Architektur - deren Anfangsphase in dieser Arbeit bereits im Abschnitt „Visionen und Entwicklungen als Voraussetzung für CAD" mit der Forschung rund um Max Bense beschrieben wurde - einen neuen Höhepunkt. Mit systematischen Problemlösungsstrategien werden analoge Vorgänge, Eigenschaften etc. in digitale logische Muster und Parameter abstrahiert. So sind zum Beispiel die im „Parametrismus" verwendeten mathematisch beschreibbaren Parameter die digitalen Äquivalente analoger Eigenschaften, welche in der Regel durch Zahlenwerte ausgedrückt werden. Die in den 1970er Jahren entwickelten systematischen Entwurfsmethoden und Strategien, die für eine rationale Herangehensweise im architektonischen Entwurf entwickelt wurden, dienen als direkter Ausgangspunkt für aktuelle Ansätze.

Neben der zunehmenden Einbindung der maschinellen Fertigung in den Entwurfsprozess zeichnen sich aktuell zwei weitere Argumentationstendenzen in der digitalen Avantgardearchitektur ab, die jeweils eine starke Verwurzelung in Arbeiten der 1970er Jahre aufweisen. Zum Einen wird der Begriff der „Partizipation" - gepaart mit dem Begriff der „Interaktivität" - zu einem wichtigen Argument in der digitalen parametergesteuerten Architektur und zum Anderen wird unter dem Schlagwort der „künstlichen Kreativität" Formfindung über die Nachahmung natürlicher selbstorganisierender Systeme betrieben.

Der Begriff der „Partizipation" wurde besonders in den 1970er Jahren geprägt und soll über die Verwissenschaftlichung, die Erläuterung und eine strukturierte symbolische Darstellung von Architekturelementen einem Laientum die Mitsprache am architektonischen Entwurfsprozess ermöglichen. Hierbei sei im Besonderen auf die „Pattern Language"

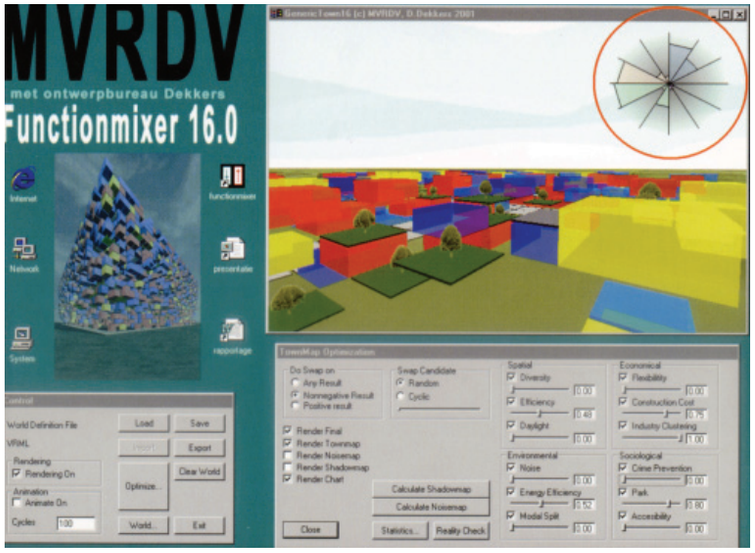


Abbildung 42: Der „Funktionmixer" von der Architektengruppe MVRDV. Es können 12 Parameter berücksichtigt werden.

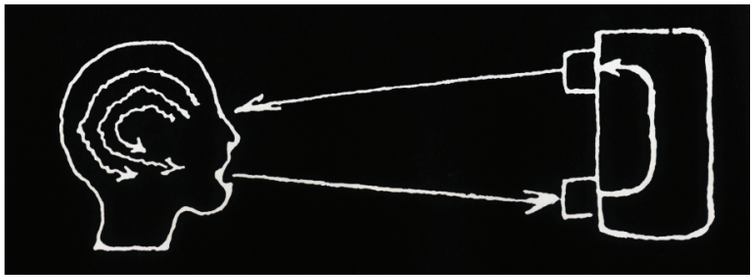


Abbildung 43: Grafik zur Interaktivität aus Negropontes "The Architecture Machine".

Christopher Alexanders und die Arbeits- bzw. Entwurfslehrebücher von Jürgen Joedickes oder Ernst Neuferts (1900-1986) hingewiesen, die mit ihren Schematisierungs-, Kategorisierungs-, Analysemethoden und Diagrammen die Ausübung der Architektur als Wissenschaft förderten. So wird heute beim Entwerfen von Algorithmen auf fertige Problemlösungsmuster zurückgegriffen, die in der Informatik in Anlehnung an Christopher Alexanders „Pattern Language" als „Design Patterns" bezeichnet werden und ähnliche Strukturen aufweisen.[102] Die Idee der „Partizipation" wird in parametergesteuerten Entwürfen aufgegriffen, wo einem breiten Benutzerkreis Zugang zum Entwurfsprozess verschafft werden kann. Als Reaktion auf den neuen Umgang mit digitaler Komplexität mittels parametrischen Modellen nennt Michael Meredith eine Grundeigenschaft des parametrischen Designs: *„...parametric design typically reduces the number of formal variables, but maximizes their variability through transformational affects which are engendered via quantity."*[103]

Somit können unterschiedlichste Nutzer über eine Softwareplattform untereinander und mit dem parametrischen Entwurfsmodell in Interaktion treten und durch Veränderung dieser reduzierten ausgewählten Parameter somit Varianten und Kombinationen des Entwurfs erzeugen. So werden in manchen Entwürfen einige Parameter dem/der zukünftigen NutzerIn bereitgestellt, wobei dieser beispielsweise über Internet Veränderungen vornehmen kann. Darüber können etwa WohnungskäuferInnen ihre Wünsche äußern, indem sie bereitgestellte Elemente beliebig kombinieren können und Parameter wie Farben, Materialien etc. auswählen und dadurch den Entwurfsprozess wesentlich mitbestimmen. Der/Die ArchitektIn stellt also eine Plattform zur Verfügung und übernimmt im Entwurfsprozess die Rolle eines Moderators/einer Moderatorin, wodurch er/sie einen Teil seiner Autorenschaft abgibt. Als Beispiel solcher Entwurfsansätze werden hier der „Functionmixer" von MVRDV und ein Projekt von KAISERSROT angeführt.

Der „Functionmixer" ist eine eigens entwickelte Software, die einen offenen Prozess widerspiegelt und ermöglicht. Mit dieser Software können unterschiedliche Funktionen wie beispielsweise Büros, leichte Industrie und Wohnen räumlich organisiert werden, wobei jeweils verschiedenste Eigenschaften und Bedürfnisse in Form von Parametern vergeben werden. Der „Functionmixer" versucht eine optimale städtebauliche Anordnung zu organisieren, indem 12 Kriterien wie Akustik, Dichte, Lichteinfall, Flexibilität, Kriminalität etc. berücksichtigt werden.[104]

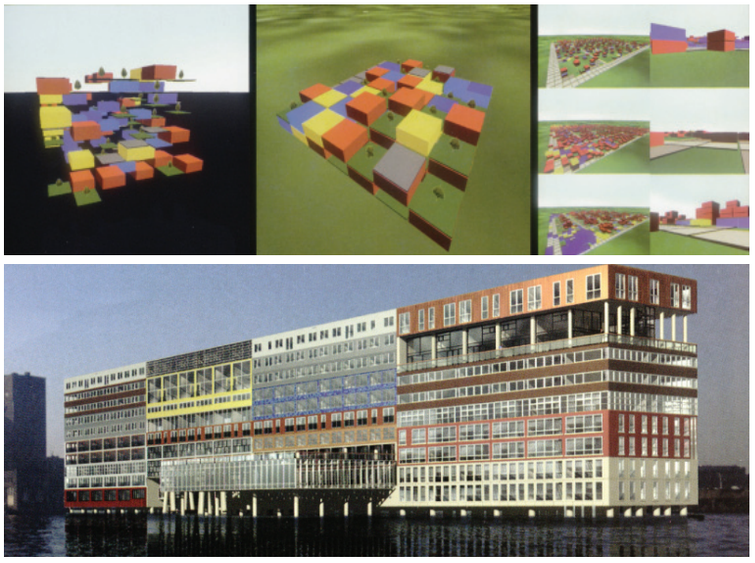


Abbildung 44: Mittels des „Funktionmixer" lassen sich unterschiedliche Variationen von städtebaulichen Raumorganisationen generieren. Das Projekt „Wohnsilo" in Amsterdam wurde von MVRDV mittels „Functionmixer" organisiert und wie abgebildet in die bauliche Realität umgesetzt.

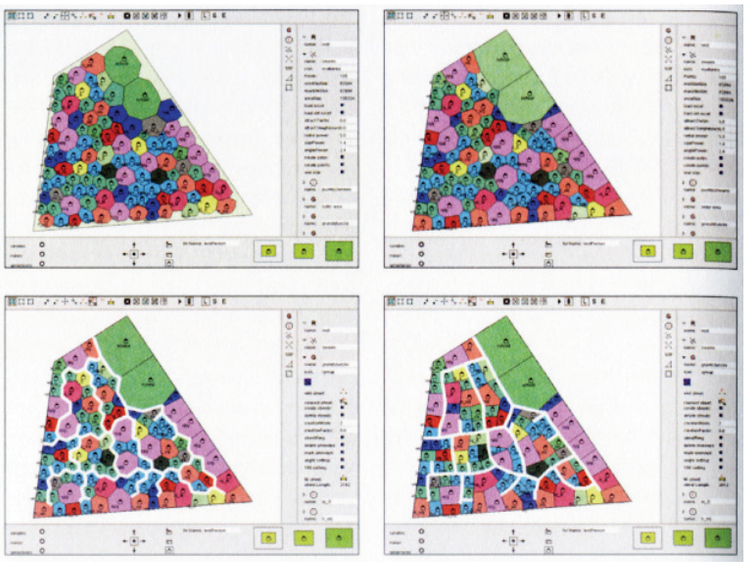
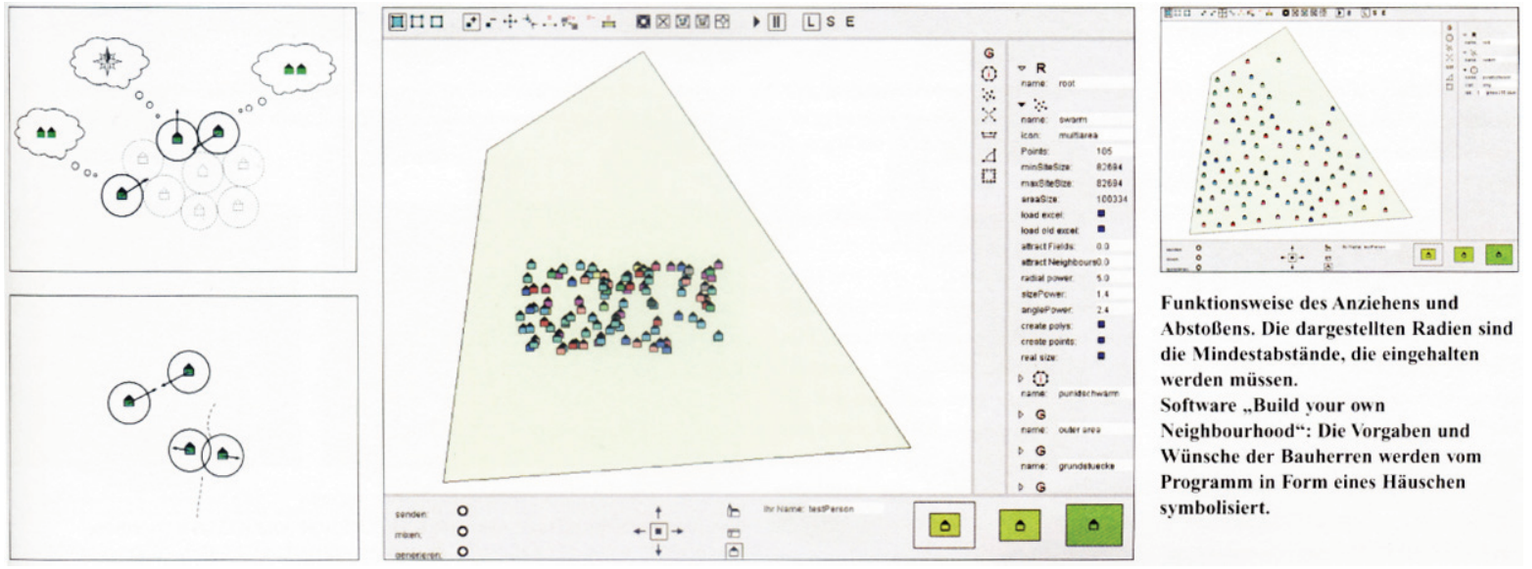


Abbildung 46: Hier werden die vier im Text bereits beschriebenen Entwicklungsschritte von der Aufteilung der Grundstücke bis zur Begradigung der Straßen dargestellt.

Ein weiteres Beispiel für ein parametergesteuertes Entwurfsmodell ist das Projekt „Build your own Neighbourhood" der Gruppe KAISERSROT. Mit einer eigens entwickelten Software lassen sich Siedlungsstrukturen planen, wobei die Wünsche der zukünftigen BewohnerInnen in die Planung einfließen können und daraus je nach Bedürfnissen eine optimale Aufteilung der Grundstücke, Zufahrtsstraßen etc. berechnet wird. So können zukünftige BewohnerInnen *„ihre Wünsche bezüglich Größe der Parzelle, Typologie und Ausstattung der Häuser sowie gewünschte Nähe zum Wald, Wasser, Bushaltestelle oder Lieblingsnachbarn angeben."* Laut Oliver Fritz könne man sich die Häuser wie in einem Schwarm organisiert vorstellen, wobei diese sich je nach Eigenschaften wie Magneten in einem Kräftefeld untereinander mehr oder weniger anziehen oder abstoßen oder zu städtebaulichen Anziehungspunkten hingezogen werden. Kommt ein neues Haus hinzu, wird das System wieder neu ausbalanciert. In einem ersten Schritt werden die Häuser auf dem gesamten Areal aufgeteilt und optimal platziert. Danach erfolgt eine Aufteilung des Areals in polygonale Grundstücke. In weiterer Folge wird ein ideales Straßensystem für die Erschließung berechnet. Zuletzt werden die Straßenzüge begradigt und die Grundstückszuschnitte nachkorrigiert. Nach der erfolgreichen Ausstellung dieses Projekts im Niederländischen Architektur Institut wurde die Software für die Planung einer neuen 10 Hektar großen Siedlung im Stadtteil „Schuytgraaf von Arnheim (NL)" eingesetzt.[105]



Funktionsweise des Anziehens und Abstoßens. Die dargestellten Radien sind die Mindestabstände, die eingehalten werden müssen.
Software „Build your own Neighbourhood": Die Vorgaben und Wünsche der Bauherren werden vom Programm in Form eines Häuschen symbolisiert.

Abbildung 45: Die Software der Gruppe KAISERSROT ermöglicht die Organisation von Häusern und Aufteilung von Grundtücken mittels Algorithmen unter Berücksichtigung einzelner Bedürfnisse der Bewohner.

Neben der „Partizipation" ist eine zweite Tendenz rund um den Begriff der „künstlichen Kreativität" erkennbar. Diese entspringt den Theorien der Kybernetiker und geht insbesondere auf die Arbeiten des Architekten John Hamilton Frazer zurück. Frazer beschäftigte sich in den 1970er Jahren besonders mit evolutionären Strategien und der damit verbundenen Entwicklung von generischen Algorithmen, wobei er diese für den digitalen Formfindungsprozess in der Architektur einsetzte. Durch dei Verwendung von generischen Algorithmen oder sogenannten Agentensystemen können in der Natur vorkommende selbstorganisierende Systeme künstlich simuliert bzw. nachgeahmt werden. Hierbei kann der linear determinierte Prozess gebrochen werden und der ursprünglich festgelegte „Code" in endlosen Schleifen „de-codiert" und „re-codiert" werden. Hierbei kann ein völlig neuer Informationsgehalt erzeugt werden, weshalb man hierbei dem Computer „künstliche Kreativität" zuschreibt. So spricht man im Kontext des sogenannten „Selfgenerativ Design" nicht mehr von „Formgebung", sondern von „Formgenerierung". Damit scheint mE Negropontes Vision vom Computer als Ideenlieferant wahr geworden zu sein. Wurden früher komplexe Systeme mit hohem Aufwand vom großen Maßstab zum kleinen abgearbeitet und dabei relativ starr organisiert, kehrt sich jetzt der Entwurfprozess um. Komplexe Systeme werden „Bottom-up" vom kleinen in Richtung großen Maßstab entwickelt.[106] Dabei orientiert man sich an natürlichen Systemen wie beispielsweise Ameisenkolonien, Bienen- oder Fischschwärmen. Beispielswiese ist das Gedächtnis einer einzelnen Ameise im Verhältnis zur gesamten Ameisenkolonie gering. Dazu ein sehr weit verbreitetes und oft zitiertes Beispiel:

„Eine Ameise die Futter gefunden hat, versprüht auf ihrem Weg zurück zum Nest ein Pheromon. Andere Ameisen folgen bei der Futtersuche der Spur dieses Duftstoffes und produzieren beim Finden erneut Pheromone. Mit der Zeit wird auf dem kürzesten Pfade zwischen Nest und Futterstelle die höchste Pheromonkonzentration vorherrschen. Die Ameisen werden diesen Weg bevorzugt wählen. Es entsteht eine Ameisenstraße."[107]

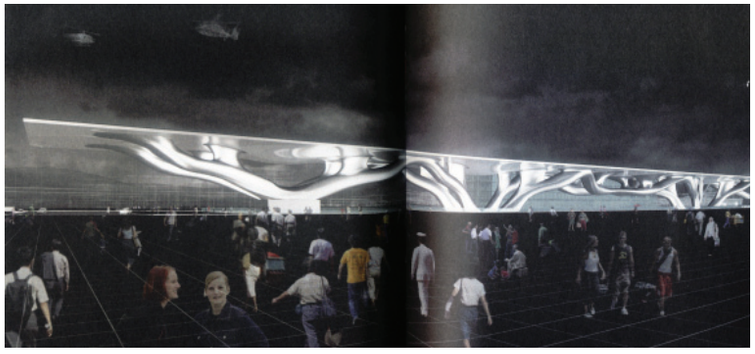


Abbildung 47: Visualisierung des Entwurfs des Bahnhofs von Florenz von Arata Isozaki.

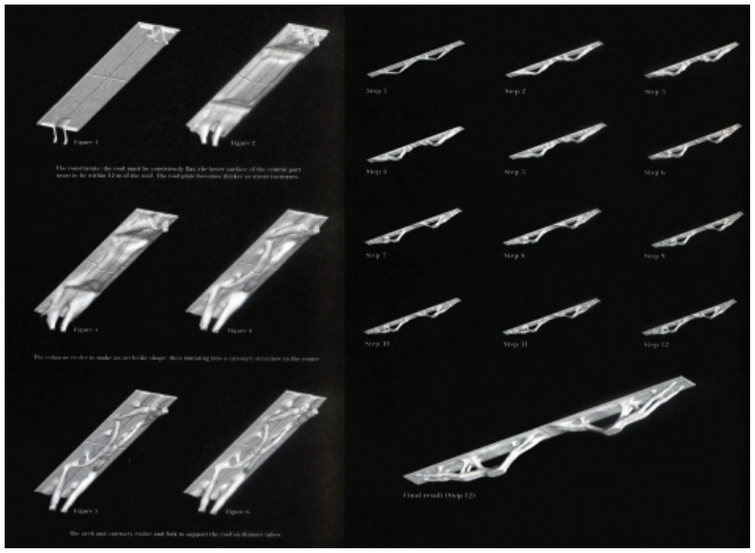


Abbildung 48: In den zwei linken Bildern kann man den evolutionären Formfindungsprozess erkennen. Schritt für Schritt wird nach der optimalen Form für die Lastableitung der Kräfte von der Deckenplatte zu den zwei definierten Auflagerpunkten am Boden.

Diese kollektiven Problemlösungen werden den sogenannten „Agentensystemen" zugeordnet, wobei die Lösungsfindung mittels „Trial and Error" einzelner Artgenossen bzw. „Agenten" erfolgt. Ein anderer der Natur entlehnter Ansatz findet sich in den evolutionären, genetischen bzw. generischen Algorithmen. Evolutionäre bzw. genetische Algorithmen orientieren sich im Gegensatz zu generischen Algorithmen an natürlichen Systemen, wobei die letzteren etwas allgemeiner „aus sich selbst heraus erzeugte Formen" bezeichnen. Evolutionäre, genetische bzw. generische Algorithmen erzeugen schrittweise von Generation zu Generation bessere Lösungen. Anfangs wird eine Anzahl von zufälligen Lösungen generiert. Diese Lösungen werden nach einer Zielfunktion bzw. nach Qualitätskriterien bewertet. Die als günstigst bewerteten Ergebnisse werden als „the fittest" leicht verändert (mutiert) und danach mit den anderen Lösungen kombiniert (gekreuzt). Somit enthält die nächste Generation von Lösungen im Mittel bessere Ansätze als die vorhergehende. Nach diesem Prinzip werden von Generation zu Generation immer bessere Lösungen für die gestellte Aufgabe erreicht.[108] *„Daher erfolgt die Bestimmung der Systemvariablen nicht zielgerichtet, auch weil das Ziel in den meisten Fällen zunächst nicht definiert werden kann."*[109] Umkehrt werden diese Algorithmen genau deshalb für die Lösung und Analyse komplexer Systeme eingesetzt, weil bei einer großen Anzahl an Einflussparametern keine Aussage über Prioritäten, Auswirkungen bzw. Vorhersagen auf das Gesamtsystem gemacht werden kann. Somit richten sich mE ArchitektInnen und DesignerInnen bei der Gestaltung digitaler Formfindungsprozesse und Aufgabenstellungen aktuell gegen streng determinierte, linear kausale Systeme und den sogenannten Laplace'schen Dämon und orientieren sich an der Chaos- beziehungsweise Emergenztheorie.

Als Beispiel das Einsatzgebietes eines generischen Algorithmus wird hier der Entwurf des Bahnhofs von Florenz von Arata Isozaki gewählt (siehe Abb.47-49). Bezugnehmend auf die vorliegende Arbeit wurde dieses Projekt gewählt, da die Form des Entwurfes im Kern aus der Auseinandersetzung mit dem Themengebiet des „Entwerfens mit Kräften" resultiert. Bei der

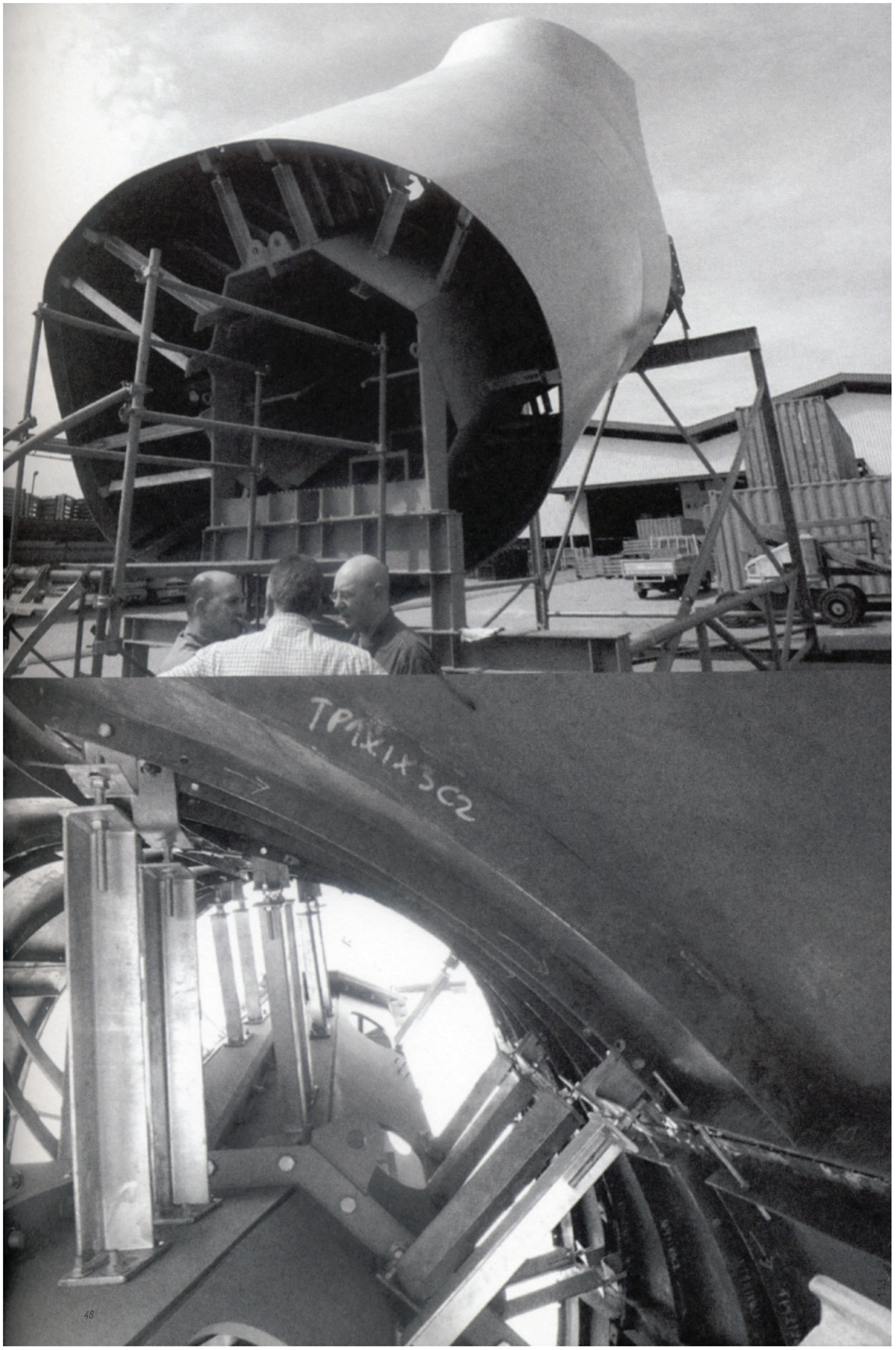


Abbildung 49: Tragkonstruktion des Bahnhofs in Florenz von Arata Isozaki. Im Inneren befindet sich der sechseckige tragende Kern, auf dem Abstandshalterungen zur Befestigung der finalen Freiformoberfläche montiert sind.

Formfindung verwendet Isozaki evolutionären Algorithmus, der in einem iterativen Prozess Schritt für Schritt eine immer optimalere Tragwerksform für das Abtragen der Kräfte von der Deckenplatte zu den zwei definierten Auflagerpunkten am Boden automatisch generiert. Der Optimierungsprozess läuft theoretisch unendlich lange und wird daher abgebrochen, wenn keine wesentlichen Veränderungen mehr ersichtlich sind. Daher spiegelt die am Computer generierte Form ein nahezu optimale Tragwerksform für dieses spezielle Kräftesystem dar.

Abschließend sei bemerkt, dass die Komplexität des Entwerfens eines Tragwerks offensichtlich nicht ganzheitlich bis zur Konstruktion durchgedacht wurde. Dies kann man mE an dem gebauten Endergebnis deutlich erkennen. Das eigentlich relativ konventionelle Tragwerk - achteckiges Stahlblech - befindet sich im Kern, wobei die vorher als optimales Tragwerk argumentierte Form als rein optisch wirksame Hülle mit Abstandshalter auf die eigentliche Kontruktion aufgebracht wird. Dieses Beispiel wird zu einem späteren Zeitpunkt in der Aufgabenstellung der vorliegenden Arbeit noch einmal aufgegriffen.

"Liberal Views have never built anything of value"

Während Architekten/-innen wie Frank O. Gehry, Greg Lynn, Zaha Hadid, Patrik Schumacher und viele andere neue digitale Methoden feiern, indem sie das Potential der neuen Technologien ausreizen und ihre Gebäudekonzepte tendenziell über Begriffe und Themen wie digitale Vernetzung, Interaktivität, Partizipation, Simpel Systems - Simplexity, evolutionäre, genetische und generische Entwurfsprozesse, künstliche Kreativität, fluide natürliche Formensprachen, File to Factory, Mass Customization, Massen-Unikate, Variantenreichtum, Nonstandard Architecture bzw. Nonstandard Structures, performative Architektur, Complex Capacities, intelligente Bauteile bzw. Form Follows Performance bis hin zum „Bilbaoeffekt" als zukunftsträchtig feiern bzw. argumentieren, äußern Kritiker wie etwa Christopher Alexanders, Peter Eisenmann, Rem Koolhaas, Peter Zumthor ihre Bedenken und verweisen auf mögliche Mängel der gegenwärtigen digital-technisierten Architektur.

Antoine Picon

„Ist die digitale Kultur tatsächlich etwas qualitativ anderes als die zur Reife gelangte industrielle Kultur der Moderne?"[110] Mit diesem Einwurf beginnt Antoine Picon in diesem Artikel (siehe Zitat) der Frage nachzugehen, ob die Architektur sich tatsächlich schon in der Postmoderne befinde oder sich im Grunde noch immer von Prinzipien und Regeln der Moderne nähre. Dies sei nicht so einfach zu beantworten, doch lässt sich laut Picon deutlich feststellen, dass es gegenwärtig zumindest zu einer Schwächung des Geistes der Moderne komme. Dafür käme es zu einer Konzentration und Neustrukturierung des Entwurfprozesses und einer Betonung von Aspekten wie der Bearbeitung von Oberflächen und der Fragen um Textur und Licht im Allgemeinen, die früher als sekundär empfunden wurden. Es bestehe ein vermehrtes Interesse an der tektonischen Theorie Gottfried Sempers (1803-1879), die das gleichzeitige Verständnis von Architektur und Ornament bzw. von Projekt und Wand gestatte. So kämen bei zeitgenössischen Arbeiten die Wirkung der Wand und ein neues Gefühl von Materialität zur Entfaltung.

„Diese neue Materialität geht oftmals einher mit der Bereitschaft, sich den Gesetzen der Wirtschaft zu fügen, und mit der Bejahung der bestehenden Ordnung, einschließlich aller Ungleichheiten und Spannungen, anstatt diese, wie es die moderne Avantgarde getan habe, in Bausch und Bogen abzulehnen. Es ist, als hätte man die politischen und sozialen Ideale, die die Herausbildung der Moderne begleiteten, ein für allemal über Bord geworfen zugunsten eines Strebens nach wirtschaftlicher und programmatischer Effizienz."[111]

Damit ist der Kern-, bzw. vorsichtiger ausgedrückt, ein häufig genannter Kritikpunkt der zeitgenössischen digitalen Avantgarde formuliert. So sei die zeitgenössische Architektur laut Picon in einer Welt, in der Effizienz zum höchsten Ideal geworden ist, weniger von einer Entmaterialisierung durch Digitalisierung gefährdet, sondern eher vom Verlust aller politischen und gesellschaftlichen Ordnung und einer damit verbundenen Orientierungslosigkeit bedroht.[112]

Peter Eisenman

„...and I do not believe you can do architecture and not have ideas that come out of the culture in someway."[113] So betont Peter Eisenman in einem Interview den Zusammenhang von Architektur und Kultur bzw. Politik und der damit verbundenen Informierung der Gebäude durch diese Diskurse. In einem anderen Interview präzisiert er wie folgt: *"I think architecture is a form of politics. I believe that architecture does make political statements. There is no doubt."*[114] Als Beispiel nennt Eisenman dabei faschistische Bauten in Italien, die er auch in seinem Buch „Giuseppe Terragni: Transformations, Decompositions, Critiques" analysiert. Eisenman betont, dass sowohl er als auch seine Bauten sehr konservativ seien und sein Klientel dadurch mehrheitlich Republikaner seien bzw. generell eher vom politischen Rechtenlager stammen würde, wobei er sich deshalb aber nicht zwingend als politisch „rechts" einordnen lasse. Doch räumt Eisenman in diesem Kontext ein:
"...I have the most rapport with right-leaning political views, because first of all, liberal views have never built anything of any value, because they can't get their act together."[115]

Damit geht es Eisenman mE grundsätzlich um Ideale bzw. Haltungen und den damit verbundenen Inhalt. Eisenman sei nach seinen Ausführungen zwar kein Fundamentalist, sei aber an den Fundamenten der Dinge interessiert. Doch geschehe eine Politisierung von Architektur auch ohne eine dezidierte Deklaration durch den/der ArchitektenIn. Als Beispiel unter seinen Projekten nennt Eisenman in diesem Kontext das „Denkmal für die ermordeten Juden Europas", welches im Jahr 2005 in Berlin fertig gestellt worden ist. Eisenman wollte damit kein bestimmtes Image suggerieren. Das Denkmal könne an einen Friedhof oder auch an ein Kornfeld erinnern. Es gäbe nichts, das den Holocaust repräsentieren könne. Eisenman selbst

dachte an ein Kornfeld, als er sich einmal in Iowa verloren hatte und daraufhin das Konzept entwickelte. Nachdem der Wettbewerb entschieden worden war, wäre Richard Karl Freiherr von Weizsäcker - der ehemalige Bürgermeister Berlins - auf Eisenman zugekommen und habe gesagt:

"You know, Peter, my problem with your project is this: the left wing hates it because they think it's right wing and the right wing hates it because they think it's left. Nobody can make an assessment. You have created something that is, in a sense, problematic for everybody, because they can't label it. And if they can't label it, then they can't tell whether they like it or dislike it."[116]

Eisenman selbst wolle kein bestimmtes Label oder seine Gebäude mit einer bestimmten Bedeutung beladen und somit nicht rechts, nicht links, nicht gut und nicht schlecht sein. Eisenman wolle Gefühle provozieren. Das „Denkmal für die ermordeten Juden Europas" würde nun als öffentlicher Platz von verschiedensten Menschen für verschiedenste Tätigkeiten, wie Picknicken, Essen, Spielen, Drogen verkaufen etc. verwendet werden. Kinder kämen vom Besuch des „Holocaust Denkmals" zurück und würden ihren Eltern mitteilen: *„We were at the holocaust memorium today, we had a great time!"*[117]. Somit könne Architektur laut Eisenman ein politisches, soziologisches Phänomen re-orientieren. In dem genannten Beispiel würde versucht, eine Brücke zu einem brisanten Thema zu schlagen, dieses in das Alltagsleben zu integrieren und damit ein normales Verhältnis und eine Verarbeitung zu ermöglichen. In diesem Zusammenhang bemerkt Eisenman weiters, dass ein Jude nicht als „Normal" behandelt werden würde, sondern sich auch heute noch in unangenehmen Situation wiederfinden und ständigen Entschuldigungen und Sonderbehandlungen ausgesetzt werden würde. Eisenman beendet seinen Vortrag in Bezug auf das "Holocaust Memorium" mit folgenden Worten: *„You cannot continue to insight guilt... and to never forget does not mean to never forgive!"*[118]

Johan Bettum

In diesem Sinn bemerkt auch Johan Bettum, dass der architektonische Entwurf mehr als eine reine Gestaltungsübung ist und zur Produktion von Raum als ein kulturelles, politisches Projekt etwas beitragen muss. Wie schon Picon stellt Bettum einen Bezug zu Sempers Bekleidungstheorie her, wobei er feststellt, dass die neuen hyperperforierten, *„um nicht zu sagen mottenzerfressnen"* Oberflächen zwar an Textilien erinnern und ähnlich eingesetzt werden, sich jedoch in keiner Art und Weise vom tatsächlichen Aufbau in den Materialeigenschaften stark unterscheiden. Bei genauerer Analyse dieser perforierten Oberflächen entdeckt man nach Bettum

„den intellektuellen Mangel, welcher charakteristisch für die meisten zeitgenössischen Entwürfe ist. Die Gebäude erinnern an reine Skelette als Drahtgitter oder an ausgedehnte Netze, ein Symptom der Mode und Fetischisierung von hyperperforierten Oberflächenmustern."[119]

Bezugnehmend auf Bettums Kritik an der derzeitigen Auseinandersetzung mit Oberflächen und der architektonischen Praxis als reine Gestaltungsübung, ließe sich an dieser Stelle mit dem Text *„Von der Box zum Blob und wieder zurück"*, der jedoch bereits im Jahr 1999 verfasst worden war, wie folgt etwas Gegenteiles behaupten:
„Denn die freie Form bzw. der Blob ist heute nicht mehr Ausdruck von Spontaneität, sondern wird am Rechner generiert und ist damit als rein geometrische und damit rationalisierte Form zu betrachten, während die Vertreter der Box offenkundig nur noch an der Wirkung von Oberflächen und Materialien interessiert zu sein scheinen."[120]

Es sei jedoch bemerkt, dass ein rationalistischer bewusster Umgang mit Formen und Oberflächen selbstverständlich nicht zwingend Objekte und Räume erzeugt, die einen Beitrag zum aktuellen kulturellen, politischen Geschehen liefern. Der Ausweg zu einer sinnvollen Gestaltung von Räumen und Oberflächen führt nach Bettums Ansicht über den Einsatz der neuen präzisen digitalen Methoden zur Untersuchung der mikrostrukturellen Logik der verwendeten Materialien und deren Übersetzung in die Makroebene. Dadurch soll eine durchgehende Logik von Mikro- zu Makrostruktur herrschen und dadurch materialgerechte Raumstrukturen entworfen werden können.[121]

Sollte Bettums Text hier im Kern richtig wiedergegeben worden sein, dann kann meines Erachtens schlussgefolgert werden, dass auch Bettums Ansatz lediglich einen rationalisierten technischen Denkansatz darstellt, der keine Aussage über die kulturellen politischen Qualitäten der damit geplanten Objekte zulässt.

Rem Koolhaas

Auch Rem Koolhaas bemerkt in einem Interview ähnlich wie Picon, dass sich die aktuelle Architektur dem Wirtschafts-„Markt" und seinen Gesetzen unterwirft und als inhaltslose Hüllen nicht mehr als eine „Landmark" -wirkung zu bieten hätte. Auf die Aussage des Interviewpartners, dass die Avantgarde-ArchitektInnen aber sehr erfolgreich wären, kontert Koolhaas:

"That's fine. But Gehry, to focus on him for a moment, would never be asked about his political views. On the other hand, our firm AMO has worked for the European Union in Brussels. We tried to make the EU's political message clearer, more attractive and easier to communicate."[122]

So ist Rem Koolhaas als Mitglied im „Rat der Weisen", welche offiziell als „Reflexionsgruppe" bezeichnet wird, für die EU tätig und soll mit dieser Gruppe Zukunftsperspektiven mit einem Zeithorizont von 10 bis 20 Jahren in unterschiedlichsten Themengebieten für die Europäische Union erarbeiten.[123] In einem anderen sehr ausführlichen Interview drückt Koolhaas die Problematik folgend aus:
„Ich glaube eher, dass wir gerade den globalen Triumph des Exzentrischen erleben. Lauter extravagante Bauten entstehen, Bauten ohne Inhalt, ohne Funktionalität. Es geht ausschließlich um spektakuläre Formen und natürlich ums Ego der Architekten."[124]

Einen wesentlichen Beitrag zu dieser Entwicklung hätten die Medien geliefert, indem diese in Sensationsgier den *„Stararchitekten"* geschaffen hätten, der ausschließlich vermarktbare Icons und Wahrzeichen produzieren solle. Die zunehmend privaten BauherrInnen würden Architektur als Werbemaßnahme bzw. Renditenbringer betrachten und auch Städte und andere staatliche Institutionen investieren nicht in Kultur, sondern in Marketing und Effekte, eine mE offensichtliche Anspielung Koolhaas' auf den sogenannten „Bilbaoeffekt". Es gebe zwar auf Grund der Risikobereitschaft vieler UnternehmerInnen zurzeit mehr Freiheiten in der Architektur, paradoxerweise führe dies aber zu mehr Begrenzungen als früher. Hier bezieht sich Koolhaas unter anderem auf die Begnügung von manchen ArchitektInnen mit der Gestaltung von Gebäudehüllen, die in diesem Text bereits mit Picon oder insbesondere Bettum erwähnt wurden. Auf die Frage nach möglichen Alternativen betont Koolhaas, keine Antwort darauf geben zu können und verweist lediglich darauf, dass er sich im Zusammenhang mit seinen Arbeiten in jedem Fall für Inhalte interessiere.[125]
Koolhaas bekräftigt seine Ansicht über die Architektur, indem er auf die seines Erachtens seit der Aufklärung bestehende Gültigkeit folgender Fragen hinweist: *„Wie wollen wir leben? Welche Möglichkeiten eröffnet uns die Moderne mit ihren Freiheiten und dem technischen Fortschritt? Am Ende also: Was macht uns glücklich?*".[126]

Christopher Alexander

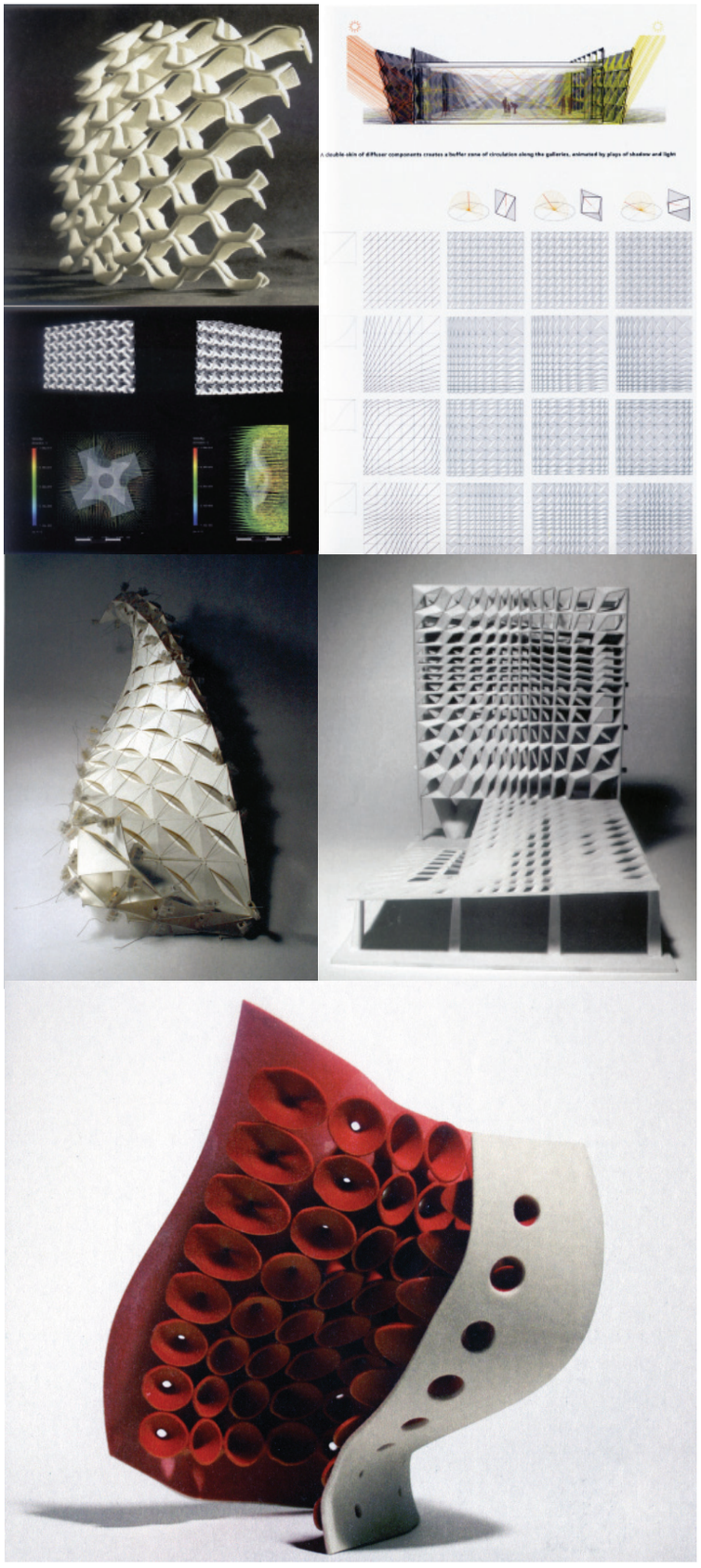
Christopher Alexander reagiert in einem Interview auf die neue extravagante Architektur und die von Koolhaas angesprochenen Betonung des mächtigen Egos der ArchitektInnen, indem er diesen Drang zum Neuen als einen Zwang aus Angstmotiven entlarvt und den gegenwärtigen KollegInnen eine sehr oberflächliche Auseinandersetzung mit dem Vergangen vorwirft.
„Einige Architekten haben Angst vor diesen [...] wie soll ich sie nennen tief verwurzelten Archetypen. Und weil sie solche Furcht davor haben, etwas zu entwerfen, das an etwas Vorhandenes erinnert, fühlen sie sich gezwungen bestimmte Dinge nicht zu tun, die ansonsten jeder vernünftige Mensch machen würde, einfach weil sie praktisch sind."[127]
Den Begriff der Morphogenese mittels biologischer Differenzierung in Verbindung mit der Computertechnik zu nennen, hält Alexander für problematisch. *„Mit einem Computer kann man zwar eine Reihe von Parametern definieren und endlos viele Kombinationen und Variationen am laufenden Band produzieren, aber wenn sie keine Bedeutung haben, dann ist das ganze nur ein banales Spiel. Die Vorstellung einer solchen Welt finde ich sogar beängstigend, weil sie nicht aus etwas Realem entsteht. Man kann ihre Unaufrichtigkeit geradezu herauslesen, denn sie propagiert eine falsche Art von Vielfalt."* So sei Morphogenese mit dem Computer nur schwer zu bewerkstelligen, da echte Variationen sich aus jeweiligen Situationen entwickeln, wobei diese Art von Differenzierung sich an allen an ihre Umgebung gut angepassten Gebäuden findet.[128]

Peter Zumthor

So bemerkt Peter Zumthor hinsichtlich des Verhältnisses des Entwurfskontextes zur Architektur und besonders zur Auseinandersetzung mit den örtlichen Gegebenheiten: *"Schöpft ein Entwurf allein aus dem Bestand und der Tradition, wiederholt er das, was sein Ort ihm vorgibt, fehlt mir die Auseinandersetzung mit der Welt, die Ausstrahlung des Zeitgenössischen. Erzählt ein Stück Architektur nur Weltläufiges und Visionäres, ohne ihren konkreten Ort zum Mitschwingen zu bringen, vermisse ich die sinnliche Verankerung des Bauwerks an seinem Ort, das spezifische Gewicht des Lokalen."*[129] Außerdem bemerkt Zumthor zur zeitgenössischen Trendarchitektur, dass häufig eine mangelnde Disziplin in der Ausführung und Detailplanung erkennbar wäre und Gebäude oft in einer viel zu kurzen Zeitdauer realisiert werden würden. Es ärgere Zumthor, wenn er an einem Gebäude erkennen könne, dass eine Knappheit im Terminplan offensichtlich zu „Schlampereien" und Oberflächlichkeiten geführt habe.[130]

Daniel Gethmann

Als Beispiel für einen Text[131], der sich im Gegensatz zu den hier bereits genannten weit verbreiteten und populären Kritiken, die in ihren Grundzügen meist die Oberflächigkeit bzw. Inhaltslosigkeit der gegenwärtigen Avantgarde betont und die sich in ähnlicher Form auch auf viele andere Phänomene in unsere Gesellschaft umlegen lässt, sei eine Auseinandersetzung gezeigt die noch eine Ebene tiefer gräbt und damit versucht eine konkrete Praxis als Ursache für ein bestimmtes Versagen ausfindig zu machen. Mittels der Rezeption eines Textes von Bruno Latour versucht Daniel Gethmann, einen möglichen Grund für die Unfähigkeit zur Selbstreflexion in der zeitgenössischen Architektur auszumachen. So seien die „Nonstandard Structures" aus ihrer eigenen Perspektive zwar aus dem Geist der Moderne entstanden, doch könne dies Gethmann zufolge mit Latour aus umgekehrter Sicht – nämlich aus dem Blickwinkel der Moderne - nicht bestätigt werden.
Der Begriff der Moderne beschreibt laut Latour im Wesentlichen zwei Praktiken. Als erstere Tätigkeit nennt Latour den Vorgang des „Übersetzens" und in weiterer Folge jenen des „Reinigens". Das „Übersetzen" als erste Praktik sei vom Bestreben zur Differenzierung bzw. der Aufteilung von Dingen in menschliche und kulturelle Objekte und einer damit verbundenen Begriffsfindung geprägt. In einem weiteren Schritt folgt das Reinigen bzw. Schärfen der Kanten der differenzierten Objekte mittels deren genaueren Definierung. Dies ermögliche einen klaren Blick und eine Reflektion und Einordnung der Objekte. Die gegenwärtigen Tendenzen kehren den ersteren Schritt um, indem sie verschiedenste Begriffe zu Neuen kombinieren und versuchen den Diskurs dennoch weiter laufen zu lassen. Dabei entstehen aus Sicht der Moderne unmögliche Begriffskombinationen, die neue Objekte als sogenannte „Hybride" gebären. Diese Hybride lassen sich im zweiten Schritt nicht mehr reinigen, also weder als kulturell noch als menschlich bestimmen und sind mit modernen Methoden in letzter Konsequenz auch nicht mehr begreif- bzw. reflektierbar. Aus dieser Sicht seien die aktuellen Tendenzen „nichtmodern", auch nicht „post-modern", höchstens „vormodern".
So zitiert Gethmann Latour wie folgt: *„Wenn man aber von Embryonen im Reagenzglas, Expertensysteme, digitalen Maschinen, Robotern mit Sensoren, hybriden Mais, Datenbanken, Drogen auf Rezept, Walen mit Funksendern,*



synthetisierten Genen, Einschaltmessgeräten, etc. überschwemmt wird, wenn unsere Tageszeitungen all diese Monstren seitenweise vor uns ausbreiten und wenn diese Chimären sich weder auf der Seite der Objekte noch auf der Seite der Subjekte, noch in der Mitte zu Hause fühlen, muss wohl oder übel etwas geschehen.[...] Wo soll man die Hybriden unterbringen? Sie sind unser Werk. Sind sie also menschlich? Aber sie sind nicht unser Tun. Sind sie also natürlich?".[132]

Die zeitgenössische Architektur werde von Mensch-Maschine Hybriden erzeugt und mit virtuellen Eigenschaften zu neuen Hybriden informiert, womit wir diese nicht mehr klar einordnen könnten. Es gäbe es also keine Schublade, keine Haltung, keine Ideologie, in der man diese Hybriden-Objekte ablegen könnte, denn es existieren nur Schubladen aus der Moderne. So schlägt Gethman mit Latour vor, beispielsweise zunächst die aus der Moderne stammende Praxis der Differenzierung zwischen „Nonstandard" und „Standard" aufzugeben, womit man zunächst gedanklich im handwerklichen vorindustriellen Zeitalter lande. Damals gab es nur individuelle Produkte und damit keinen Standard. Übertragen in ein Zeitalter des digitalen Handwerks spielt diese Differenzierung ebenfalls keine Rolle mehr, da es für die maschinelle Fertigung keinen Unterschied macht, ob ein Objekt als Kopie vervielfältigt wird oder ein Unikat hergestellt werden soll.[133]

Zusammenfassend

Um die Wende vom 20. zum 21. Jahrhundert waren populäre Themen in der Architekturdebatte vorwiegend über den Einsatz von Animationsprogrammen im Entwurf, das vermehrte Auftauchen von „Blobs" und die zunehmende globale Vernetzung via Internet beeinflusst. Dies regte beispielsweise zu Überlegungen bezüglich eines neuen Raumzeitverständnisses, des Zusammenhangs zwischen Form und Bewegung, der Generierung und Konstruktion von Freiformen oder auch zu Diskursanalysen der geschichtlichen bzw. stilistischen Verbindung zwischen „Box" und „Blob" an. Häufig wurden Projekte der Kritik einer beliebigen Formgebung, resultierend aus dem ziellosen Experimentieren mit den neuen digitalen Methoden, einer mangelnden Funktionalität und der verhältnismäßig aufwendigen Umsetzbarkeit ausgesetzt. Eine Dekade später richtet sich der Fokus der Diskussionen nach dem vermehrten Einsatz digitaler Werkzeuge und der Realisierung einiger Projekte mittels industrieller Fertigungstechniken, auf die mögliche Digitalisierung bzw. Automatisierung des architektonischen Entwurfsprozesses und den damit verbunden Chancen, die sich daraus für die nahe Zukunft ergeben könnten. Der Grundtenor ist, in einer Zeit der „Krisen" wie jenen der globalen Erwärmung oder der Wirtschaft und den damit verbundenen oft genannten Begriffen wie etwa der „Nachhaltigkeit" oder maximalen „Effizienz" die Umsetzung einer performativen Architektur mittels neuer Technologien. Dementsprechend wird beispielsweise eine kostengünstige Produktion von individuellen Objekten für die breite Masse, die Herstellung von intelligenten bzw. leistungsoptimierten Bauteilen mittels

Abbildung 50: An den Masterarbeiten der Architectural Assosiation London kann man die starke Fokussierung auf leistungsfähige Architektur bzw. Fassaden deutlich erkennen.

digitaler Analyse-, Generierungs- bzw. Optimierungsalgorithmen und dem Einsatz digitaler Fertigungsmethoden forciert. Gleichzeitig ist die in diesem Kontext entstehende Architektur der Kritik ausgesetzt, sich aufgrund eines mangelnden Intellekts zu sehr an den Gesetzen der Markwirtschaft zu orientieren, sich widerstandslos den gegenwärtigen Ordnungen zu unterwerfen und damit keinen bzw. einen geringen Beitrag zur kulturellen bzw. politischen Entwicklung der Gesellschaft zu liefern. Insofern würden die neuen Werkzeuge für die Erweiterung der bestehenden Machstrukturen eingesetzt, wobei sich die aktuelle Argumentation tendenziell mit den Worten *„Ich tue es, weil ich es kann."*[134] pointieren lassen kann. So schlussfolgern Hensel und Menges bezogen auf die Debatten der 1970er Jahre wie folgt:

„Damals wurde der Mythos von Objektivität, Neutralität und Allgemeingültigkeit methodisch hinterfragt – heute könnte man auf ähnliche Art und Weise den Mythos von Individualität, Selbstorganisation und die Hoffnung auf marktförmig organisierte Prozesse hinterfragen."[135]

68 Patrik Schumacher, Arch +, Zeitschrift für Architektur und Städtebau, Ausgabe 195, Nov. 2009, S.106-113. Original Text: : "Parametricism - A New Global Style for Architecture and Urban Design", erschienen in der Zeitschrift Architectural Design, Digital Cities, Ausgabe 79, Julie/August 2009, Editoren Neil Leach und Hellen Castle.

69 -Vgl. Patrik Schumacher, Arch +, Zeitschrift für Architektur und Städtebau, Ausgabe 195, Nov. 2009, S.11.

70 Patrik Schumacher, "Latente Utopien", The Autopoesis of Architecture, Zaha Hadid/Patrik Schumacher, steirischer Herbst 2002, S.17.

71 Niklas Luhmann, "Soziale Systeme, Grundriss einer allgemeinen Theorie", Suhrkamp Verlag, Erste Auflage 1987, S.592.

72 Patrik Schumacher, "Latente Utopien", The Autopoesis of Architecture, Zaha Hadid/Patrik Schumacher, steirischer Herbst 2002, S.14.

73 Ebd., S.17.

74 Patrik Schumacher, Arch +, Zeitschrift für Architektur und Städtebau, Ausgabe 195, Nov. 2009, S.108.

75 Bernd Cache, "After Parametrics", Grazer Architektur Magazin, Ausgabe 06, 2009. S.51-61.

76 Ebd. S.52.

77 Ebd., S.56

78 Ebd., S.53.

79 Jörg H. Gleiter, "Zur Genealogie des neuen Ornaments im digitalen Zeitalter", Arch+, Ausgabe 189, 2008, S.79.

80 Ebd., S.80.

81 Michael Müller, "Die Verdrängung des Ornaments – Zum Verhältnis von Architektur und Lebenspraxis", Suhrkamp Verlag, Erste Auflage 1977.

82 Jörg H. Gleiter, "Zur Genealogie des neuen Ornaments...", a. a. O., S.83.

83 Ingraham, Hays, Kennedy, Artikel, Computeranimismus, 1995, "architektur_theorie.doc", 2003, Hrsg. Gerd de Bruyn, Stephan Trüby, Birkhäuser Verlag, S.149.

84 Vgl. Fabian Scheurer, Was da gefordert wird sind Kathedralen zum Nulltarif, Grazer Architektur Magazin, Ausgabe 06, 2009. S.214.

85 Vgl. Urs Hirschberg, "Augmented Architecture", Kulturtechnik Entwerfen, a. a. O., S.306.

86 Vgl. Oliver Fritz, "CAM of Freeforms in Architecture", Hg. Institut für Raumplanung der Hochschule Liechtenstein, Michael Imhof Verlag GmbH & Co. KG, S.37.

87 Beispielsweise: http://www.tailorstore.de/, http://www.zazzle.de/, http://nikeid.nike.com/, http://www.lbs-city.de/ (Stand 24.03.2010).

88 Fabian Scheurer, "Was da gefordert wird sind Kathedralen zum Nulltarif", a. a. O., S.208.

89 Ebd., S.213.

90 Vgl. Bonwetsch, Gramazio, Kohler, "Digitales Handwerk", Grazer Architektur Magazin, Ausgabe 06, 2009, S.175.

91 Ebd., S.177.

92 Vlg. Gramazio, Kohler, "Digital Materiality in Architecture", Lars Müller Verlag, 2008, S.7. 87

93 Vgl. Kostas Terzidis, "Algorithmic Architecture", Architectural Press is an imprint of Elsevier, First Edition 2006, Prologue S.12.

94 Vgl. Ingeborg Rocker, "Re-coded: Studio Rocker", Aedes Berlin, Juni 2005, Introduction.

95 Claus Dreyer, "Semiotik und Ästhetik in der Architekturtheorie der sechziger Jahre", Kulturtechnik Entwerfen, a. a. O., S.200.

96 Vgl. Kostas Terzidis, Algorithmic Architecture, a. a. O., S.56.

97 Antoine Picon, Das Projekt, Arch+, Ausgabe 189, 2008. S.17.

98 Vgl A. Sutcliffe and N. Mehandjiev, "End-user development: tools that empower users to create their own software solutions," Communications of the ACM, vol. 47(9), 2004, S. 31-32.

99 Vgl. D. Benslimane, S. Dustdar, and A. Sheth, "Service Mashups: The New Generation of Web Applications," IEEE Internet Compting, vol. 12(5), 2008, S. 13-15.

100 N. Seyff and F. Graf, "User-Driven Requirements Engineering for Mobile Social Software," Proceedings of 3rd. International Workshop on Social Software Engineering, 24. Feb. 2010 Paderborn, Germany.

101 Gespräch mit Dipl. Ing. Florian Graf und Dipl. Ing. Dr. Norbert Seyff, City University of London am 23.02.2010.
Bemerkung: Graf und Seyff forschen an Endbenutzerzentrierten Anforderungserhebungsmethoden und entwickelten im Rahmen ihrer Forschung den werkzeugunterstützen iRequire Ansatz.

102 Vgl. Christian Kühn, "Christopher Alexanders Pattern Language", Arch +, Ausgabe 189, 2008. S.27.

103 Michael Meredith, From Control to Design, Editiert von Sakamoto u. Ferré, Verlag Actar-D 2008, S.6.

104 Vgl. MVRDV, The Functionmixer, "Latent Utopias", a. a. O., S.174-179.

105 Oliver Fritz, "Programmiertes Entwerfen", Arch +, Ausgabe 189, 2008, S.61-62.

106 Vgl. Ebd. S.60 oder Georg Vrachliotis, "Generatives Design", Arch +, Ausgabe 189, 2008, S.59.

107 Oliver Fritz, "Programmiertes Entwerfen", Arch +, Ausgabe 189, 2008, S.61.

108 Vgl. Georg Vrachliotis, "Generatives Design", Arch +, Ausgabe 189, 2008, S.56-57 oder Oliver Fritz, Programmiertes Entwerfen, a. a. O.,, S.63.

109 Achim Menges, "Unkomplzierte Komplexität", Grazer Architektur Magazin, Ausgabe 06, 2009, S.144.

110 Arch +, Ausgabe 188, Juli 2008, S.25.

109 Antoine Picon, Das Projekt, Arch+, Ausgabe 189, 2008. S.17.

110 Vgl. ebd. S.17.

111 Vgl. ebd. S.17.

112 Vgl. ebd. S12-17.

113 Peter Eisenmann, Interview von Urban Center Books, About Books and book collecting, YouTube Video, http://www.youtube.com/watch?v=JvdDQzT56ks (03.04.2010).

114 Interview mit Peter Eisenmann von Robert Locke am 27.07.2004, http://archinect.com/features/article.php?id=4618_0_23_0_M (02.04.2010).

115 Ebd.

116 Ebd.

117 Peter Eisenman, Vortag an der Vanderbilt University http://www.youtube.com/watch?v=AJMMnb0qrXA&NR=1, (Stand 05.04.2010).

118 Ebd.

119 Johan Bettum, Architectural Form and Saturated Space, Grazer Architektur Magazin, Ausgabe 06, 2009, S.86-103.

120 Nikolaus Kuhnert, Angelika Schnell, "Von der Box zum Blob und wieder zurück", Arch +, Ausgabe 148, 1999, S.20.

121 Johan Bettum, Architectural Form and Saturated, a. a. O., S.86-103.

122 Interview mit Rem Koolhaas am 27.03.2006, Spiegel Online International, http://www.spiegel.de/international/spiegel/0,1518,408748,00.html (02.04.2010).

123 Vgl. Europa Journal, Ö1, Nachdenken über die Zukunft Europas, Sendung am 14.08.2009, oder http://oe1.orf.at/highlights/141799.html oder http://www.euractiv.com/de/prioritaten/eu-rat-weisen-erwgt-ausschluss-ffentlichkeit-arbeiten/article-178725.

124 Interview mit Rem Koolhaas am 12.06.2008, Zeit Online, http://www.zeit.de/2008/24, Koolhaas-Interview (02.04.2009).

125 Vgl. ebd.

126 Ebd.

127 Interview mit Christopher Alexander von Koolhaas und Orist, Von fließender Systematik und generativen Prozessen, Arch+, Ausgabe 189, 2008, S.24.

128 Vgl. ebd., S.24-25.

129 Peter Zumthor, Zitiert nach http://irge-uni-stuttgart.de/lehre/themenreihe/identitaet.html, (03.04.2010).

130 Vgl. Peter Zumthor in einem Interview am 14.06.2008 mit dem Schweizer Radiosender DRS2, Audiodatei erhältlich bei http://www.drs2.ch/www/de/drs2/sendungen/focus/2655.sh10040734.html (Stand 03.04.2010).

131 Daniel Gethmann, "Nichtmoderne Objekte", Grazer Architektur Magazin, Ausgabe 06, 2009,, S.43-49.

132 Ebd., S. 44-45.

133 Vgl. ebd., S.43-49.

134 Mario Carpo, The Digital „Mouvance", and the End of History, Grazer Architektur Magazin, Ausgabe 06, 2009,, S.22.

135 Michael Hensel, Achim Menges, Am Anfang einer neuen Architektur des Performativen, Arch+, Ausgabe 188, 2008, S.16.

Allgemeines zu Brücken

Überlagerung von Zeit, Technik und Kultur

Seit Jahrtausenden prägen Brücken das Erscheinungsbild von verschiedensten Orten und Landschaften entscheidend mit. Die Gestalt einer Brücke wird im Kern wesentlich durch den Umgang mit den an und in der Konstruktion auftretenden Kräften, den verwendeten Materialien und der Wegführung über die gesamte Länge bestimmt. Trotz alltäglicher Präsenz und der mehrfachen Nutzung durch Menschen, Autos, Züge, Flugzeuge oder auch Boote erregen die meisten Brücken als Verbindungswege zwischen Orten als reines Mittel zum Zweck in der Regel wenig Aufmerksamkeit. *„Im Gegensatz zu einem Wohnhaus, einem Bürogebäude, einer Konzerthalle oder einem Krankenhaus müssen Brücken keine umfassende, dauerhafte Beziehung mit menschlichen Tätigkeiten, Bedürfnissen oder Dienstleistungen aufbauen; die Menschen leben oder arbeiten nicht auf Brücken."*[136] Demzufolge hinterlässt eine große Anzahl von Brücken bei ihren BenutzerInnen meist einen sehr vergänglichen Eindruck. Doch ermöglichen Brücken mehr als den Fluss von unterschiedlichsten Bewegungen und dokumentieren neben dem aus Kräftespiel resultierendem Tragwerk mehr als nur den technischen Fortschritt der Geschichte. Als gebaute Symbole, Icons, Wahrzeichen oder Denkmäler verkörpern Brücken über eine bewusste Wahl von Konstruktionen und den Einsatz von bestimmten Materialien und ihrer Fähigkeit zur Verbindung von Gesellschaften, Kulturen und Ideologien den Willen, die Politik und die Kultur einer Gesellschaft. So schreiben Ursula Baus und Mike Schlaich in diesem Kontext:

„Denn manche Konstruktionstypen sind das Ergebnis technologischer, wissenschaftlicher Entwicklungen, die an bestimmte Zeiten gebunden sind; manche gestalterischen Ansätze gehören daneben zu Epochen, die zu einem bestimmten formalen Ausdruck gefunden haben. Mal stachelt das Ausreizen der Leichtigkeit die Ingenieure an, mal begreifen Architekten den Typus als homöopathisch wirksames Mittel zur Reparatur der geschundenen Stadt, mal verliert sich das technische Artefakt im ästhetischen Ganzen einer arkadischen Landschaft."[137]

So werden Brücken laut Baus und Schleich beispielsweise nach der Eroberung der Städte durch das Automobil und deren damit verbundenen Zerstückelung sowie der Zerschneidung von Landschaften durch mehrspurige Straßen zur Überwindung dieser Hindernisse und damit zur Reparatur dieser Orte eingesetzt. Weiters können durch Flüsse, Täler oder Meere getrennte Orte oder Wohn-, Arbeits- bzw. Industriegebiete in Städten mittels Brücken miteinander verknüpft und dadurch deren Integration in einer Gesamtstruktur verbessert werden.[138] Auch die Auswirkungen wirtschaftlicher Phänomene können sich auf die Wahl von Brückenkonstruktionen auswirken. Mit Beginn der „Ölkrise" im Jahr 1973 wurden Begriffe wie „Umweltverantwortung" und „Effizienz" zu populären politischen Zielen, wodurch der Leichtbau als besonders ressourcensparende Konstruktion bevorzugt wurde. Als Vorbilder für diese sogenannten „ehrlichen" und „materialgerechten" Konstruktionen dienten beispielsweise Pionierarbeiten wie die von Buckminster Fuller (1895-1983). So waren etwa Fullers Arbeiten Inspiration für Frei Ottos filigrane Bauten wie etwa das Dach der Münchner Olympiastätten.[139] Als Beispiel für die Verwüstung eines politisch sozialen Symbols nennt Dirk Bühler die Zerstörung der alten „Brücke von Mostar", die als Zeichen des Zusammenlebens verschiedener Volksgruppen im Bosnienkrieg bewusst bombardiert wurde. Damit sei der Beschuss des muslimischen Stadtteils durch die Kroaten und der Zusammenbruch der Brücke am 9. November 1993 nicht bloß als militärischer Eingriff, sondern als bewusster symbolischer Akt, der die Spaltung der Bevölkerung vertiefen sollte, zu deuten.[140] Auch im Sprachgebrauch lässt sich in der Verwendung des Wortes wie etwa „überbrücken" im Sinne von „überwinden", „verbinden" etc. die kulturelle Verankerung und symbolische Deutung von Brücken erkennen.

Bezugnehmend auf die vorliegende Arbeit sei erwähnt, dass seit der Jahrtausendwende nach dem Vorbild von Weltausstellungen vermehrt Milleniumparks, -towers und auch -bridges gebaut werden. Dabei wird häufig auf kleinere multifunktionale Objekte wie Fußgängerbrücken mit „Icon-Wirkung" bzw. den damit in Verbindung stehenden „Bilbaoeffekt" gesetzt. Besonders kleinere Brücken werden in urbanen Gebieten und Parks zu Orten der Begegnung, die neben ihrer Wahrzeichenwirkung von den Menschen auch als Joggingwege, Boulevards, Promenaden, Orten für Rendez-vous etc. adaptiert werden. Ein Meister dieser zeichenhaften extravaganten Skulpturen ist der seit den 1970er Jahren besonders erfolgreiche Architekt und Ingenieur Santiago Calatrava, der vorzugsweise den „hohen Bogen" in verschiedensten Ausführungen vorwiegend gekippt, verdreht oder verzogen einsetzt. Weitere gegenwärtig bekannte aktive BrückenentwerferInnen in diesem Bereich sind beispielsweise Ney & Partners in Belgien, Feichtinger Architekten oder Marc Mimran in Frankreich, Schlaich Bermann und Partner oder Leonhardt-Anrä und Partner in Deutschland, Juan Arenas in Spanien oder Wilkinson Eyre, Buro Happold oder Knight Architects in Großbritannien.[141]

Mit diesen Beispielen sei nun verdeutlicht, dass Brücken in verschiedenster Hinsicht ein Spiegel bzw. Ausdruck einer bestimmten Zeit, eines Ortes und einer Kultur sind. „Gute Brücken sparen Ressourcen, sind also ökologisch, schaffen Arbeit, sind also sozial und am Ende sind sie auch noch auf natürliche Weise schön. Ökologisch, sozial, kulturell, was könnte zeitgemäßer sein?"[142]

Ingenieure und Architekten – Bridge Designer

Die geschichtliche Entstehung des Ingenieurwesens und dessen Abspaltung von der Architektur wird üblicherweise mit der „Industrialisierung" gegen Ende des 18. Jahrhunderts in Verbindung gebracht. Anfangs steht dabei oft die im Jahre 1747 in Frankreich gegründete „Zivilingenieurschule" oder auch die später folgende polytechnische Schule sowie die Schule für Straßen- und Brückenbau „École Nationale des Ponts et Chaussées"

mit ihren Bestrebungen, wissenschaftlich und konstruktiv zu bauen, im Vordergrund.[143] Die geschichtliche Entwicklung wird in dieser Arbeit nicht weiter behandelt, da der Fokus auf der gegenwärtigen Rollenverteilung von IngenieurIn und ArchitektIn bei aktuellen Brückenentwurfsaufgaben liegt. Daher werden die Entstehungsumstände hier abschließend mit den Worten des erfolgreichen Brückenbauers und Ingenieurs Jörg Schlaich pointiert. Schlaich verwies in einem Interview auf die Frage nach der Geburtsstunde des Ingenieurs auf die im Jahre 1889 abgehaltene Weltausstellung in Paris:

„Die Fußgelenke dieser riesigen Hallen stellten das Denken und bisherige Verständnis der Baumeister auf den Kopf, weil sie dort den Querschnitt einschnüren, wo die Kräfte am größten sind: am Fuß, knapp über dem Boden, dort, wo das Bauwerk doch am dicksten sein müsste, wie uns die Natur lehrt! Damit war denen die Bauen ohne Statik gelernt und höchstens ein Gefühl für, aber keine Erfahrung mit Kraftfluss haben, den Architekten also, der Boden entzogen. Sie überließen schmollend solche Konstruktionen den Ingenieuren und dekorierten deren filigrane Bahnhofshallen und Brücken mit prunkvollen Fassaden und bombastischen Türmen."[144]

Wie bereits erwähnt, ist die Gestalt von Brücken im Allgemeinen wesentlich durch die „an" und „in" der Konstruktion auftretenden Kräfte und den Umgang mit den daraus resultierenden Spannungen in der Konstruktion geprägt. So wird bei der Ableitung der Kräfte in die Auflager besonders darauf geachtet, dass sich die Brückenform möglichst mit dem Kräftefluss deckt, wodurch die Konstruktion optimal ausgenützt werden soll und folglich kein unnötiger Materialverbrauch entsteht. Wird die Brückenform nicht aus ihrem Tragverhalten entwickelt und ist der Kraftfluss an ihrer Gestalt nicht ablesbar, ist diese nach Schlaich nicht wahrhaftig, sondern unehrlich.[145]
In diesem Kontext gibt es unzählige Kommentare wie beispielsweise jenen von Michel Virlogeux:

„...selecting shapes for pure fantasy and appearance without any structural logic, and then trying to amend the design to balance loads and effects is a pure nonsense."[146]

Nach weit verbreiteter Ansicht deckt sich die Form einer optimierten Tragkonstruktion mit einer ästhetisch gelungenen Brückengestalt. Demgemäß äußerte sich der berühmte Bauingenieur und Betonbauexperte Robert Maillart (1872-1940) wie folgt:

„Ein gut konstruiertes Tragwerk ist auch schön."[147]

So hätte nach Schlaich auch Maillart weniger Wert auf Ästhetik gelegt, sondern setzte vorrangig auf die Wirtschaftlichkeit der Konstruktion, woraus eine schöne Gestalt automatisch folge. So bemerkt Schlaich: *„Dass ein konstruktiv falsch angelegtes Bauwerk ästhetisch gut ist, kann ich mir kaum vorstellen, wobei wie immer Ausnahmen die Regel bestätigen, wie das Sydney Opera House von Jorn Utzon."*[148]

Bezugnehmend auf die vorliegende Arbeit ist darauf hinzuweisen, dass sich Schlaich zu den gegenwärtigen Entwicklungen, den digitalen Methoden und den damit verbundenen formalen Freiheiten und zur Ästhetik äußerst kritisch wie folgt äußerte:

„Wenn also zukünftig dank dem Computer ‚alles möglich' ist, muss man sich um beispielsweise in einem Wettbewerb aufzufallen, künstlich etwas einfallen lassen, etwas draufsetzen [...]. So übertreffen sich ‚die Wilden' mit aufgeblasenen und zerknautschten Blobs, oft Persiflagen und Karikaturen missinterpretierter ‚Vorbilder' – ich weiß aus unserer spannenden Zusammenarbeit mit Frank O. Gehry, wovon ich rede, bar jeden Bezugs zwischen Form, Funktion und Kraftfluss. Hoffentlich eine schnell vergängliche Mode der Spaßgesellschaft."[149]

Um eine möglichst effiziente Tragwerkskonstruktion entwickeln bzw. die verwendeten Materialen optimal nutzen zu können, sind fundierte Kenntnisse in der Statik, Mechanik und Materialkunde erforderlich. Schlaich behauptet, dass es aufgrund der engen Verknüpfung mit dem Tragverhalten eher dem/der BauingenieurIn mit einer Zusatzausbildung in Baukultur sowie auch im Entwerfen möglich wäre, eine gelungene Brücke zu konzipieren, als einem/einer Architekten/in mit einer Grundausbildung in der Tragwerkslehre. So würden *„Architekten zwar in der Tragwerkslehre mit Grundkenntnissen versorgt, doch daraus erwächst nur selten eine Tragwerksentwurfskompetenz."*[150]
Aus diesem Grund schrieb Schlaich über den Brückenentwurf als „Königsdisziplin" der Ingenieure/Innen: *„Dieser gesellschaftspolitische und kulturelle Hintergrund verpflichtet die Bauingenieure, über ihr Wissen und Können hinaus ihre Phantasie einzusetzen, um ihre Bauten, insbesondere in ihrer Königsdisziplin, dem Brückenbau, so zu gestalten, dass sie die Natur, die sie verbauen (müssen), mit der einzigen adäquaten Gegenleistung entschädigen: mit Baukultur."*[151]
Solche Aussagen enthalten naturgemäß Konfliktpotential. Meines Erachtens wird bewusst der Begriff „Baukultur" und nicht „Baukunst" verwendet, da letzterer Begriff aus einer technisch funktionalen Perspektive höchstwahrscheinlich zu homöopathisch klingt. ArchitektInnen könnten im Gegenzug behaupten, dass der/die BauingenieurIn aufgrund seiner/ihrer mangelnden Ausbildung in Kultur- oder Kunstgeschichte oder anderen gestaltungsrelevanten Themen und auch der fehlenden Übung im Entwerfen die Qualitäten des Entwurfskontexts nicht lesen, beurteilen bzw. in ein entsprechendes Konzept umsetzen können. Für einen/eine technisch versierten/e BauingenieurIn ist es höchstwahrscheinlich schwierig, ein abstraktes Gesamtkonzept zu entwerfen und beispielsweise unterschiedlichste örtliche Gegebenheiten zu beurteilen und die Atmosphäre mit dem notwendige Vokabular bzw. Entwurfswerkzeug in eine adäquate Sprache von Proportion, Rhythmus, Anordnung, Materialien, Oberflächen, Lichtkonzepten etc. zu übersetzen.

Nach Virlogeux, Baus, Schlaich u.a. können ArchitektInnen jedoch viel zum Brückenentwurf beitragen und dem/der IngenieurIn neben der Beschäftigung mit dem Ort, Licht und Schatten und den Oberflächenwirkungen beim Ausdrücken und Formulieren der gewählten Strukturen helfen und elegante Details entwerfen. Auch Michael Blaschko thematisierte das Spannungsfeld

zwischen ArchitektInnen und BauingenieurInnen und merkt dazu an, dass sich eine Rivalität nicht leugnen lässt:
„Bezeichnen die einen die anderen gerne mit Technokraten, so charakterisieren die anderen die einen schon mal mit Phantasten"[152]. Über die Vorteile der Ausbildung in beiden Fachgebieten und deren Nutzung für den Entwurfsprozess antwortet der Stararchitekt Santiago Calatrava in einem Interview:
"I think certainly [...] the knowledge of materials [...] of mechanism [...] of mechanics helps you a lot in this process, because it becomes part of yourself and your thoughts and you can make natural references and at the same time implement them in terms of mechanical processes [...] it's so interesting [...] to be well trained as an engineer [...] being an architect. [...] The building itself want to be an expression of this what you can do with the language of an engineering to the service of an architecture"[153].
Um die angesprochene Qualität und Expression in einer Zusammenarbeit von ArchitektIn und BauingenieurIn zu erreichen, ist es allerdings zwingend, dass der/die ArchitektIn von Anfang an in den Entwurfsprozess integriert ist.[154] Der Bauingenieur Fritz Leonhardt (1909-1999) war ein berühmter Brückenbauer und arbeitete in vielen Projekten mit unterschiedlichen Architekten zusammen, wobei er sich vor allem in ästhetischen Formulierungen wie bezüglich günstiger Proportionen, Ordnungen und der Ausformulierung von Details beraten ließ. Zur Zusammenarbeit bemerkt Leonhardt folgendes:
„Der Ingenieur entwirft die Brücken, und der Architekt hilft als künstlerischer Berater. [...] Die künstlerische Beratung durch geeignete Architekten sollte auch in Zukunft für das Entwerfen von Brücken gepflegt, ja sogar gefordert werden."[155] Je größer der Umfang des Brückenentwurfes, desto eher ergibt sich über die steigende Anzahl von technischen Fragen bezüglich der Machbarkeit in der Regel eine klare Kompetenzverteilung zu Gunsten des/der Ingenieurs/Ingenieurin. So wird zum Beispiel bei der 2460 Meter langen und 343 Meter hohen Schrägseilbrücke „Viaduc de Millau" üblicherweise der Ingenieur Michel Virlogeux als Entwerfer und der Stararchitekt Norman Foster als Gestalter genannt[156]. Laut Virlogeux hat der/die IngenieurIn bei jedem Brückenentwurf generell die Hauptverantwortung zutragen; insbesondere im Bereich der Finanzen und der Sicherheit.[157] Ferner schreibt Virlogeux:
"The Designer – who must be an engineer – has the responsibility for the bridge design, or at least for the conceptual and detailed design which is the basis of the tender."[158]

Aus diesen Gründen kommt es besonders bei kleineren Projekten - dem Einfamilienhaus vergleichbaren Bautypus „Fußgängerbrücke" - zu Konflikten zwischen ArchitektInnen und IngenieurInnen, sodass die viel beschworene Zusammenarbeit auf die Bewährungsprobe gestellt wird.[159] Gegenwärtig wird die Zusammenarbeit besonders seitens der Politik immer häufiger gefordert. Wie bereits erwähnt, werden vermehrt Prestigeobjekte gewünscht, die als Wahrzeichen und Touristenattraktoren wirken sollen. Solche Entwürfe traut man eher ArchitektInnen als BauingenieurInnen zu. In der Öffentlichkeit bringt man Ingenieurbrücken vor allem mit den oft misslungenen Autobahn- bzw. Eisenbahnbrücken, die im Zuge der rasch vorangetriebenen Infrastrukturerweiterungen der 1960er Jahre in westlichen Staaten in großer Anzahl gebaut wurden, in Verbindung.

Als einen naheliegenden Grund für die Zusammenarbeit beider Disziplinen schrieb der Architekt Kurt Ackermann folgendes: *„Außerdem bin ich noch heute der Überzeugung, dass Entwicklungen und Erneuerungen in der Architektur vor allem durch neue Technologien, neue Baustoffe und neue Konstruktionen ermöglicht werden; [...]"*[160].
Um das gegenseitige Rollenverständnis zu fördern, die Kommunikation zu verbessern und dadurch schließlich auch später in der Praxis in einer Partnerschaft qualitativ hochwertige Projekte entwickeln zu können, fordert Ackermann bereits im Studium die gemeinsame Arbeit an Projekten für Studierende der Architektur und des konstruktiven Ingenieurbaus.[161] Dabei beschreibt Hans Kammerer die Annäherung seitens der BauingenieurstudentInnen an die Architektur folgend:
„Auch die Auflockerung der strengen Zucht des Grundstudiums der Unterstufe reizte manchen; der Architekt als gelegentlicher Alleinunterhalter schien anziehend. [...] Stufe um Stufe gewann dieses zunächst unverbindlich-naive Bild schärfere Konturen."[162]. Ackermann stellte allerdings mit Bedauern fest, dass es den Fakultäten für Bauingenieurwesen und der Architekturfakultät nicht gelungen ist, gemeinsame Lehrveranstaltungen wie *„Tragwerkslehre, Baustoffkunde, Baubetrieb, Bauphysik, Teile des Baurechts, Haustechnik oder auch Teile der Baugeschichte"* anzubieten.[163]
Hiezu sei zu erwähnen, dass es an der Technischen Universität Graz seitens des Instituts für Tragwerksentwurf den Versuch gibt, zusammen mit dem Institut für Betonbau einer Lehrveranstaltung anzubieten, in welcher Architektur- und BauingenieurstudentInnen zusammen an einem Entwurf arbeiten. Als Aufgabenstellung dient dabei seit ein einigen Jahren die „Concrete Student Trophy", die jährlich von der Vereinigung der Österreichischen Zementindustrie ausgeschrieben wird.

Zusammenfassend wird festgestellt, dass die Rolle und Verantwortung von ArchitektInnen oder IngenieurInnen im Brückenentwurf ein heikles Thema bleibt, da diese nur tendenziell und nicht eindeutig definiert werden kann. Auch Baus und Schlaich weisen diplomatisch darauf hin, dass die Bewertung der einzelnen Leistungen höchst individuell je nach Projekt und am finalen Ergebnis orientiert vorgenommen werden muss.[164] Um diesem Kategorisierungskonflikt zu entkommen, bezeichnete sich bereits Fritz Leonhardt selbst weder als „Bauingenieur" noch als „Architekt", sondern als "Baumeister. In diesem Sinn sei abschließend die von Sir Ove Nyquist Arup (1895-1988) versuchte professionsunabhängige Definition des "Brücken-Architekten" bzw. „Brücken-Designers" genannt:

„bridge architects, or bridge designers can be considered those professional who have been designing bridges with a specific concern for aesthetics for at least ten continuous years, or that have been involved in at least one hundred projects over time or that have been involved in the actual construction of ten bridges."[165]

Bezugnehmend auf die vorliegende Arbeit, die sich im Hauptteil wesentlich mit der Bemessung eines Tragwerkes auseinander setzt, soll für diejenigen, die in der Tragwerkslehre nicht bewandert sind, skizziert werden, welche Kenntnisse bzw. Grundprinzipien für die Bemessung eines Kräftesystems unerlässlich sind. Ohne das Verständnis dieser Zusammenhänge kann die „Scripted Bridge" im Kern nicht ernsthaft verstanden werden. Im Folgenden werden die Bemessungsschritte verallgemeinert dargestellt.[166] Die nachstehenden Beispielgrafiken beziehen sich auf das vereinfachte Tragsystem der „Scripted Bridge" - ein Kragarmsystem.

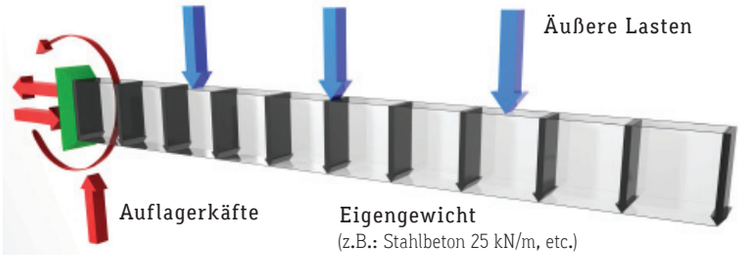


Abbildung 51: Kragarm mit den auftretenden Lasten. Die „Scripted Bridge" wurde auch nach diesem System berechnet.

Bemessung eines Tragwerks

Schritt 1: Idealisierung der Wirklichkeit - Modellbildung

In einem ersten Schritt muss der überlegte Entwurf in ein statisches System abstrahiert werden. Dazu müssen zunächst die Auflagerrandbedingungen definiert sowie Einwirkungen determiniert werden.

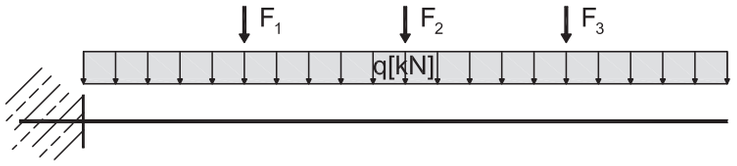


Abbildung 52: Statisches System. Kragarm mit drei Einzellasten und dem Eigengewicht als Gleichlast.

Diese Einwirkungen (auch äußere Kräfte genannt) werden durch ständige (Eigengewicht) oder veränderliche Lasten wie beispielsweise Nutzlasten (Personen, Autos, etc.) Wind, Schnee etc. verursacht. Lasten werden in Kilonewton pro Volumen [kN/m^2], Fläche [kN/m^2], pro Länge [kN/m] oder in einem einzelnen Punkt wirkend [kN] angegeben.

Die äußeren Kräfte müssen über die Auflager in den Baugrund abgeleitet werden können. Prinzipiell kann sich ein Tragwerk im Raum in x-, y-, z-Richtung verschieben sowie um selbige Richtungen verdrehen (Freiheitsgrade). Je nach Anzahl und Art der gesperrten Freiheitsgrade spricht man auch von der Wertigkeit eines Lagers, wobei zwischen Gleitlagern, Festlagern und Einspannungen unterschieden wird.

Schritt 2: Auflagerreaktionen berechnen

Allen angreifenden Lasten (Aktionskräften) muss durch die Auflagerreaktionskräfte insofern das Gleichgewicht gehalten werden, dass sich das Tragwerk weder verschieben noch verdrehen darf. Nach diesem Prinzip werden die Gleichgewichtsbedingungen (Gleichungen) aufgestellt und die unbekannten Auflagereaktionen berechnet.

Schritt 3: Schnittkraftermittlung

Allein die Bestimmung der Auflagerreaktionen sagt jedoch noch nichts über die Beanspruchung des Materials aus. Da die auftretenden Kräfte in der Regel nicht an jener Stelle, an der sie angreifen, in den Baugrund abgeleitet werden können, müssen diese über eine gewisse Distanz innerhalb der Konstruktion zu den Auflagern geführt werden (Lastableitungspfad), wodurch

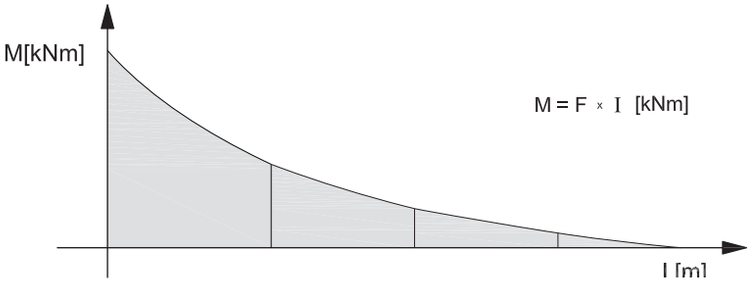


Abbildung 53: Beispiel für einen Momentenverlauf über die Länge eines Kragarmsystems.

ein Kraftfluss im Tragwerk entsteht. Bei dieser inneren Beanspruchung des Tragwerks können Kräfte in Richtung der Stabachse (Normalkräfte in x-Richtung), rechtwinklig zur Stabachse (Querkräfte in y- und z- Richtung) sowie Momente, welche die Stabachse verbiegen (um y- bzw. z-Achse) und verdrehen (um x-Achse = Torsion), unterschieden werden. Generell werden die oben angeführten Schnittkräfte an jeder Stelle des Tragwerkes ermittelt, wodurch sich die sogenannten Schnittkraftverläufe ergeben.

Schritt 4: Tragfähigkeit überprüfen

Durch die Kenntnis des gesamten Schnittkraftverlaufes über dem Tragwerk lassen sich jene Stellen mit extremalen[167] Werten leicht bestimmen. Es wird sichtbar, wo das System am meisten gezogen, gedrückt, gebogen, abgeschert bzw. tordiert wird. Dort muss überprüft werden, ob das Material den Belastungen genügend Widerstand entgegen zu bringen vermag oder versagt.

Das Verhältnis von Schnittkraft zu Fläche bei Normal- und Querkraftbeanspruchung bzw. zu Widerstandsmoment bei Momentenbeanspruchung wird Spannung [kN/cm^2] genannt. Durch die inneren Kräfte wird der Bauteil stets auch verformt.

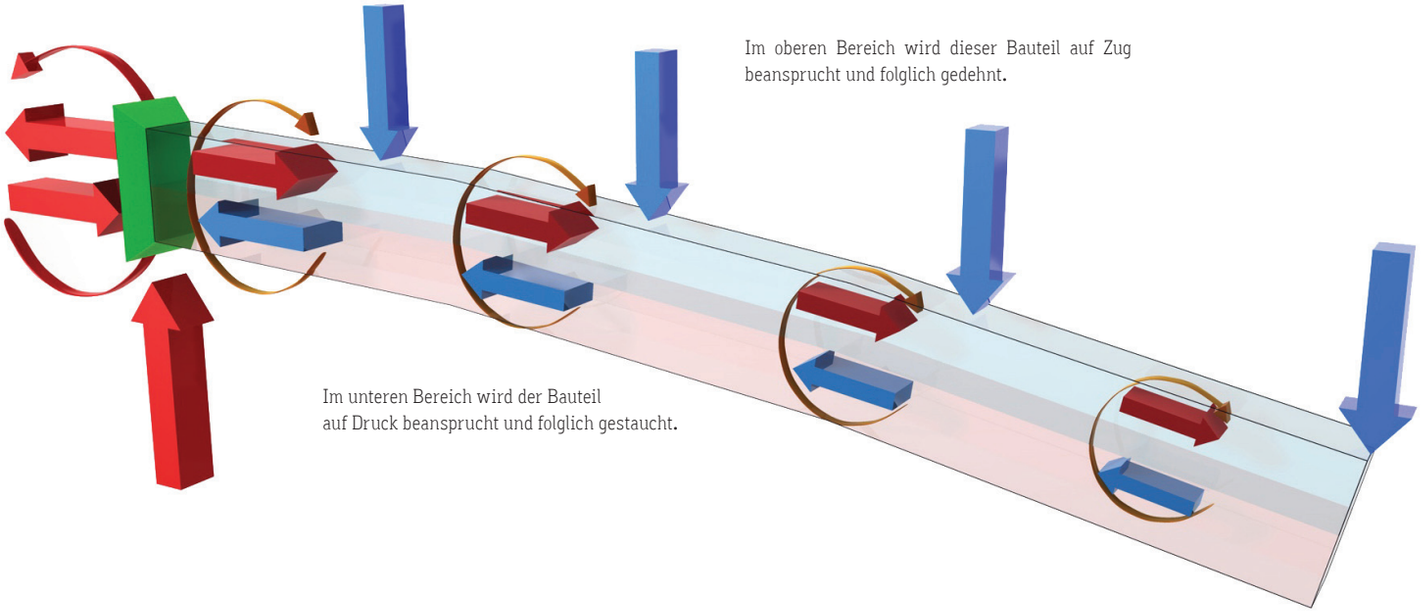
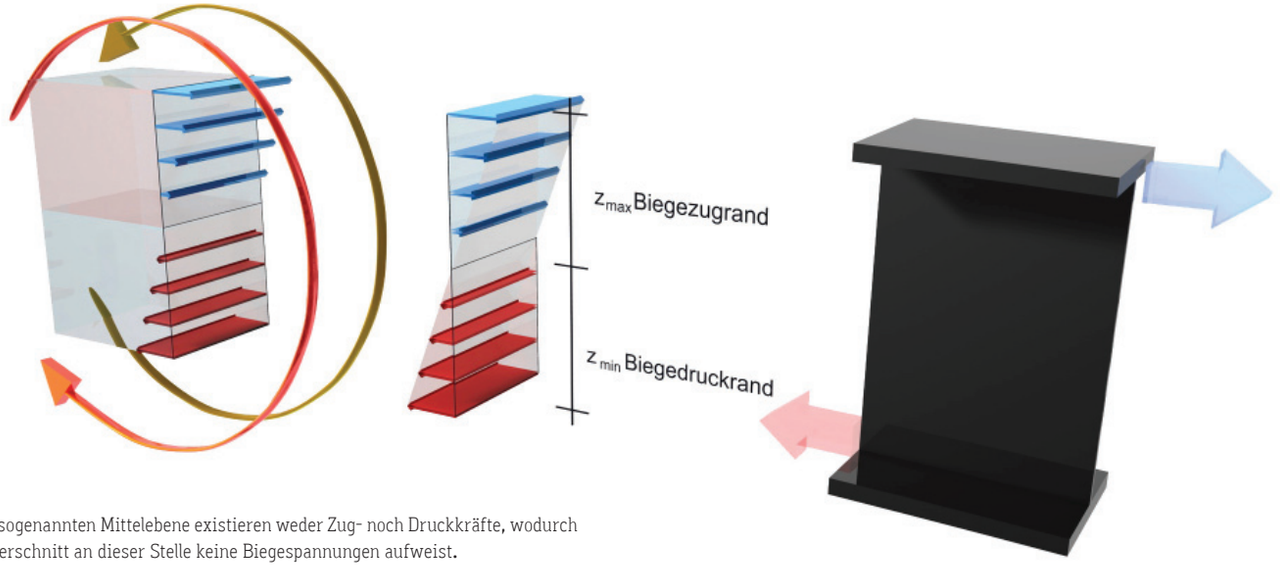


Abbildung 54: Zerlegen einer Momentenbeanspruchung in eine kombinierte Zug- und Druckbeanspruchung (Kräftepaar).

Der Querschnitt muss dem durch die Spannungen verursachten Biegemoment (Orange) ein Widerstandsmoment (Gelb) entgegensetzen.



In der sogenannten Mittelebene existieren weder Zug- noch Druckkräfte, wodurch der Querschnitt an dieser Stelle keine Biegespannungen aufweist.

Abbildung 55: Kragarm. Spannungsverteilung bei Biegebeanspruchung.

Die größte Beanspruchung (Spannung) tritt an den jeweiligen Rändern des Querschnitts (Randspannungen) auf. Der Spannungsverlauf im Querschnitt in Folge "Biegemomentenbeanspruchung" errechnet sich wie folgt:

$$\sigma(z) = \frac{M}{I} \times z \quad [kN/cm^2]$$

Das Moment „M" entspricht dem Maximalmoment „Mmax" aus der Schnittkraftermittlung, „z" ist die Laufvariable der vertikalen Richtung des lokalen Koordinatensystems mit Ursprung in der Schwerachse des Querschnitts.

Flächenträgheitsmoment [I]: $I = \int z^2 \, dA$

Das Flächenträgheitsmoment gibt ebenso wie das Widerstandsmoment an, wie resistent ein bestimmter Querschnitt gegenüber Biegung ist. Das Flächenträgheitsmoment hängt lediglich von der Geometrie des Querschnittes (Querschnittwert) ab.

Für beliebige Querschnitte wird die Ermittlung schnell komplex. Freiformquerschnitte werden zur Vereinfachung in linear berandete Teilflächen zerlegt. Das Flächenträgheitsmoment eines zusammengesetzten Querschnittes um seine Schwerachsen ergibt sich durch Anwendung des „Satz von Steiner". Für Standardgeometrien (Quadrat, Rechteck, Kreis, etc.) gibt es fertige Formeln.

Aus obiger Formel für den Spannungsverlauf im Querschnitt „σ(z)" kann abgeleitet werden, dass die Spannung für maximale bzw. minimale Werte von „z" ebenfalls extremal wird. Für diesen Fall kann der Ausdruck „I/zmin/max" durch das Widerstandsmoment „W" ersetzt werden.

Es ergibt sich die mininale bzw. maximale Randspannung zu:

$$\sigma_{min/max} = \frac{M}{W} \quad [kN/cm^2]$$

Das Sigma min/max wird mit der zulässigen des Materials verglichen. Damit der Spannungsnachweis erfüllt ist, muss die vorhandene Maximalspannung im Querschnitt kleiner sein als die zulässige:

$$\sigma_{min/max} \leq \sigma_{Zulässig}$$

Schritt 5: Gebrauchstauglichkeit überprüfen

Zusätzlich zum Nachweis, dass die vorhandenen Spannungen in sämtlichen Querschnitten nicht größer sind als die zulässigen, muss überprüft werden, ob auch die maximal vorhandenen Verformungen des Tragwerkes nicht über die zulässigen Werte hinausgehen. Zulässige Verformungen liegen je nach statischem System zwischen l/150 und l/400.

$$f_{vorh.} \leq f_{zul.}$$

Die Berechnung der Durchbiegung der Scripted Bridge wurde mittels der „Winkelgewichte Theorie"[168] ermittelt, die hier nicht genauer erklärt werden kann.

Bemessungsschritte zusammengefasst:

1. **Statisches System bestimmen:**
   Lasten und Auflagerkräfte definieren

2. **Lasten ermitteln und Auflagerreaktionen berechnen:**
   Prinzip "Aktion ist gleich Reaktion"

3. **Schnittkraftermittlung:**
   Normalkraft, Querkraft sowie Momentenverläufe über den Bauteil bestimmen.

4. **Tragfähigkeit überprüfen:**
   Überprüfen, ob die Querschnitte den auftretenden Kräften genug Widerstand entgegensetzen.

5. **Gebrauchstauglichkeit nachweisen:**
   Überprüfen, ob die maximal zulässige Verformung nicht überschritten wird.

Diesem Ablauf folgt auch die automatisierte Berechung der Scripted Bridge.

Bemerkung:
Sollte ein Bauteil erheblich unterdimensioniert oder auch überdimensioniert sein und wird daher der Querschnitt oder das Material geändert, reicht es nicht, lediglich die Überprüfung der Tragfähigkeit bzw. der Gebrauchstauglichkeit zu wiederholen. Es muss wieder bei „Schritt 2" angesetzt werden, da sich über die Veränderung des Querschnittes bzw. des Materials auch das Eigengewicht und damit eine Last im System verändert.

Sicherheit
Die Berücksichtigung unterschiedlichster Sicherheitsbestimmungen in Form von unterschiedlichsten Normen wie beispielweise der geforderte Nachweis der Stabilität gegenüber unterschiedlichsten Lasten machen besonders den Brückenentwurf komplex. In der vorliegenden Arbeit wurde versucht zumindest das sogenannte semiprobabilistische Sicherheitskonzept in die Berechnungen einzubeziehen. Das semiprobabilistische Sicherheitskonzept sieht sowohl auf Einwirkungsseite als auch auf Widerstandsseite sogenannte Teilsicherheitsbeiwerte vor. Mit anderen Worten werden Lasten mit Faktoren größer eins multipliziert (erhöht) sowie die Widerstände mit einem Faktor ebenfalls größer eins dividiert (abgemindert).

Generell lautet ein Nachweis im semiprobabilistischen Sicherheitskonzept. wie folgt:

$$S_k \times \gamma_k \leq \frac{R_k}{\gamma_k} \qquad S_d \leq R_d$$

Bemerkung zur Torsionsbeanspruchung:

Grundsätzlich wird bei den Querschnitten zwischen offenen oder geschlossenen unterschieden. Dabei leisten die offenen Querschnitte im Verhältnis zu geschlossenen Querschnitten gegen Torsionsbeanspruchung marginalen Widerstand. Auf eine detaillierte mathematische Definition wird hier nicht weiter eingegangen.

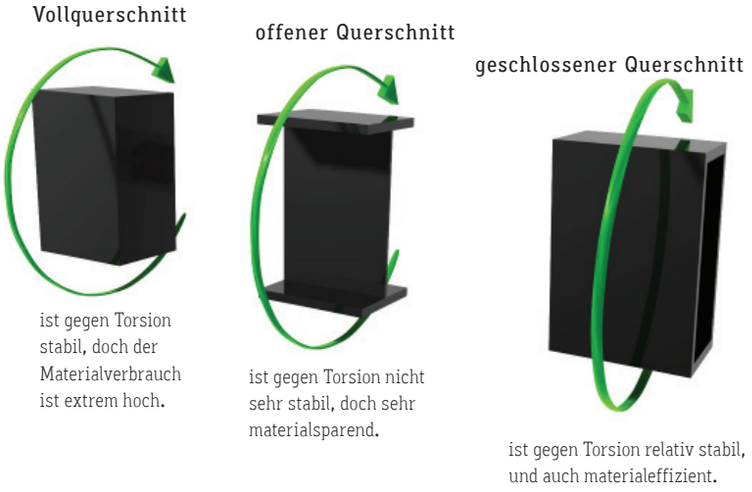
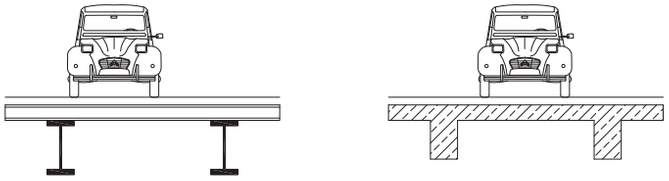


Abbildung 56: Unterschiedliche Querschnitte auf Torsionsbeanspruchung

Abschließend muss in Bezug auf die vorliegende Arbeit erwähnt werden, dass die Torsion um die Brückenlängsachse nicht berücksichtigt wurde, da vor allem bei beliebigen Querschnitten die Berechnung sehr komplex ist. Es wurde ein mehrzelliger Kastenquerschnitt vorgesehen, der für diese Art von Brückenform günstig und bei dem in der Regel ein Versagen aufgrund von Torsion unwahrscheinlich ist.

Typische Brückenquerschnitte:

offene Querschnitte:



geschlossene Querschnitte:

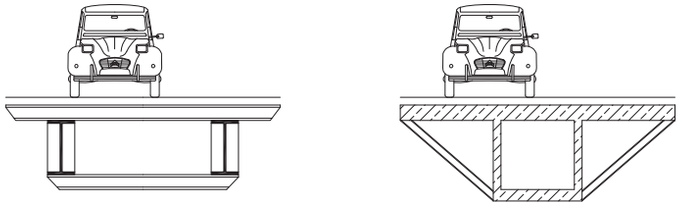


Abbildung 56: Ähnlich wie die einzenen Profile unterteilt man ganze Brückenquerschnitte in "offene" und geschlossene Querschnitte.

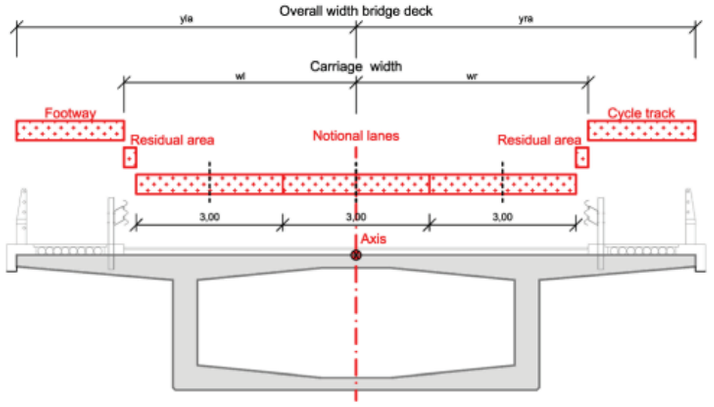


Abbildung 60: Definieren der Lastaufteilung auf dem Querschnitt.

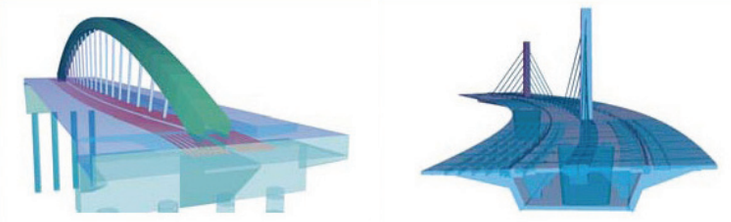


Abbildung 57: Brückenmodelle mit der Software Sofistik.

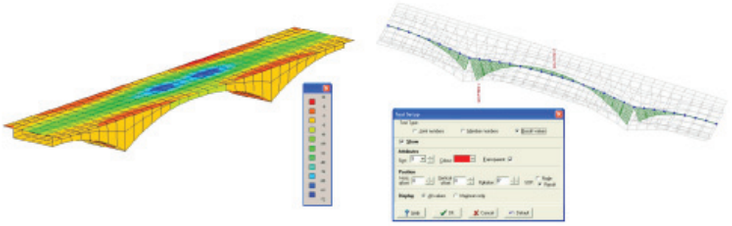


Abbildung 62: Analyse der Spannungen im Tragwerk.

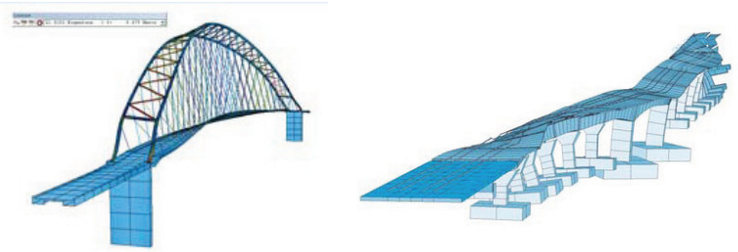


Abbildung 63: Simulation von Wind und Erdbebenlasten.

Brückenentwurf im digitalen Zeitalter

Der Computer wurde schon relativ früh für die Berechnung von Tragwerken eingesetzt. Der Bauingenieur Konrad Zuse arbeitete als Statiker in Berlin und entwickelte seinen ersten Rechner zu dem Zweck, Berechnungsaufgaben, die hin und wieder mehrere Tage in Anspruch nehmen konnten, zu automatisieren und damit rasch lösen zu können.[169]
Bereits Vannevar Bush (zu Bush siehe S.20-21) entwickelte am MIT ein mechanisches System zur Lösung von Differentialgleichungen und schuf damit die Grundlagen für die heutige „Finite Elemente Methode" (in Folge FEM). Mit der FEM können komplexe Differentialgleichungen gelöst werden, indem zwei- oder auch dreidimensionale Objekte in einen Raster zerlegt werden, der aus einer begrenzten Anzahl von Elementen zusammengesetzt wird. Je feiner der Raster, desto genauer und zeitaufwändiger wird die Berechnung. Gegenwärtig werden die sogenannten FE-Programme in Kombination mit den bereits erwähnten Entwicklungen im CAD und CGS Bereich im Bauingenieurwesen als leistungsfähige Analyse- bzw. Simulationsprogramme für das Entwerfen und Dimensionieren von Tragwerken eingesetzt. [170]

Je nach Aufgabengebiet der AnwenderInnen werden mehr oder weniger stark spezialisierten Softwarepakete verwendet. Dementsprechend haben sich einige Hersteller ausschließlich auf Brückendesignsoftware (BDS) spezialisiert oder bieten eigenständige Programme oder Erweiterungen bzw. Aufsätze zu ihren anderen Produkten an.[171]

Bezugnehmend auf die vorliegende Arbeit sei erwähnt, dass das Büro „Structural Design Olipitz (SDO)" zwar einige Brücken entworfen und realisiert hat, dazu jedoch keine spezifische Brückenentwurfssoftware verwendet. Daher wurde die im Zuge des Wettbewerbs „Connecting Link" entstande Variante der „Scripted Bridge" mit RStab (Dlubal) analysiert und bemessen.[172]

Die Spezialisierung einzelner Programme auf den Brückenentwurf erlaubt eine schnellere Bearbeitung der Konzepte von der Idee bis zur Realisierung der Entwürfe. Die Software Sofistik stellt neben den weitverbreiteten parametergesteuerten Brücken- und Straßenquerschnitten beispielsweise komplexe Bauteile wie Autobahnauffahrten, Rampen, Kurven, Straßenabzweigungen etc. bereit, wobei diese sich ebenfalls über Parameter wie Radien, Steigungen usw. an individuelle Situationen anpassen lassen. Es wird versucht, eine hohe Automatisierung und eine möglichst durchgehende Kette im Planungsprozess und damit eine möglichst starke Einbindung in den Entwurfsprozess zu erreichen. In der Regel sind Brückenentwurfsprogramme sehr ähnlich aufgebaut.

Laut Georg Pircher[173] unterstützen die meisten Programme benötigte Normen, typische Lastfälle (Fußgänger, Autos, Züge, etc.) und eine große Anzahl an parametrisierten Querschnitten. Brückenentwürfe werden zwar häufig mittels veränderlicher Parameter entworfen, diese seinen laut

Abbildung 58: Aashto Girder Brücke Typ 4 in Georgia USA. Diese Brücken werden mittels Tabelleneingabe automatisch generiert.

Pircher aber sehr umfangreich und bei jedem Entwurf individuell. Komplett fertige Brückenentwurfssystem gibt es vor allem in den USA. Dort werden etwa die "AAshto Girde Bridges"[174] mittels Ausfüllen von fertigen Tabellen entworfen und folglich autmatisch generiert.

Der Planungsprozesses ist meist in unterschiedliche Modellierungsschritte wie die der Geometrie bzw. des statischen Grundmodells, Lastenmodellen sowie auch Simulationsszenarien unterteilt. Durch diesen in Schritte unterteilten Prozess ensteht in der Regel ein linearer Arbeitsfluss bei der Konstruktion von statischen Systemen. Dabei ergibt sich das Problem, dass bei Veränderungswünschen vor allem an komplexeren Systemen, diese in das interne oder externe Modellierungssystem zurück exportiert bzw. nach einer neuen Modellierung wieder importiert werden müssen und dabei die Informationen über das System wie etwa Auflager, Gelenke, Lasten

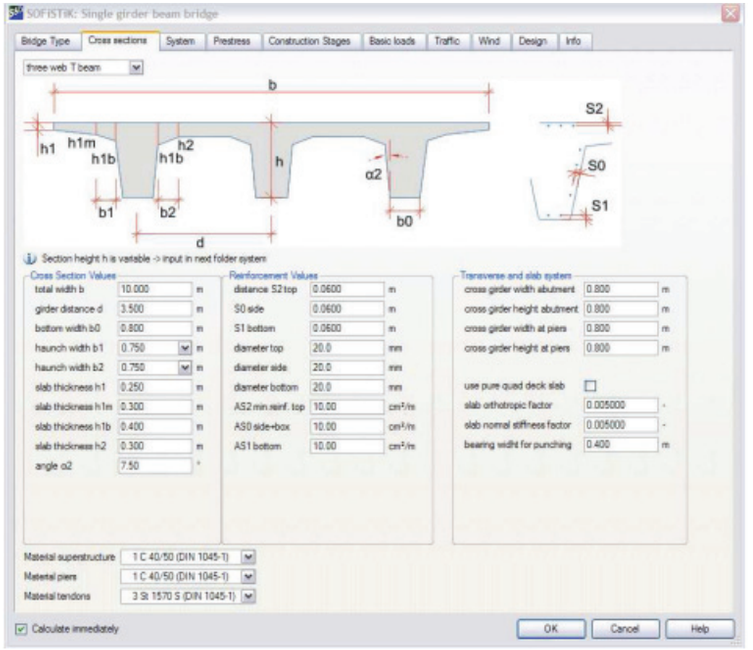


Abbildung 59: Typische Querschnittseditor. Querschnitte können mittels Parameter definiert werden.

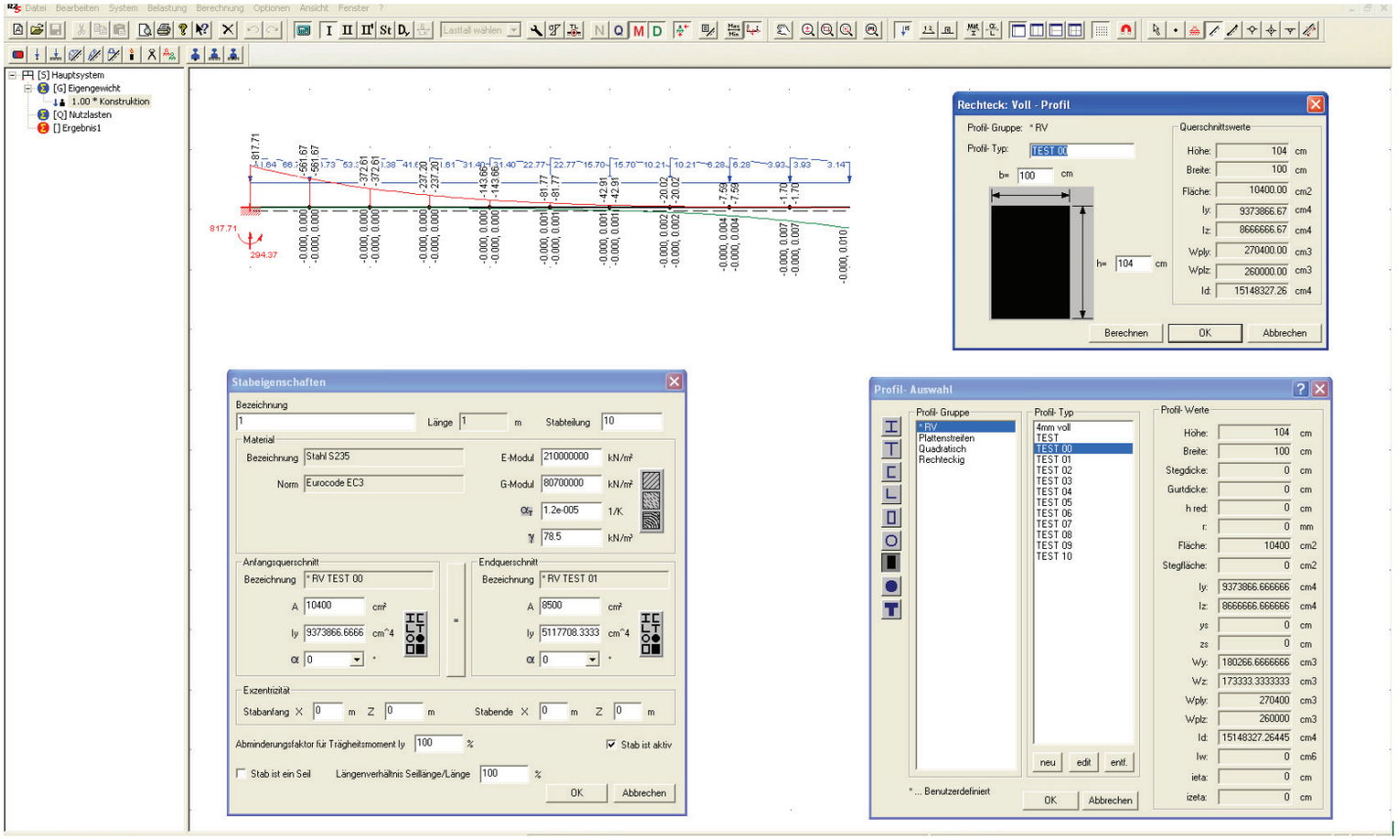


Abbildung 61: Benutzeroberfläche der Software Ruckzug von Mursoft.

etc. verloren gehen. Dadurch wird ein intuitives direktes Modellieren und spielerisches Ausprobieren im Entwurfsprozess unmöglich. Wird etwa die Grundgeometrie geändert, müssen die Folgeschritte meist angepasst oder komplett neu modelliert werden.

Üblicherweise wird in einem ersten Schritt das statische Grundmodell als Achsenmodell mit Hilfe eines einfachen Editors erstellt.

In einem weiteren Schritt werden den einzelnen Linien des Achsmodells Querschnitte zugewiesen, wodurch das gesamte Modell ein Volumen und ein Material erhält. Die Modellierungsmöglichkeiten für Flächen- oder Volumengeometrien sind meist nicht vorhanden oder sehr beschränkt. Funktionen wie Loften, Extrudieren, das Erstellen von Rotationskörpern etc. sind zumeist nicht möglich. Aus diesen Gründen wird die Grundgeometrie komplexer Modelle in der Regel aus anderen Programmen (CAD oder CG-Software) über gängige Austauschformate (dxf, dwg, 3ds, obj, iges, etc.) importiert werden. Die Software Sofisitik (SOFiSTiK AG) kann beispielsweise als Plugin in AutoCad (Autodesk) integriert werden, wodurch das Grundmodell in AutoCad erstellt werden kann. Nach Fertigstellung des Models muss diese zur Berechnung in Sofistik konvertiert werden.

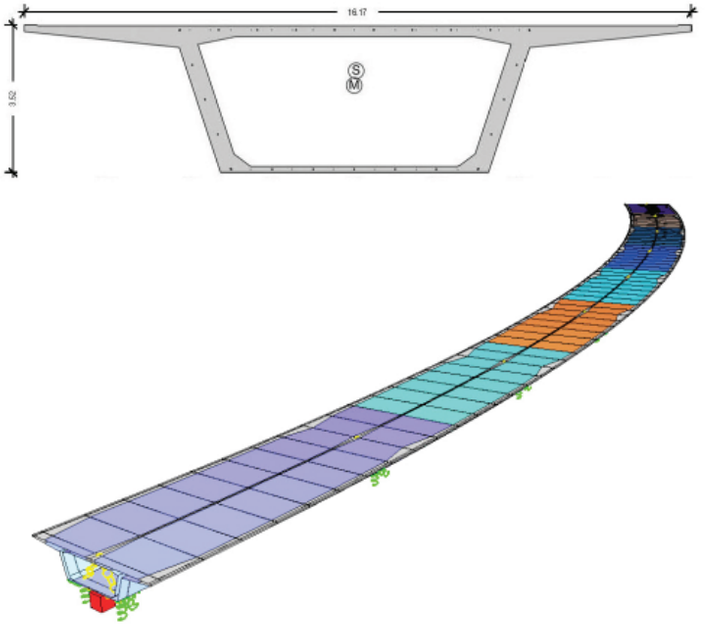


Abbildung 64: Hier wird ein Brückenquerschnitt entlang einer Achse definiert.

Nachdem die Grundgeometrie feststeht, kann das statische System modelliert werden, indem die Verbindung der einzelnen Stäbe bzw. Elemente und die Lagerung des Gesamtsystems definiert wird. Je nachdem, welches Verhalten analysiert bzw. simuliert werden soll, müssen nun ständige Lasten wie Eigenlast oder Lasten durch Vorspannungen etc. sowie nicht ständige Lasten wie Wind, Schnee, Erd-/Wasserdruck etc. sowie auch außergewöhnliches Lasten wie Erdbeben, Brände, Explosionen etc. definiert werden. Möglich sind auch Kombinationen einzelner Lastfälle. Ähnlich wie in CAD- bzw. Animationsprogrammen werden auch in diversen Statikprogrammen Scripting-Techniken unterstützt, um dem/der BenutzerIn individuelle Funktionserweiterungen und Automatisierungen bei der Modellierung zu ermöglichen. Dabei geht es laut Georg Pircher meist um das Scripten unterschiedlicher Lastfälle.[175]

So wird beim Entwerfen von Brücken in der Regel auf Statiksoftware zurückgegriffen, die für eine sinnvolle Verwendung bereits für die Beurteilung einfacher Systeme große Fachkenntnisse voraussetzt, wodurch diese Software vorwiegend von BauingenieurInnen verwendet wird. Die Software „RuckZuck" der Firma Mursoft ist ein Beispiel für eine relativ einfach zu bedienende Software, weshalb diese an der Technischen Universität Graz am Institut für Tragwerksentwurf den ArchitekturstudentInnen einführend nähergebracht wird. Ein großer Nachteil besteht jedoch darin, dass dieses Programm stark zwei dimensional operiert und die Modellierung von freieren Systemen nicht möglich ist.

Eine als Alternative für die Findung von Tragwerksstrukturen sind etwa Topologieoptimierungssysteme wie sie etwa Arata Isozaki im Entwurf einsetzt (Beispiel siehe S.48-49). Diese Programme sind zwar einfach zu bedienen, sind jedoch lediglich wie in der Natur auf einen Lastfall hin optimiert und verwenden isotrope Materialien zur Generierung der Formen (siehe Freeware Beispiel-Applikation im Internet)[176]. Daher wird die Konstruktion beim Entwerfen nicht berücksichtigt, wodurch sich meist eine große Diskrepanz zwischen der finalen Geometrie und dem Tragwerk ergibt. Das Problem besteht mE etwa beim Bahnhof von Arata Isozaki nicht darin, das das Tragwerk nicht wirklich effizient ist, sondern in der gebauten Form so tut als ob. Weiters produzieren letztgenannte Programme feinmaschige Netze mit einer Vielzahl an mehr oder weniger regelmäßigen Polygonen. Dies erschwert die nachträgliche Weiterverarbeitung insbesondere bei Veränderungen der Geometrie erheblich.

Schlaich und Baus bemerken in ihrem Buch „Fußgängerbrücken", dass Brückenentwürfe zwar immer häufiger mit dem Computer generiert werden, weisen aber darauf hin, dass nur ambitionierte ArchitektInnen und IngenieurInnen, *„die das Entwerfen und Konstruieren von Tragwerken von Grund auf gelernt haben"*[177], bei diesem Experimentieren sinnvolle Brückenentwürfe hervorbringen können. Bezüglich der Rolle des Computers im Entwurf führen Schlaich und Baus aus: *„Mehr als ein Werkzeug kann und darf der Computer beim Experimentieren nicht sein."*[178]

Softwarehersteller:

D.I.E. CAD und Statik Software GmbH:
D.I.E, http://www.die.de

Ing.-Software Dlubal GmbH:
RFEM, RSTAB, RX-Holz, etc., http://www.dlubal.de
Mursoft:
RuckZuck, http://www.ruckzuck.co.at

Computers & Structures Inc.:
SAP2000 Bridge Design (BDS), SAP2000, ETABS, SAFE, PERFORM-3D, etc., http://csiberkeley.com

RISA Technologies:
RISA-3D, RISA Tower, RISA Base, RISA Masonry, etc.
http://www.risa.com

SOFiSTiK AG:
SOFiSTiK Brückenbau (BDS), SOFiSTiK Structural Desktop, SOFiSTiK Dynamik, etc.
http://www.sofistik.de

LUSAS:

Lusas Bridge (BDS), Lusas Civil Structural, Lusas High Precision Moulding, etc., http://www.lusas.com/

Bentley Systems:
RM Bridge (BDS), STAAD.beava (Brückendesign), STAAD.Pro, STAAD(X). Tower, etc., http://www.bentley.com

Bestech Systems Limited:
Sam (BDS), http://www.bestech.co.uk

LARSA, Inc.:
LARSA 4D (BDS), http://www.larsausa.com

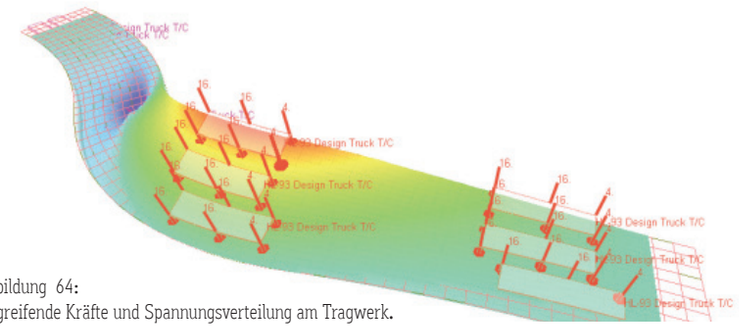


Abbildung 64:
Angreifende Kräfte und Spannungsverteilung am Tragwerk.

136 David J. Brown, "Brücken: Kühne Konstruktionen über Flüsse Täler Meere", deutsche Übersetzung, Verlag Georg D. W. Callwey GmbH & Co. KG, 2005, S.6.

137 Ursula Baus, Mike Schlaich, "Fußgängerbrücken: Konstruktion Gestalt Geschichte", Birkhäuser Verlag 2008, S.6.

138 Vgl. ebd., S.120.

139 Vgl. ebd., S.71.

140 Vgl. Dirk Bühler, "Brückenbau im 20. Jahrhundert", Verlag Georg D. W. Callwey GmbH & Co. KG, 2005, Jörg Schlaich, S.10.

141 Vgl. Ursula Baus, Mike Schlaich, "Fußgängerbrücken [...]", a. a. O.,, S.163.

142 Vgl. Dirk Bühler, "Brückenbau im 20. Jahrhundert", a. a. O., Jörg Schlaich, S.7.

143 Vgl. ,Volkwin Marg, "leicht weit: Light Structures", Jörg Schalich, Hg. Annette Bögle, Peter Cachola Schmal, Ingeborg Flagge, Prestel Verlag Deutschland, 2004 S.44-51 oder mit dem Fokus auf Brücken Vgl. Leonardo Fernández Troyano, Bridge Engineering: A Global Perspective, 2003, S.9-20.

144 Jörg Schlaich, im Gespräch mit Ingeborg Flagge, "leicht weit: Light Structures", a. a. O., S.10.

145 Vgl. Dirk Bühler, "Brückenbau im 20. Jahrhundert", a. a. O., Jörg Schlaich S.8.

146 Michel Virlogeux, International Ferderation for Structural Concret (fib), "Guidance for good bridge design", 2000, S.1.

147 Maillart, zitiert nach Schlaich, Robert Maillart, "Brückenschläge", Höhere Schule für Gestaltung Zürich, Schriftenreihe Nr. 13, Redaktion Claude Lichtenstein, 1990, S.31. Originalquelle: D. P. Billington, Civil Engineering: History, Heritage of Humanities II, Princton 1973, S.149.

148 Jörg Schlaich, im Gespräch mit Ingeborg Flagge, a. a. O., S.8.

149 Ebd., S.14.

150 Vgl. Ursula Baus, Mike Schlaich, "Fußgängerbrücken [...]", a. a. O., S.12.

151 Dirk Bühler, "Brückenbau im 20. Jahrhundert", Schlaich, Jörg Schlaich S.7.

152 Michael Blaschko, "Massivbau in ganzer Breite", Förderverein Massivbau der TU München e.V. u. Roland Niedermeier, Springer Verlag Berlin Heidelberg, 2005, Abschnitt: Architektur fordert Ingenieure - Die neue Svinesundbrücke, S.43.

153 Santiago Calatrava, Interview mit Charlie Rose, http://kempton.wordpress.com/2009/07/30/calgary-peace-bridge/ (09.04.2010).

154 Vgl. Michel Virlogeux, International Ferderation for Structural Concret (fib), a. a. O., S.37-38 oder Ursula Baus, Mike Schlaich, "Fußgängerbrücken [...]", a. a. O., S.16.

155 Fritz Leonhardt, Brücken: Ästhetik und Gestaltung, Deutsche Verlags-Anstalt, 3. Auflage 1990. S.10.

156 Vgl. http://de.wikipedia.org/wiki/Viaduc_de_Millau, (12.04.2010) oder Richard Hammond's, Fernsehserie: Engineering Connections, Millau Bridge, 8. September 2008 (UK).

157 Vgl. Michel Virlogeux, International Ferderation for Structural Concret (fib), a. a. O., S.37-38.

158 Ebd., S.27.

159 Vgl. Ursula Baus, Mike Schlaich, "Fußgängerbrücken [...]", a. a. O., S.12.

160 Kurt Ackermann, Architekt – Ingenieur, Karl Krämer Verlag, 1997, S.7.

161 Ebd., S.8.

162 Ebd., S.15.

163 Vgl. ebd., S.9.

164 Vgl. Ursula Baus, Mike Schlaich, "Fußgängerbrücken [...]", S.12.

165 Arup, Definiton Bridge Architect bzw. Bridge Designer http://en.wikipedia.org/wiki/Bridge_architect (11.04.2010).

166 Als Studienassistent des Instituts für Tragwerksentwurf sind mir diese Grundlagen bekannt. Die Informationen wurden mit Ass.Prof. Dipl.-Ing. Dr.nat.techn Andreas Trummer dem Betreuer dieser Arbeit genau durch besprochen. Besondere Dank für detaillierte Informationen gilt Dipl.-Ing. Helmut SCHOBER.Diese allgemeinen Grundlagen finden sich beispielsweise in den Büchern: Leicher Gottfried, Tragwerkslehre in Beispielen und Zeichnungen, 2006, Werner Neuwied Verlag oder auch Franz Krauss, Wilfried Führer, Hans J. Neukäter Grundlage der Tragwerkslehre, 2002, Verlagsgesellschaft Müller.

167 Anmerkung: Der Grundbegriff „extremal" kommt aus der Graphentheorie und bezeichnet lokale extreme Werte wie maximale und minimale Werte (beispielsweise Hoch- und Tiefpunkte in einer Kurve). Siehe dazu: http://de.wikipedia.org/wiki/Extremwert (16.04.2010).

168 Die Berechnung der Durchbiegung mittels Winkelgewichten wird beispielsweise im Baustatik 1 Skript des Instituts für Baustatik an der TU-Graz detailliert erklärt. http://www.ifb.tugraz.at/backup_old_homepage/educ/teaching_material/Baustatik/Baustatik1_Skriptum_IBK.pdf, (Stand 04.04.2010).

169 Vgl. Edgar P Vorndran, Entwicklungsgeschichte des Computers, VDE-Verlag GmbH Berlin und Offenbach, S.75.

170 Vgl. http://en.wikipedia.org/wiki/Finite_element_method (15.04.2010).

171 Die Aussagen dieses Kapitels beziehen sich auf die Information die ich in Gesprächen am Institut für Tragwerksentwurf mit dem Betreuer dieser Arbeit geführt habe. Detaillierte Informationen bezüglich der Einsatzgebiete bestimmter Statiksoftware erhielt ich in einem Gespräch mit dem Softwareentwickler Georg Pircher von ABES Pircher & Partner GmbH am 20.10.2010 in Graz. Als weiter Quellen diente die Information auf den Homepages verschiedener Softwarehersteller (siehe Liste).

172 Gespräch mit Dipl.- Ing. Helmut Schober; Schober arbeitet oft mit SDO zusammen, http://www.olipitz.com, (19.04.2010).

173 Georg Picher ist Geschäftsführer und Gesellschafter von der Software Entwickler Firma ABES Pircher & Partner GmbH. http://www.abes-austria.com, (20.10.2010).

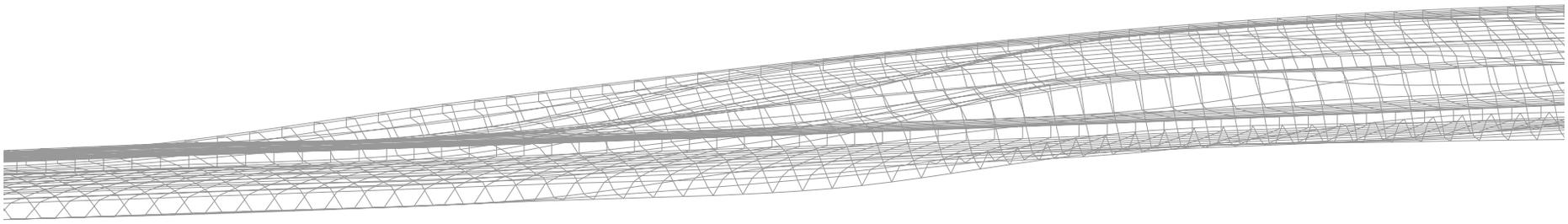
174 AASHTO steht für "American Association of State Highway and Transportation Officials". Die Brückenentwürfe für das amerikanische Straßennetzwerk sind hochautomatisiert. http://www.transportation.org (20.10.2010).

175 Gespräch mit Georg Pircher am 20.10.2010, Graz, im Büro von ABES Pircher & Partner GmbH. Bemerkung: In diesem Gespräch hat mich Georg Picher allgemein über Statiksoftware informiert.

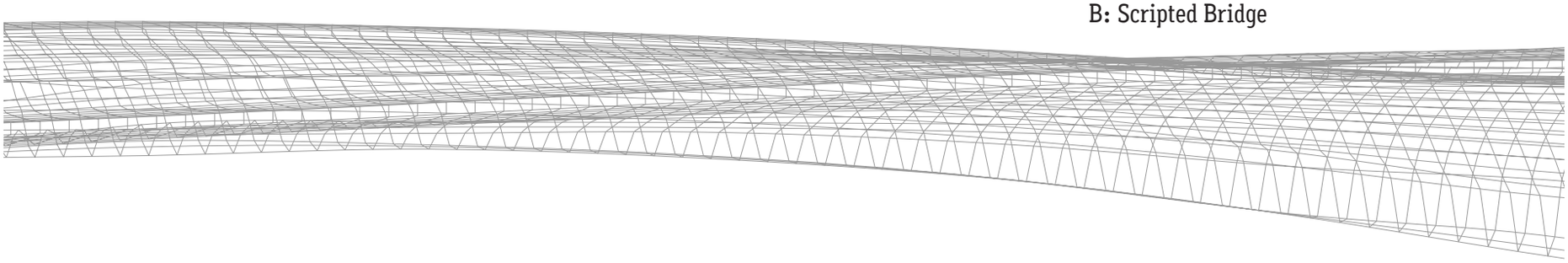
176 Beispiel für ein frei zugängliches Topologie Optimierungsprogramm im Internet, TopOpt 3d, Topopt Research Group, Dänemark, http://www.topopt.dtu.dk/?q=node/11, (Stand 20.04.2010).

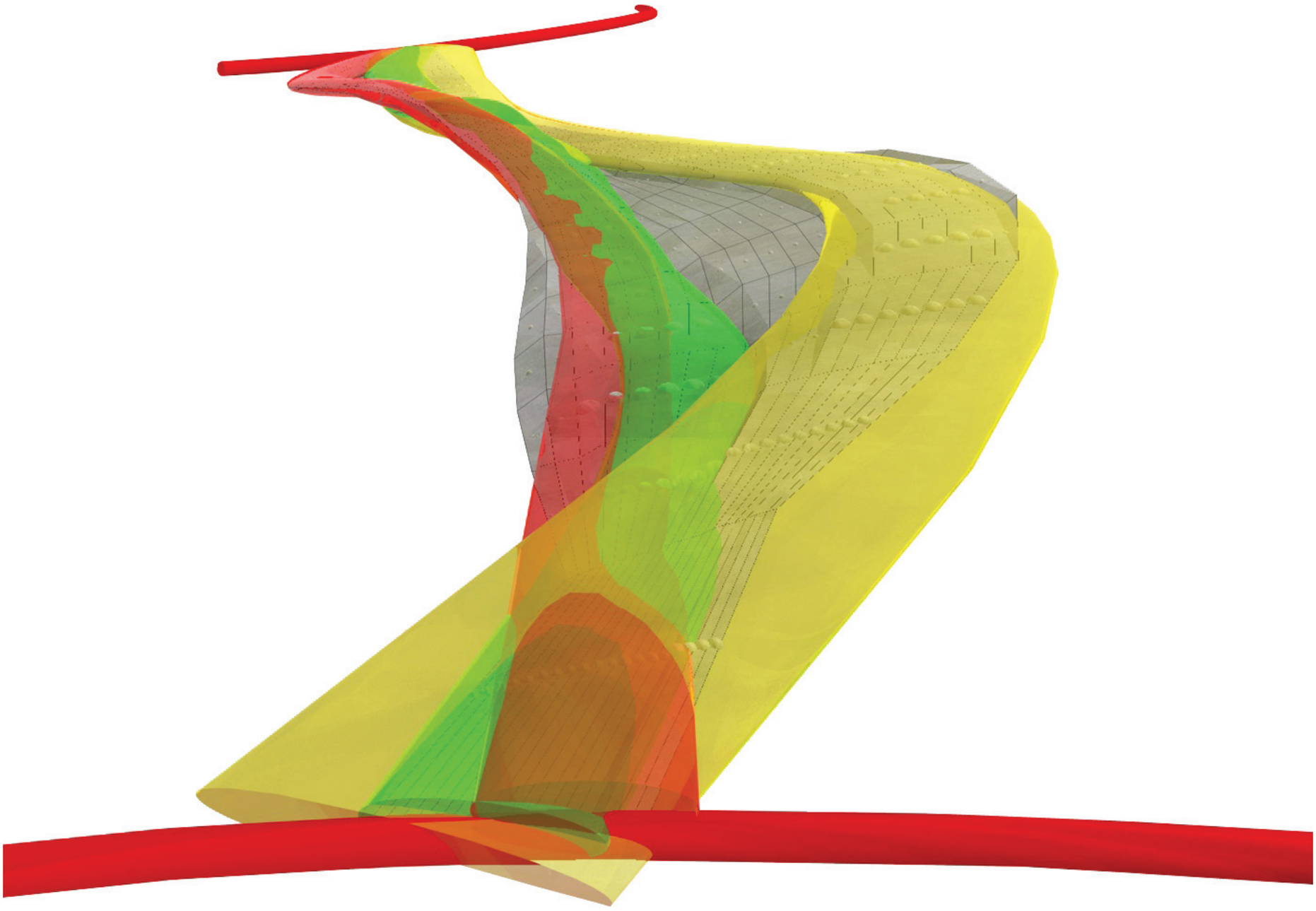
177 Ursula Baus, Mike Schlaich, Fußgängerbrücken: Konstruktion Gestalt Geschichte, Birkhäuser Verlag 2008, S.105.

177 Ebd., S.105.



B: Scripted Bridge





B: Scripted Bridge

Aufgabenstellung

Allgemeines

Wie bereits in der Einleitung erwähnt, soll es Ziel dieser Arbeit sein, anhand eines konkreten Entwurfes zu überprüfen, inwieweit sich dieser über Parametersteuerung automatisieren lässt, um etwa unterschiedliche Varianten erzeugen zu können und zu erwägen, wie stark sich die Animationssoftware mittels „Scripting" und der Verwendung von „Mashup-Tools" in den Entwurfsprozess einbinden lässt. Im folgenden wird in der Aufgabenstellung das übergeordnete Ziel für den Brückengenerierungsprozess in Form einer Grundidee präzisiert, die Themenwahl des Brückenentwurfes begründet, um anschließend die konkreten Rahmenbedingungen anhand der Wettbewerbsausschreibungen zu erläutern.

Grundidee

Für das Entwerfen von Tragsystemen greifen insbesondere Architekten zumindest in der Entwurfsphase häufig auf gängige Vorbemessungsformeln zurück. Diese allgemeinen Formeln gelten nur sehr eingeschränkt für gängige statische Systeme, übliche Konstruktionstypen sowie nur für Querschnitte, die über die gesamte Länge eines Systems konstant bleiben. Dementsprechend wird in dieser Arbeit in Bezug auf gegenwärtige Tendenzen in der Architektur geprüft, inwieweit mittels Scripting- bzw. Mashup-Systemen eine Vorbemessung für Konstruktionen, die von der Norm abweichen, möglich ist. Da die Gestalt von Brücken vor allem aus Sicht der ArchitektInnen nicht ausschließlich aus Problemen des Tragverhaltens bestimmt wird, wurde dabei versucht, einen offenen Prozess zu gestalten, in den auch andere Kriterien einfließen können.

Entwurfsthema

Die Motivation, einen Brückengenerierungsprozess als allgemeines Entwurfsthema zu wählen, wird damit begründet, dass das Entwerfen von Brücken als Königdisziplin der BauingenieurInnen stark verwissenschaftlicht gilt und für die Organisation in einem parametrisierten Regelwerk hervorragend geeignet zu sein schien. Außerdem markiert der Brückenentwurf als „Königsdisziplin der BauingenieurInnen" ein interessantes Spannungsfeld zwischen ArchitektInnen und IngenieurInnen, dessen Betrachtung insbesondere in Hinblick auf meine Tätigkeit als Studienassistent am Institut für Tragwerksentwurf besonders anredend darstellte.

Abbildung 65: Der Entwurfsort liegt am Donaukanal und befindet sich nordöstlich des historischen Zentrums von Wien. Der gelbe Pfad markiert den aktuellen Hauptradfahrweg während das kurze rote Stück die geplante Lückenschließung kennzeichnet.

Rahmenbedingung: Brücke über den Wienfluss

Im Folgenden werden die Ausschreibungen der zwei Wettbewerbe "Concrete Student Trophy 2009" und "Connecting Link" zusammenfassend dargestellt.[179] Diese Ausschreibung geben die grundlegenden Rahmenbedingungen für den Brückenentwurf vor und werden für die Parameterfindung verwendet.

Allgemeines

Die Stadt Wien ist bestrebt, im Zuge von Infrastrukturverbesserungen das derzeit bestehende, lückenhafte Radwegenetz auszubauen. Dadurch soll insbesondere die Sicherheit von Fußgängern und Radfahrern erhöht werden. Als derartige Infrastrukturverbesserung soll eine Brücke im Mündungsbereich des Wienflusses in der Nähe der Urania im ersten Wiener Gemeindebezirk entstehen. Die Brücke soll das Ufer des Herrmannparks mit dem Vorkai der Urania verbinden. Derzeit sind Radfahrer und Fußgänger an dieser Stelle gezwungen, den Wienfluss über die Radetzkybrücke zu überqueren, wobei dies nur durch die Nutzung von bestehende Stiegen und Rampen möglich ist. Für diese Vorhaben sind vorwiegend zwei örtliche Gegebenheiten prägend. Zum einen befindet sich der gewünschte Brückenstandort in historischem Kontext bzw. in denkmalgeschützer Umgebung. Zum anderen dient der Donaukanal im Bereich zwischen Aspern und Franzensbrücke als Wendeplatz für den Schiffsverkehr. Dabei wird für diese Wendemanöver besonders der Mündungsbereich des Wienflusses bis zur Sohlschwelle knapp vor den Pfeilern der Radetzkybrücke ausgenützt.

Der Entwurfsort

Wie bereits erwähnt liegt das gesamte Planungsgebiet in einem denkmalgeschützten Bereich. Demzufolge ist das Planungsgebiet seit dem Jahr 2001 ein Teil des Weltkulturerbe-Areals "Wien-Innere Stadt". Die direkt umliegenden Bauten wie beispielweise das Gebäude der Urania, die Radetzkybrücke oder der Zollamtsteg stehen unter Denkmalschutz. Außerdem sind die Kaimauern des Donaukanals und die Wienflussverbauung denkmalgeschützt.

Der Entwurfsort liegt an der Schnittsstelle zwischen historischem und modernem Wien. Dies bedeutet, es herrscht ein Kontrast zwischen den um die Wende des 19. Jahrhunderts entstandenen späthistoristischen Gebäuden und den modernen Gebäuden des 20. und 21. Jahrhunderts. Beispielsweise steht sich das historistische „Urania" Gebäude von Max Fabian dem „Uniqua Tower" von Neumann und Partner direkt gegenüber.

Das heutige Erscheinungsbild von Wienfluss und Donaukanal geht auf den gegen Ende des 19. Jahrhunderts erfolgten großen Ausbau der Verkehrsanlagen in Wien zurück. „Während der Donaukanal bis heute im Wesentlichem von der architektonischen Gestaltung durch Otto Wagner gekennzeichnet ist, verdankt der Wienfluss im Mündungsgebiet seine architektonische Prägung vor allem den Architekten Friedrich Ohmann und Josef Hackhofer."

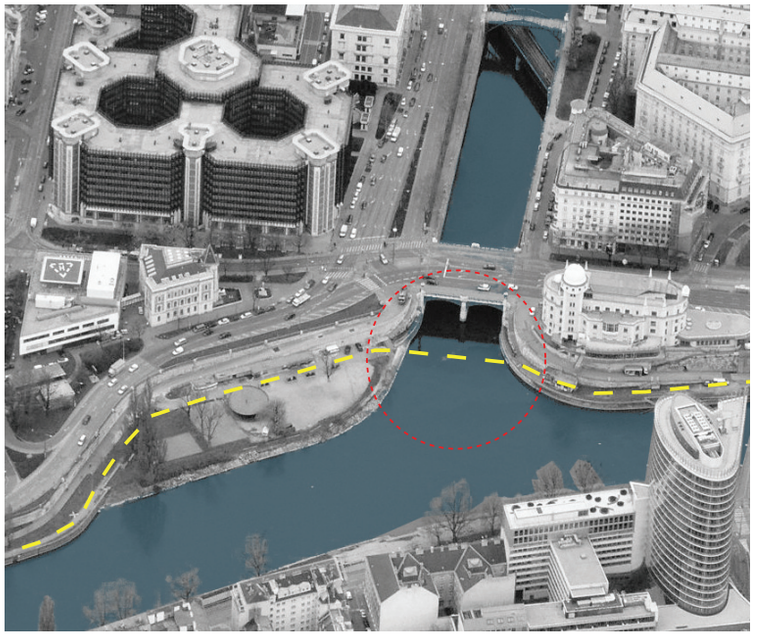


Abbildung 66: Der Donaukanal fließt von rechts nach links während der Wienfluss vom Zollamsteg - oberer Bildrand) kommend nach der Radezkybrücke in den Donaukanal mündet. Links im Mündungsbereich bedindet sich der Herrmannpark und rechts gegenüber das Gebäude der Urania. Der Uniqua Tower ist rechts im unteren Bildrand gegenüber der Urania zu erkennen.

Der Donaukanal wurde als natürlicher Seitenarm der Donau erstmalig um die Wende vom 16. zum 17. Jahrhundert künstlich ausgebaut und wurde später wie bereits erwähnt wesentlich von Otto Wagner als Mitglied der ehemaligen Donauregulierungskommission architektonisch geprägt. Während im 20. Jahrhundert das Erscheinungsbild des Areals hauptsächlich durch den Neubau der Stadtbahn bzw. der Einbeziehung des U-Bahnnetzes bestimmt wurde, unterliegt dieses nun Veränderungen durch gegenwärtige Tendenzen in Richtung einer Planung eines Freizeit- und Nahversorgungsraumes. Dennoch sollen laut der Stadt Wien auch in Zukunft die historischen Gestaltungselemente, Promenaden und Ufergestaltungen kennzeichnend für diesen Stadtraum bleiben.

Der Wienfluss entspringt dem Wienerwald westlich der Stadt. Zwar wurde der natürliche Flusslauf bereits im 18. Jahrhundert verändert. Wie

Abbildung 67: Der denkmalgeschützte Zollamtsteg mit darunterliegender U-Bahntrasse.


MITTLERER SAAL
HAGRO TECHNIK
LÜFTUNG · FILTER · KLIMA

bereits erwähnt wird das heutige Erscheinungsbild von den Änderungen gegen Ende des 19. Jahrhunderts bestimmt, welches bereits im Jahr 1903 vom Stadtbauamt bis auf wenige Teilarbeiten fertig gestellt wurde. *„Der gesamte Mündungsbereich ist durch das regulierte Wienflussbett mit den hohen, senkrechten, mit Naturstein verkleideten Stützmauern mit ihren gusseisernen Gittern, gekennzeichnet. [...] Für den unmittelbaren Bauplatz ist die Radetzkybrücke sowie die Sicht auf den Zollamtssteg mit der darunter schräg kreuzenden ehemaligen Stadtbahnbrücke von besonderer Bedeutung."*

Die Radetzkybrücke wurde an der Stelle eines Vorgängerbaus errichtet und wurde im Jahre 1899 zur Benützung freigegeben. Die Planung und Ausführung erfolgte durch das Stadtbauamt, wobei für die architektonische Gestaltung die Architekten Josef Hackhofer und Friedrich Ohmann verantwortlich waren. Hackhofer und Ohmann waren zur selben Zeit auch für die Gestaltung der Stubenbrücke, der kleinen Marxerbrücke und den Zollamsteg zuständig. Das Tragwerk der Radetzkybrücke wird mit Bogenträger aus Eisen realisiert, die an den Uferzonen jeweils auf Mauern und in der Mitte des Wienflusses auf einem Steinpfeiler aufliegen. Die geplanten Leuchttürme an den Brückenenden wurden erst Jahre später realisiert. Im Jahr 1945 wurde die Brücke beschädigt und erst 1952 wieder in Betrieb genommen. Nach einer zweiten Renovierung im Jahre 2001 erscheint die Brücke wieder in ihrer ursprünglichen Gestalt.

Das Gebäude der Wiener Urania bildet den städtebaulichen Abschluss der Wiener Ringstraße und wurde vom Architeken Max Fabiani – einem Schüler Otto Wagners - in den Jahren 1909-1910 als Volksbildungshaus errichtet. Das Erscheinungsbild des späthistoristischen Gebäudes wird vor allem durch den orginalen Warteturm und dem später im Jahr 1935 hinzugefügtem Kassenanbau bestimmt. Auch dieses Gebäude erlitt wie auch schon die Radetzkybrücke schwere Kriegsschäden und wurde mehrmals saniert. Das heutige innere Erscheinungsbild wurde nach Ausbauplänen des Architekten Dimitris Manikas im Jahr 2003 verändert.

Der Hermannpark ist nach Emanuel Herrmann (1839 - 1902) dem Schöpfer der ersten Postkarte (1869) der Welt benannt. Im Jahr 1930 wurde an der näher der Flussmündung ein Park geschaffen indem das Anschwemmungsgebiet des Donaukanalufers erhöht wurde. Ein im Jahr 1981 oberirdisch geführte Fermwärmeletung wurde aus gestalterischen Gründen im Jahre 2004 wieder abgetragen durch eine Leitung in Tieflage ersetzt. „Eine weitgehende Neugestaltung erhielt das Gelände im Rahmen der Aufwertung des Donaukanals zu einem Freizeit- und Nahversorgungsraum mit der Zielsetzung die Menschen barrierefrei näher an das Wasser zu bringen. Kern dieses neuen Nutzungskonzeptes ist eine Strandbar mit Boule- bzw. Boccia-Spielplätzen, 2003 eröffnet. Mittelfristig wird es in diesem Bereich zur Verlängerung des Entlastungskanals des rechten Sammelkanals kommen, die allerdings in Tieflage geplant ist."[179]

Wettbewerbsaufgabe

Als Resultat der speziellen Anforderungen seitens des Entwurfortes wurde in den Wettbewerben eine klappbare und barrierefreie Fuß- und Radwegbrücke gefordert. Diese sollte für bis zu 25 Hübe pro Tag ausgelegt werden. Im günstigsten Fall beträgt die zu überbrückende Distanz zwischen den zwei Uferzonen in etwa 43 Meter. In diesem Fall befindet sich die Brücke über dem unterirdischen Dükersystem, welches unter dem Wienfluss von einem Ufer zum anderen geführt wird und auf welchem keine Gründung erlaubt wird. Weiters dürfen auch die Uferzonen nicht wesentlich verändert werden, da diese dem Denkmalschutz unterliegen. Bereits in der Einführungsveranstaltung der "Conrete Student Trophy" wurde besonders häufig auf die Problematik des historischen Umfeldes hingewiesen. Ebenso deuten die Beschreibungen des Entwurfortes in der Auschreibung des

Abbildung 68: Unmittelbare Entwurfsumgebung. Linker Bildrand - Herrmannpark; Linke Bildmitte - Radetzkybrücke; Rechte Bildmitte - Urania; Rechts neben der Urania - Aspernbrücke; Rechts - Uniqua Tower

Abbildung 69: Unmittelbare Entwurfsumgebung. Linker Bildrand - Herrmannpark; Linke Bildmitte - Radetzkybrücke; Rechte Bildmitte - Urania; Rechts neben der Urania - Aspernbrücke; Rechts - Uniqua Tower

Realisierungswettbewerbes "Connecting Link" auf die Wichtigkeit dieser Gegebenheiten hin.

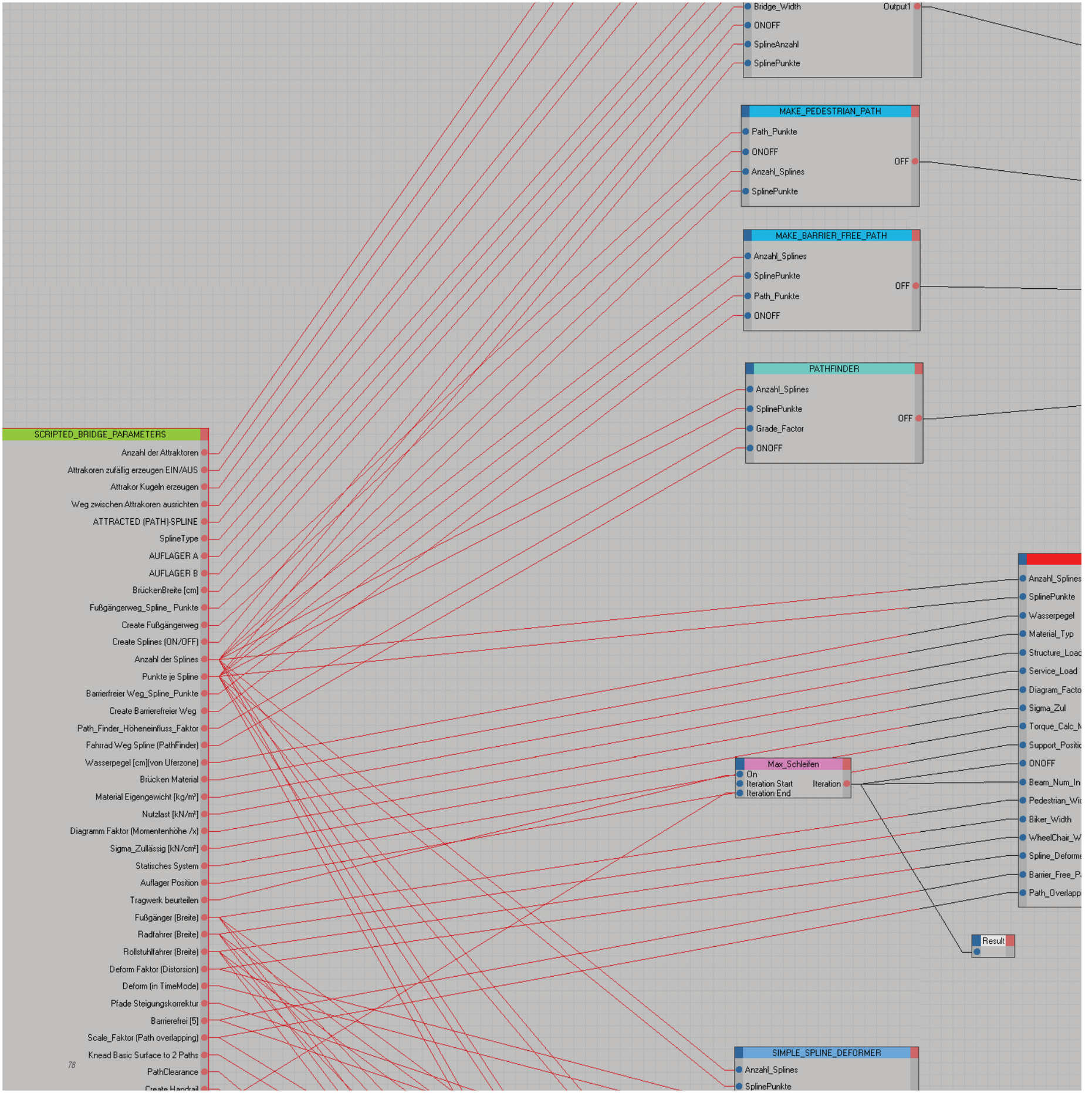
Abschließend werden exemplarisch die geforderten technischen Rahmenbedingungen der Concret Student Trophy aufgelisetet:

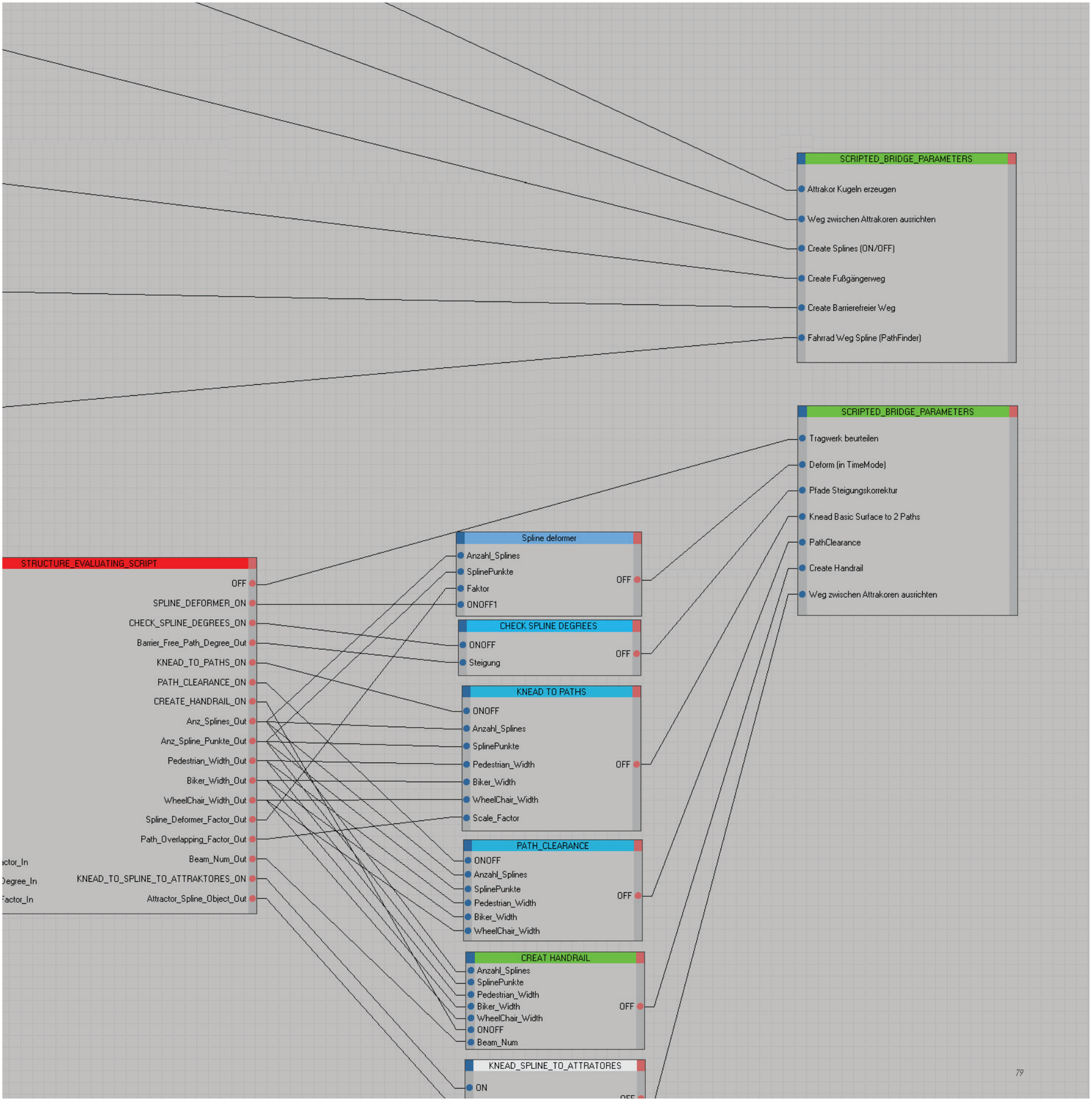
- Lastannahmen: EUROCODE 1 speziell für Fuß- und Radfahrstege: 500 kg/m2, kl. Schneeräumgerät, ev. kl. Einsatzfahrzeuge.
- Auf den Betrieb des Twin City Liners und damit auf die Wendemanöver der Schiffe soll durch die Ausformung als Klappbrücke Rücksicht genommen werden. Der Wendeplatz befindet sich im Donaukanal zwischen Aspernbrücke und Franzensbrücke. Große Schiffe manövrieren dabei in den Wienfluss bis zur Sohlschwelle knapp vor dem Pfeiler der Radetzkybrücke, in diesem Bereich ist auch die Lage der Fuß- und Radwegbrücke geplant. Für die Wendemanöver muss unbedingt der gesamte Mündungsbereich des Wienflusses zur Verfügung stehen (freie Durchfahrtsbreite 30 m, lichte Durchfahrtshöhe 8 m über höchstem Schifffahrtswasserstand). Eine Klappbrücke kann die erforderliche Manövrierfreiheit gewähren. Es ist je nach Fahrplan ist mit 5 bis 25 Hüben pro Tag zu rechnen.
- Hochwassersituationen sind für die Klappbarkeit der Brücke mit zu berücksichtigen. Aus Gründen des Personenschutzes wird es erforderlich sein den Zugang ab einem kritischen Wasserstand – vor Überflutung des Vorkais – zu verhindern. Da das Hochwasserregime des Wienflusses durch sehr rasch ansteigende Hochwasserwellen charakterisiert ist, ist ein Vorwarnsystem anzudenken, das auch bei Brückenklappung eingesetzt werden kann.
- Im Bereich des geplanten Standortes befindet sich der Dücker im Zuge des Rechten Hauptsammelkanal Entlastungskanals. Die Brückenfundierung ist aus Erhaltungs- und Haftungsgründen vom Kanal entkoppelt zu planen.
- Es ist darüber hinaus ein Tragwerk zu entwickeln, das den umliegenden historischen und unter Denkmalschutz stehenden Objekten wie Urania, Radetzkybrücke und Zollamtsteg Rechnung trägt. Der visuelle Gesamteindruck der Wienflussmündung mit dem tiefen Mauerprofil, den dahinter liegenden Brücken und Stegen soll möglichst wenig gestört werden und die offene Wasserfläche möglichst erhalten bleiben.
- Unterschiedliche Höhenangaben auf den Planunterlagen: „Wiener Null": Bezugspunkt für absolute Höhenangaben im Wiener Höhenkotensystem (156,680 Meter über Adria Null) im Gegensatz zum Meeresspiegel (in Landkarten) oder "Niveau" (als Ausgangspunkt für Bauklassen oder Bauhöhen). Das "Wiener Null" ist ein historisch vom Pegel der Ferdinandbrücke (heutige Schwedenbrücke) abgeleiteter Höhenbezug.

Anmerkung:
Die obige Auflistung wurde wörtlich übernommen und aus Layoutgründen nicht unter Anführungszeichen gesetzt. Weitere Informationen wie etwa Lagepläne, Ansichten etc. sind den zwei Fallbeispielen gegen Ende der vorliegenden Arbeit zu entnehmen.

178 Vgl. gesamtes Kapitel mit den Ausschreibungen der zwei Wettbewerbe:
"Concrete Student Trophy 2009", Vereinigung der Österreichischen Zementindustrie, http://www.zement.at.
"Connectiong Link", Magistrat der Stadt Wien Magistratsabteilung MA 29, http://www.wien.gv.at/verkehr/brueckenbau/index.html.
Bemerkung: Die Wettbewerbsunterlagen sind inzwischen nicht mehr öffentlich erhältich.

179 Vgl. Ausführungen in der Ausschreibung des Wettbewerbes "Connectiong Link".


Bridge_Width
ONOFF
SplineAnzahl
SplinePunkte
Output1
MAKE_PEDESTRIAN_PATH
Path_Punkte
ONOFF
Anzahl_Splines
SplinePunkte
OFF
MAKE_BARRIER_FREE_PATH
Anzahl_Splines
SplinePunkte
Path_Punkte
ONOFF
OFF
PATHFINDER
Anzahl_Splines
SplinePunkte
Grade_Factor
ONOFF
OFF
SCRIPTED_BRIDGE_PARAMETERS
Anzahl der Attraktoren
Attrakoren zufällig erzeugen EIN/AUS
Attrakor Kugeln erzeugen
Weg zwischen Attrakoren ausrichten
ATTRACTED (PATH)-SPLINE
SplineType
AUFLAGER A
AUFLAGER B
BrückenBreite [cm]
Fußgängerweg_Spline_ Punkte
Create Fußgängerweg
Create Splines (ON/OFF)
Anzahl der Splines
Punkte je Spline
Barrierfreier Weg_Spline_Punkte
Create Barrierefreier Weg
Path_Finder_Höheneinfluss_Faktor
Fahrrad Weg Spline (PathFinder)
Wasserpegel [cm](von Uferzone)
Brücken Material
Material Eigengewicht [kg/m²]
Nutzlast [kN/m²]
Diagramm Faktor (Momentenhöhe /x)
Sigma_Zullässig [kN/cm²]
Statisches System
Auflager Position
Tragwerk beurteilen
Fußgänger (Breite)
Radfahrer (Breite)
Rollstuhlfahrer (Breite)
Deform Faktor (Distorsion)
Deform (in TimeMode)
Pfade Steigungskorrektur
Barrierefrei [5]
Scale_Faktor (Path overlapping)
Knead Basic Surface to 2 Paths
PathClearance
Anzahl_Splines
SplinePunkte
Wasserpegel
Material_Typ
Structure_Load
Service_Load
Diagram_Facto
Sigma_Zul
Torque_Calc_M
Support_Positic
ONOFF
Beam_Num_In
Pedestrian_Wi
Biker_Width
WheelChair_W
Spline_Deforme
Barrier_Free_P
Path_Overlapp
Max_Schleifen
On
Iteration Start
Iteration End
Iteration
Result
SIMPLE_SPLINE_DEFORMER
Anzahl_Splines
SplinePunkte


SCRIPTED_BRIDGE_PARAMETERS
Attrakor Kugeln erzeugen
Weg zwischen Attrakoren ausrichten
Create Splines (ON/OFF)
Create Fußgängerweg
Create Barrierefreier Weg
Fahrrad Weg Spline (PathFinder)
SCRIPTED_BRIDGE_PARAMETERS
Tragwerk beurteilen
Deform (in TimeMode)
Pfade Steigungskorrektur
Knead Basic Surface to 2 Paths
PathClearance
Create Handrail
Weg zwischen Attrakoren ausrichten
Spline deformer
Anzahl_Splines
SplinePunkte
Faktor
ONOFF1
OFF
CHECK SPLINE DEGREES
ONOFF
Steigung
OFF
KNEAD TO PATHS
ONOFF
Anzahl_Splines
SplinePunkte
Pedestrian_Width
Biker_Width
WheelChair_Width
Scale_Factor
OFF
PATH_CLEARANCE
ONOFF
Anzahl_Splines
SplinePunkte
Pedestrian_Width
Biker_Width
WheelChair_Width
OFF
CREAT HANDRAIL
Anzahl_Splines
SplinePunkte
Pedestrian_Width
Biker_Width
WheelChair_Width
ONOFF
Beam_Num
OFF
KNEAD_SPLINE_TO_ATTRATORES
ON
STRUCTURE_EVALUATING_SCRIPT
OFF
SPLINE_DEFORMER_ON
CHECK_SPLINE_DEGREES_ON
Barrier_Free_Path_Degree_Out
KNEAD_TO_PATHS_ON
PATH_CLEARANCE_ON
CREATE_HANDRAIL_ON
Anz_Splines_Out
Anz_Spline_Punkte_Out
Pedestrian_Width_Out
Biker_Width_Out
WheelChair_Width_Out
Spline_Deformer_Factor_Out
Path_Overlapping_Factor_Out
Beam_Num_Out
KNEAD_TO_SPLINE_TO_ATTRAKTORES_ON
Attractor_Spline_Object_Out

Brückengenerierungsprozess

Entwurfskonzept

Zwischen den Uferzonen sollen drei Wege generiert werden können, welche die Grundgeometrie der Brücke definieren. Für diese Geometrie soll ein statisches System gewählt werden können, mittels dem eine automatische Vorbemessung durchführbar sein soll. Bei dieser Vorbemessung soll ein Kubatur- und Querschnittsvorschlag für die aktuelle Brückenfläche gemacht werden können, wobei weiters signifikante Bemessungsdaten wie der Momentenverlauf über das System, das Gesamtgewicht der Brücke, die Schwerpunktsposition sowie die Durchbiegung der Konstruktion etc. für den Entwerfer textlich sowie auch grafisch ausgegeben werden können sollen.

Diese drei genannten Wege beeinflussen über ihre Breiten und jeweiligen Positionierungen untereinander die Form des Brückenquerschnittes wesentlich mit und haben dadurch auch Einfluss auf die Tragwirkung bzw. auf das Flächenträgheitsmoment. Somit können die Wege über das Flächenträgheitsmoment direkt miteinander in Beziehung gebracht werden. Aus diesen Gründen sollen nach der Tragwerksbeurteilung bei automatischen Verbesserungsversuchen auch die Wegeverläufe und damit die gesamte Brückenform markant geändert werden können. Dies bedeutet, dass Querschnitte nicht ausschließlich durch Aushöhlen oder Höhenveränderung von der bestehenden Oberfläche ausgehend vorgenommen werden können sollten, sondern auch durch Verändern der Wegeführungen untereinander.

Pfade

Neben einer jeweils einzuhaltenden Weglichte und maximal zulässigen Steigungen besitzen die Pfade noch weitere Qualitäten, die in weiteren Modulentwicklungen berücksichtigt werden sollen.

Fußgängerweg

Der Fußgänger benützt die Brücke am häufigsten und verharrt im Vergleich zu den anderen Nutzern tendenziell am längsten in dieser Umgebung. Daher soll der Verlauf des Weges durch Ortseinflüsse geprägt werden. Dies beutet, dass lokale Qualitäten bewertet werden können sollen und der Weg aufgrund dieser Bewertung verzerrt werden können soll.

Barrierefreier Weg

Der barrierefreie Weg unterliegt einer starken Steigungseinschränkung und hat diesbezüglich gegenüber den anderen Wegen höchste Priorität.

Radfahrweg

Da der Radfahrer meist schnell und direkt die Brücke überqueren will, weil er ohnehin nicht stehenbleiben kann, ist der Verlauf durch eine möglichst kurze und gerade Verbindung zwischen den Brückenenden geprägt. Es soll möglich sein, diesen Weg automatisch generieren zu können.

Nach diesem Entwurfskonzept soll nun der gesamte Brückenentwurfsprozess mit den einzelnen Modulen entwickelt werden. Es sei jedoch bemerkt, dass bewusst versucht wurde, die Module, die aus diesem Konzept resultieren, wenn möglich flexibel zu konzipieren, damit diese auch für andere als die ursprünglich vorgesehenen Zwecke verwendet werden können. Wie dieses Entwurfskonzept im Detail umgesetzt wird, ist in der Beschreibung der einzelnen Module im nächsten Kapitel oder gleich im jeweiligen Script im Appendix nachzulesen.

Veränderungsmöglichkeiten des Brückenquerschnittes

Dem Tragwerksbeurteilungmodul sollen Veränderungsmöglichkeiten zur Verfügung stehen, um die Tragfähigkeit der Brücke zu verbessern. Wird in der Berechnung des Vorentwurfes ein ungünstiges Tragverhalten der aktuellen Brücke festgestellt, sollen vom Modul automatisch Verbesserungsvorschläge gemacht werden. Dabei soll auf vier unterschiedliche Veränderungsmöglichkeiten zurückgegriffen werden können.

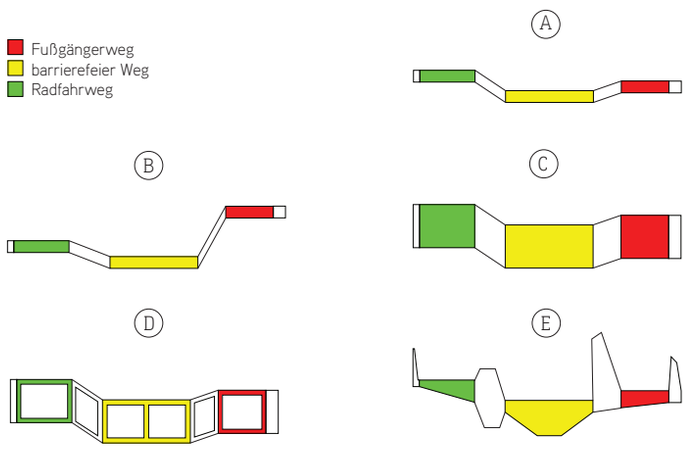
A: Skizze "A" stellt den Querschnitt beispielsweise nach dem ersten Vorbemessungsversuch dar.

B: Als erste Möglichkeit stellt sich Skizze "B" dar, in der eine Verbesserung des Widerstandmomentes des gesamten Querschnittes durch Höhenveränderung der Wege untereinander zu erreichen versucht wird.

C: Durch Variation der Höhe des Trägers kann ebenfalls das Tragverhalten günstig beinflusst werden.

D: Durch das Aushöhlen des Trägers wird die gesamte Konstruktion leichter, wobei bei einer geschickten Vorgehensweise das Flächenwiderstandsmoment realtiv hoch bleibt.

E: Generell ist es möglich, für eine Verbesserung der Tragleistung alle Oberflächenpunkte des Brückenprofils zu manipulieren. Es muss lediglich die Begehbarkeit der Brücke garantiert werden.



Ablaufskizze

Aus dem vorhergehenden Entwurfskonzept folgt zunächst eine Überlegung bezüglich eines möglichen Entwurfsvorganges. Dabei soll sichtbar gemacht werden, welche Freiheiten während des Entwerfens in diesem Prozess herrschen sollen beziehungsweise können. In jedem der folgenden Schritte können einige Parameter geändert werden (z.B. Weglichten, Maximalsteigungen, Grad der Aushöhlung der Querschnitte, Nutzlasten, Betongewicht).

- Es werden zwei Pfade generiert, welche die Kontur der Brücke definieren. Diese Pfade werden manuell nach einer architektonischen Konzeptvorstellung geformt. Zwischen den zwei Pfaden wird eine Fläche aufgespannt, woduch eine erste Brückenoberfläche entsteht.

- Nun wird ein dritter Weg automatisch generiert. Dieser Weg ist der kürzeste Weg von einem Brückenende über die aktuelle Oberfläche zum anderen.

- Da die einzelnen Wege laut Normen jeweils maximal zulässige Höhenentwicklungen nicht überscheiten dürfen, wird in diesem Schritt kontrolliert, ob die zulässigen Maximalsteigungen der einzelnen Wege über die Brücke eingehalten wurden. An den Stellen, wo die zulässigen Steigungswinkel überschritten werden, werden die Wege geglättet.

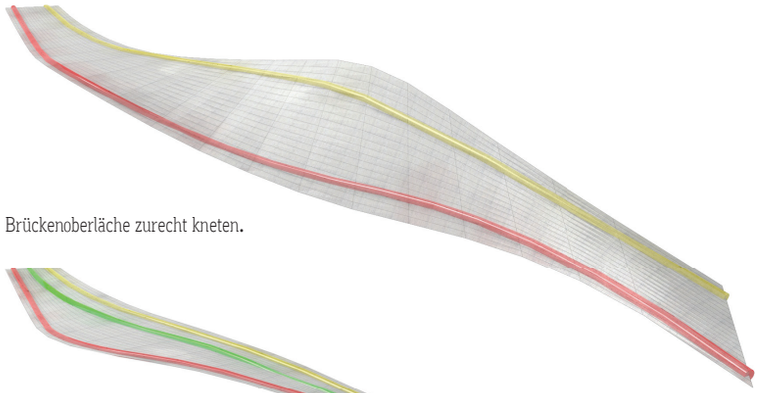
- Damit die Änderungen von einzelnen Pfaden bzw. Wegen auch Veränderungen an der aktuellen Brückengeometrie verursachen, muss eine Art "Zurechtkneten-Befehl" ausgeführt werden, der die Oberfläche den Wegeführungen anpasst.

- Damit die Wege je nach Nutzung nutzbar zu machen, wird die Brückenoberfläche in den Weglichten-Bereichen eingeebnet.

- Schließlich wird eine automatische Vordimensionierung des Brückentragwerks vorgenommen. Da das Eigengewicht der Brücke für die Dimensionierung maßgeblich ist, wurde ein Algorithmus programmiert, der die Brücke nach bestimmten Bedingungen aushöhlt und genügend Obergurt-, Untergurthöhen sowie Steganzahl und Stegbreiten im Querschnittsinneren übrig lässt.

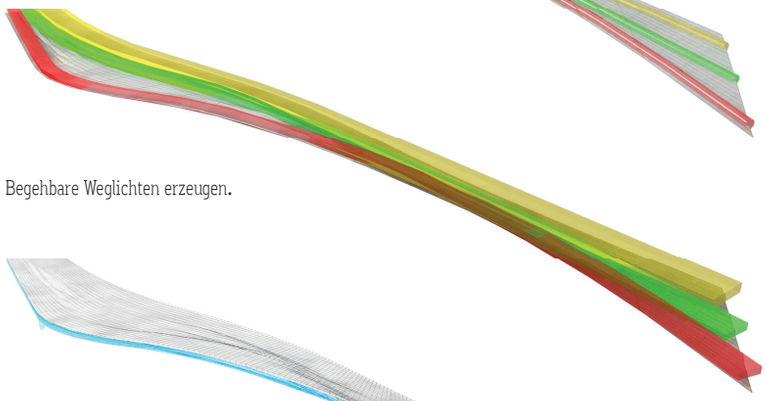
- Das Brückengeländer kann am Ende automatisch passend zur Brücke generiert werden. Hier lässt sich eine Mindestgeländerhöhe einstellen.

Wenn einzelne Ergebnisse nicht der gewünschten Ästhetik entsprächen, kann der Prozess an beliebiger Stelle neugestartet werden. Nahezu alle Schritte können untereinander ausgetauscht oder gar nicht ausgeführt werden, woraus sich ein nicht-linearer Entwurfsprozess ergibt.



Brückenoberläche zurecht kneten.

Steigungen korrigieren und dritten Weg generieren.

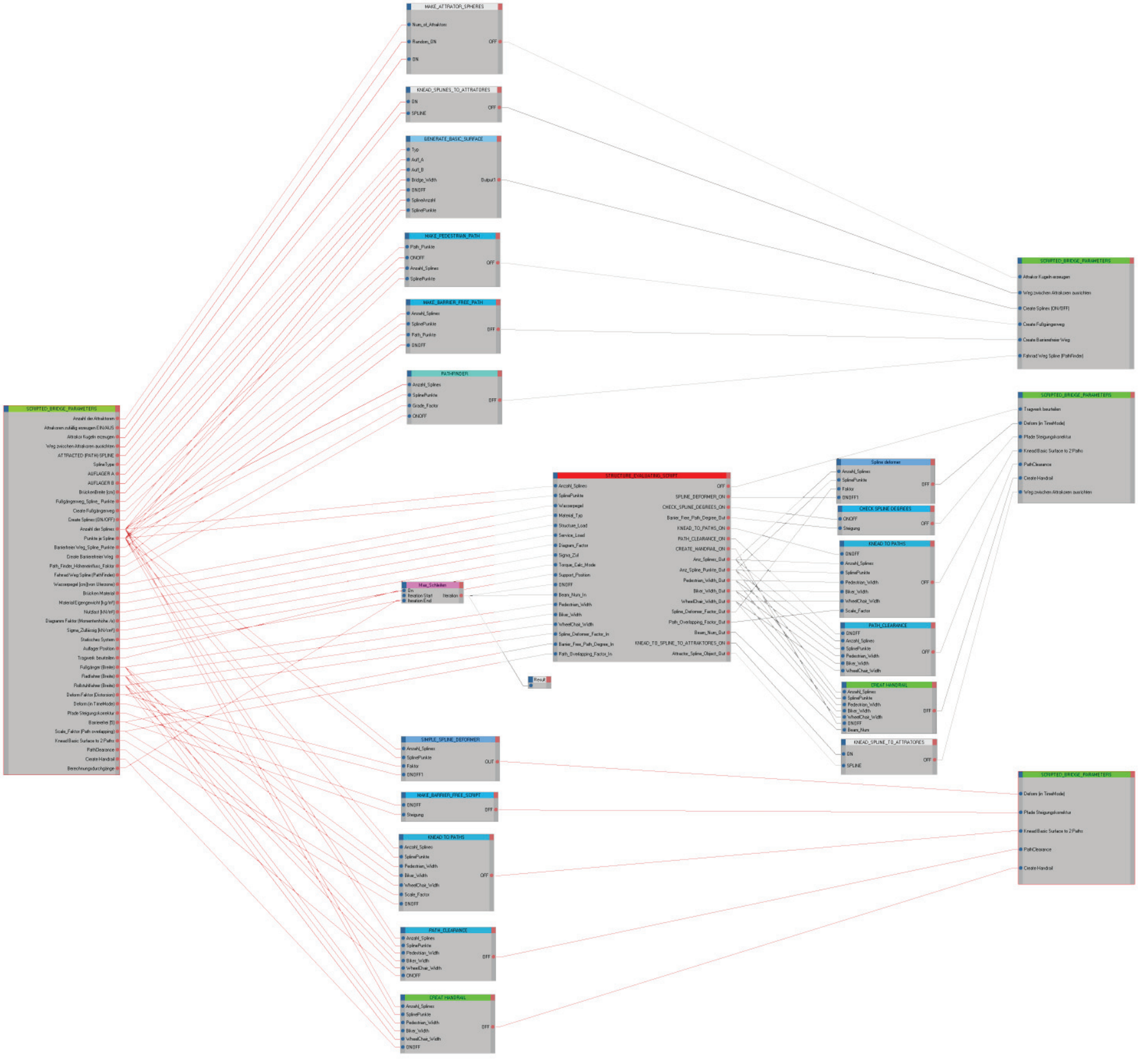


Begehbare Weglichten erzeugen.

Tragwerksvorbemessung und Kubaturvorschlag.

Aufgrund des hohen Eigengewichtes muss die Konstruktion
unter Berücksichtigung konstruktiver Aspekte ausgehöhlt werden

Nach dieser Skizze werden folgend die einzelnen Module enwickelt und zu einer Gesamtschaltung vernetzt. Über dieses Netzwerk von Einzelfunktionen wird der Rahmen der Entwurfsmöglichkeiten definiert, innerhalb dessen mehrere Varianten von ähnlichen Brücken rasch generiert werden können.


MAKE_ATTRATOR_SPHERES
Num_of_Attraktors
Random_ON
ON
OFF
KNEAD_SPLINES_TO_ATTRATORES
ON
SPLINE
OFF
GENERATE_BASIC_SURFACE
Typ
Auf_A
Auf_B
Bridge_Width
ON/OFF
SplineAnzahl
SplinePunkte
Output1
MAKE_PEDESTRIAN_PATH
Path_Punkte
ON/OFF
Anzahl_Splines
SplinePunkte
OFF
MAKE_BARRIER_FREE_PATH
Anzahl_Splines
SplinePunkte
Path_Punkte
ON/OFF
OFF
PATHFINDER
Anzahl_Splines
SplinePunkte
Grade_Factor
ON/OFF
OFF
SCRIPTED_BRIDGE_PARAMETERS
Anzahl der Attraktoren
Attraktoren zufällig erzeugen EIN/AUS
Attraktor Kugeln erzeugen
Weg zwischen Attraktoren ausrichten
ATTRACTED (PATH) SPLINE
SplineType
AUFLAGER A
AUFLAGER B
BrückenBreite [cm]
Fußgängerweg_Spline_ Punkte
Create Fußgängerweg
Create Splines (ON/OFF)
Anzahl der Splines
Punkte je Spline
BarrierfreierWeg_Spline_Punkte
Create BarrierefreierWeg
Path_Finder_Höhenverlfluss_Faktor
Fahrrad Weg Spline (PathFinder)
Wasserpegel [cm] von Ulterzone)
Brücken Material
Material Eigengewicht [kg/m²]
Nutzlast [kN/m²]
Diagramm Faktor (Momentenhöhe /x)
Sigma_Zulässig [kN/cm²]
Statisches System
Auflager Position
Tragwerk beurteilen
Fußgänger (Breite)
Radfahrer (Breite)
Rollstuhlfahrer (Breite)
Deform Faktor (Dotorsion)
Deform (in TimeMode)
Pfade Steigungskorrektur
Barrierefrei [0]
Scale_Faktor (Path overlapping)
Knead Basic Surface to 2 Paths
PathClearance
Create Handrail
Berechnungsdurchgänge
SCRIPTED_BRIDGE_PARAMETERS
Attraktor Kugeln erzeugen
Weg zwischen Attraktoren ausrichten
Create Splines (ON/OFF)
Create Fußgängerweg
Create Barrierefreier Weg
Fahrrad Weg Spline (PathFinder)
SCRIPTED_BRIDGE_PARAMETERS
Tragwerk beurteilen
Deform (in TimeMode)
Pfade Steigungskorrektur
Knead Basic Surface to 2 Paths
PathClearance
Create Handrail
Weg zwischen Attraktoren ausrichten
STRUCTURE_EVALUATING_SCRIPT
AnzahL_Splines
SplinePunkte
Wasserpegel
Material_Typ
Structure_Load
Service_Load
Diagram_Factor
Sigma_Zul
Torque_Calc_Mode
Support_Position
ON/OFF
Beam_Num_In
Pedestrian_Width
Biker_Width
WheelChair_Width
Spline_Deformer_Factor_In
Barrier_Free_Path_Degree_In
Path_Overlapping_Factor_In
OFF
SPLINE_DEFORMER_ON
CHECK_SPLINE_DEGREES_ON
Barrier_Free_Path_Degree_Out
KNEAD_TO_PATHS_ON
PATH_CLEARANCE_ON
CREATE_HANDRAIL_ON
Anz_Splines_Out
Anz_Spline_Punkte_Out
Pedestrian_Width_Out
Biker_Width_Out
WheelChair_Width_Out
Spline_Deformer_Factor_Out
Path_Overlapping_Factor_Out
Beam_Num_Out
KNEAD_TO_SPLINE_TO_ATTRAKTORES_ON
Attractor_Spline_Object_Out
Max_Schleifen
Bn
Iteration Start
Iteration End
Iteration
Result
Spline deformer
AnzahL_Splines
SplinePunkte
Faktor
ON/OFF1
OFF
CHECK_SPLINE DEGREES
ON/OFF
Steigung
OFF
KNEAD TO PATHS
ON/OFF
AnzahL_Splines
SplinePunkte
Pedestrian_Width
Biker_Width
WheelChair_Width
Scale_Factor
OFF
PATH_CLEARANCE
ON/OFF
AnzahL_Splines
SplinePunkte
Pedestrian_Width
Biker_Width
WheelChair_Width
OFF
CREAT HANDRAIL
AnzahL_Splines
SplinePunkte
Pedestrian_Width
Biker_Width
WheelChair_Width
ON/OFF
Beam_Num
OFF
KNEAD_SPLINE_TO_ATTRATORIES
ON
SPLINE
OFF
SIMPLE_SPLINE_DEFORMER
AnzahL_Splines
SplinePunkte
Faktor
ON/OFF1
OUT
MAKE_BARRIER_FREE_SCRIPT
ON/OFF
Steigung
OFF
KNEAD TO PATHS
AnzahL_Splines
SplinePunkte
Pedestrian_Width
Biker_Width
WheelChair_Width
Scale_Factor
ON/OFF
OFF
PATH_CLEARANCE
AnzahL_Splines
SplinePunkte
Pedestrian_Width
Biker_Width
WheelChair_Width
ON/OFF
OFF
CREAT HANDRAIL
AnzahL_Splines
SplinePunkte
Pedestrian_Width
Biker_Width
WheelChair_Width
ON/OFF
OFF
SCRIPTED_BRIDGE_PARAMETERS
Deform (in TimeMode)
Pfade Steigungskorrektur
Knead Basic Surface to 2 Paths
PathClearance
Create Handrail

Mashup-System

Allgemeines

Nach dem im vorhergehenden Kapitel genannten Entwurfskonzept bzw. den formulierten Forderungen an das Entwurfstool wurden nun einzelne Funktionen entwickelt. Diese einzelnen Funktionen werden ähnlich wie in einem elektrischen Schaltplan in kleinen "Blackboxes" organisiert, die gängigerweise als Funktionsmodule, -boxen, -knoten oder auch als Nodes bezeichnet werden (mehr dazu siehe S.35-37). Die Nodes können über eine Vernetzung miteinander interagieren und bilden zusammen eine Schaltung. Die Schaltung spiegelt nun den gesamten parametrisierten Entwurfsprozess wider. Das gesamte Konzept wurde in der Animationssoftware Cinema 4D mittels der internen Pluggins mit "Xpresso" und die Scripte mit "Coffee" umgesetzt. Alle Entwicklungen werden in dem sogenannten "Xpresso-Tag" gespeichert. Diese wird als kleines Icon dargestellt und kann ähnlich wie ein 3d-Objekt kopiert und in die Gesamtszene eines anderen Entwurfsprojektes eingefügt werden. Damit kann dieser parametrisierte Prozess wie ein Plugin in anderen Projekten verwendet werden (siehe Beispiel S.110-111).

Im Folgenden werden zunächst die grafische Eingabemöglichkeit und danach die einzelnen Nodes näher erklärt. Die Scripte, die sich hinter den einzelnen Funktionsboxen befinden, werden im Appendix dargestellt.

Bemerkungen

Die Vernetzung der einzelnen Nodes erfolgt über ihre Ein- und Ausgänge, die in der Schaltung mittels blauen und roten Kreisen dargestellt werden.

Damit die Funktionen der Boxen nicht ständig unkontrolliert automatisch ausgeführt werden, wurden in dieser Arbeit alle Nodes mit "ON/OFF" Eingang ausgestattet. Da dieser Ausführungsbefehl etwa über die grafische Benutzeroberfläche durch Aktivieren (enable) eines "Kästchens" gegeben wird, wurden die einzelnen Module mit OFF-Ausgängen versehen, um damit nach Ausführen der Funktion diese Kästchen wieder automatisch zu deaktivieren (disable).

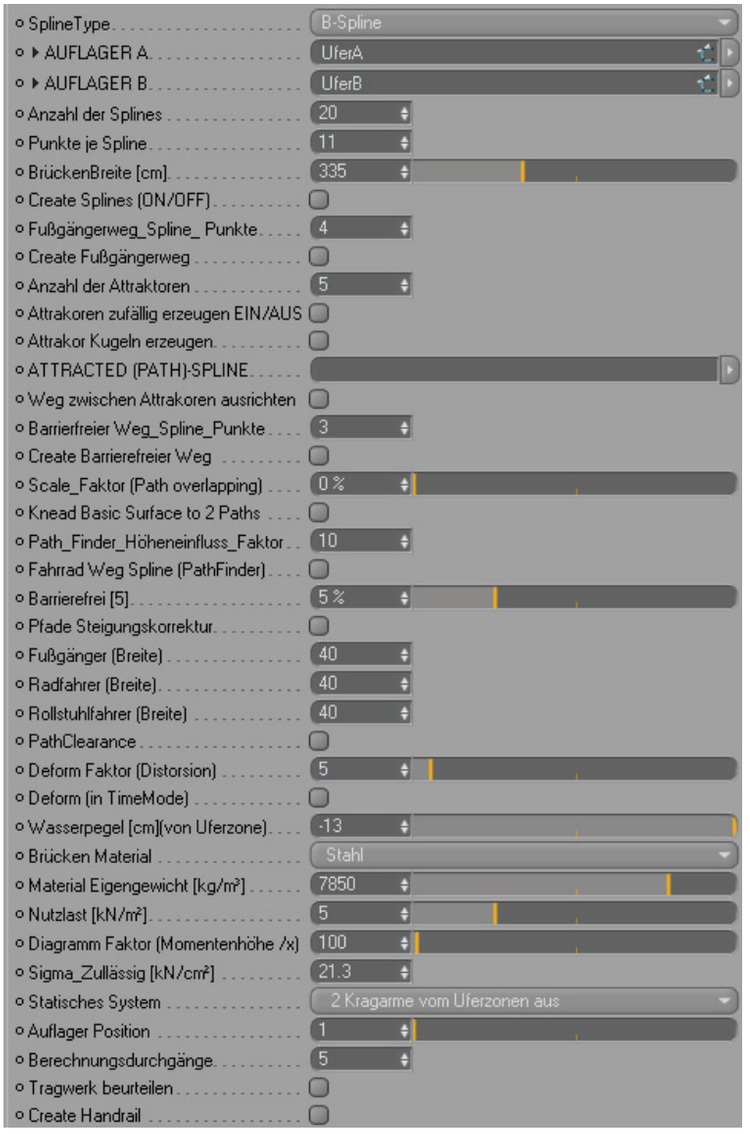
Die Ein- und Ausgänge besitzen unterschiedliche Grundeigenschaften. Daher muss beim Hinzufügen eines Ein- oder Ausgangs entschieden werden, ob beispielsweise eine Zahl, ein String, ein ganzes Objekt etc. aus- oder eingegeben werden soll.

Da viele dieser Module relativ am Anfang der Entwicklung programmiert wurden und noch nicht genau bekannt war, wie man genauere Objektdetails ausliest, wurden die Module mit Eingängen ausgestattet, über die immer die Splineanzahl und Splinepunkte der Grundmatrix übergeben werden. Diese Eingänge könnten bei einer Überarbeitung wegfallen.

Die Abbildung links zeigt die fertige Xpresso-Schaltung.

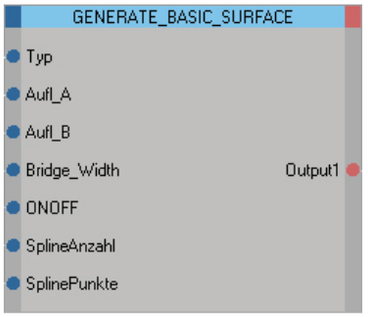
User Interface - Externe Parameter

Da es Ziel war, zwar einen flexiblen Prozess über Parametersteuerung zu gestalten, wurde während der Entwicklung der Module bzw. dem Schreiben der einzelnen Scripte generell darauf geachtet, dass möglichst viele Funktionen und Abläufe mit allgemeinen Variablen beschrieben werden anstatt jeweils fixe Werte zu vergeben. Dabei enstand eine große Menge nachträglich verstellbarer Variablen, wobei einige davon dem Entwerfer als Parameter direkt über eine grafische Eingabeoberfläche - dem "User-Interface" - zugänglich gemacht wurden. Dabei wurde darauf geachtet, dass vorerst nur die notwendigsten Paramter im Auswahl-Menü zur Verfügung gestellt werden, da dieses ansonsten sehr komplex wird und das ursprüngliche Ziel einer leichten intuitiven Benutzbarkeit verloren gehen würde. Natürlich lassen sich die Menüpunkte ohne großen Aufwand verändern. Diese Auswahl-Menü bzw. User-Interface ist direkt über einen Funktionsnode (Grün) mit den anderen Nodes verbunden.



```
//SURFACE_CALC
BRIDGE_SURFACE_CALC(); //ERGEBNIS: EntirePlane,SinglePlane_BalancePoint[cnt][a],SinglePlanes[cnt][a], Sector_Plane[a]

//Vom tiefsten Uferpunkt das Wasserlevel abziehn
WATERLEVEL_POINT_CALC();

Beam_Number_Intern=Beam_Num_In;


if(Beam_Num_In<=1) //Externe BeamNummer
{
  println("Es werden die Daten für den IDEALTRÄGER erzeugt! BeamNr.: ",Beam_Num_In,"!");
  println("Minimalhöhe vergeben!");

   if(Material_Typ==0) Formula_Divident_Arm_1=20; //Stahl
   if(Material_Typ==0) Formula_Divident_Arm_2=20; //Stahl
   if(Material_Typ==1) Formula_Divident_Arm_1=16; //Beton
   if(Material_Typ==1) Formula_Divident_Arm_2=16; //Beton


  //Volumshöhenpunkte zuweisen -> Für die Erste Volumsberechnung - Momentenverlauf
  for (a=0;a<SplinePunkte;a++)
  {
    for (cnt=0;cnt<Anzahl_Splines;cnt++)
    {
      BRIDGE_VOLUME_POINTS_ARRAY[cnt][a]=BRIDGE_POINTS_ARRAY[cnt][a];
      BRIDGE_VOLUME_POINTS_ARRAY[(Anzahl_Splines*2-1)-cnt][a]=BRIDGE_POINTS_ARRAY[cnt][a];
      BRIDGE_VOLUME_POINTS_ARRAY[(Anzahl_Splines*2-1)-cnt][a].y=BRIDGE_VOLUME_POINTS_ARRAY[(Anzahl_Splines*2-1)-cnt][a].y-Minimal_Construction_Height;
    }
  }

  //START DER BERECHNUNG DES IDEALTRÄGERS -> 1.Vorbemessung
  //-----------------------------------------------------------------

   //Profiltypen pro Segmente vergeben
  DEFINE_PROFILETYPES();

  //Berechnung der Einzel_Volumen und des Gesamten Trägers
  BRIDGE_VOLUME_CALC();

  //LOADCALC
  LOAD_CALC(); //Achtung Load stimmt nicht da konstante Höhenannahme -> Minimal_Construction_Height

  //TORQUE_CALC
  TORQUE_CALC(); //Achtung Torque stimmt nicht da konstante Höhenannahme -> Minimal_Construction_Height

  //IDEAL_BEAM_CALC
  Section_Shape_Mode=0; //Rechtecksquerschnitt;
  Faktor_1_Enable==False;//Für Idealträger wird dieser Faktor ausgeschalten -> Sollte aber eigentlich eingeschalten werden!!!
  Torque_Curve_Calc=True;
  BEAM_CALC(Beam_Number_Intern);
  //Section_Shape_Mode=1; //Konkaver Querschnitt

} //if(Beam_Num_In<=1) ENDE

//-----------------------------------------------------------------
//IDEALTRÄGER WURDE BERECHNET -> ES WURDE EIN ERSTES VOLUMEN VERGEBEN -> BRIDGE_VOLUME_POINTS_ARRAY
//-----------------------------------------------------------------
println("-------------------------------------------------------------------------------------");

//-------------------------------------------------------------------------------------------------------------
//ACHTUNG: DER VORHERGENDE TEIL KÖNNTE ÜBERSPRUNGEN WORDEN SEIN -> ES WIRD EIN AKTUELLES VOLUMEN BENÖTIGT

if(Beam_Number_Intern>1)
{ Beam_Number_Intern=Beam_Number_Intern-1; //Da die vorhergehende Geometrie eingelesen werden muss
  println("Es wird der vorhergende Träger Nummer: ",Beam_Num_In," eingelesen!");

  //VERÄNDERTE VARIABLEN EINLESEN
  READ_VARIABLE_FOLDER();

  //DAS BRIDGE_VOLUME_POINTS_ARRAY BERECHNEN
  READ_BRIDGE_SHAPE();
  Beam_Number_Intern=Beam_Number_Intern+1;
  //-----------------------------------------------------------------------
  //VORHER ERZEUGTES TRÄGERVOLUMEN WURDE EINGELESEN -> Soll nun beurteilt werden!!!
  //-----------------------------------------------------------------------
  //CHANGE Bridge Shape;
  CHECK_AND_CHANGE_BRIDGE_SHAPE();

  //Profiltypen pro Segmente vergeben
  DEFINE_PROFILETYPES(); //Wenn die Beam_Number_Intern<=2 dann wird ein Vollquerschnitt berechnent
  //Externe Daten den Internen übergeben
  Formula_Divident_Arm_1=Formula_Divident_Arm_1_Extern;
  Formula_Divident_Arm_2=Formula_Divident_Arm_2_Extern;
  if(Torque_Curve_Calc_Mode==1) Torque_Curve_Calc=True;
  if(Torque_Curve_Calc_Mode==0) Torque_Curve_Calc=False;
  if(Faktor_1_Enable_Extern==0) Faktor_1_Enable=False;
  if(Faktor_1_Enable_Extern==1) Faktor_1_Enable=True;
  if(Faktor_2_Enable_Extern==0) Faktor_2_Enable=False;
  if(Faktor_2_Enable_Extern==1) Faktor_2_Enable=True;
```

Beschreibung der einzelnen Nodes



Generate_Basic_Surface[]

Allgemeine Beschreibung

Mit diesem Modul kann eine Loft_Oberfläche mit einer bestimmten Grundauflösung bzw. Punktmatrix zwischen zwei beliebigen dreidimensionalen Splines im Entwurfsraum aufgespannt werden.

Motivation

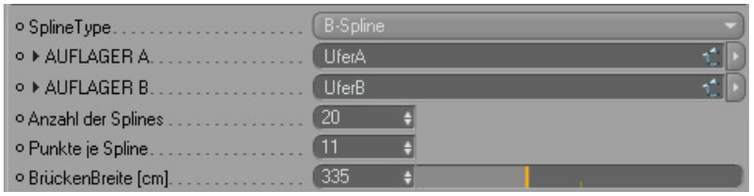
Der Grundgedanke, anfangs eine solche Matrix bzw. in diesem Fall Brückenoberfläche definieren zu müssen, war die Annahme, dass eine Brücke sich immer über eine Längsausdehnung von einem Bereich" A" nach "B" entwickelt. Diese Zonen können als Auflager, Uferzonen oder allgemein als Brückenenden verstanden werden. Die Punkte an den Brückenenden können zwar manuell verändert werden, jedoch werden diese im Regelfall von keinem der anderen Module automatisch verzerrt. Diese Fläche wird als Grundmatrix verstanden, mit der alle anderen Module interagieren können, indem sie aktuelle Punktkoordinaten auslesen oder auch verändern können.

Im Entwurf

Grundsätzlich bestimmt diese Fläche die Oberflächengeometrie der Brücke. Während des Entwerfens kann diese Fläche als Knetfläche verstanden werden, die mit verschiedensten Modulen verändert werden kann. Diese Funktion ist die einzige, die als Grundvoraussetzung vor allen anderen Funktionen ausgeführt werden muss, da ohne Knetfläche kein Kneten bzw. Entwerfen möglich ist.

Detail

Die Loftoberfläche wird nicht direkt über die zwei genannten Splines (Rot) bestimmt, sondern über eine definierbare Anzahl von Splines (Grün) erzeugt. Diese (Grün) werden zwischen zwei Splines (Rot) generiert und können selbst jeweils durch eine bestimmte Anzahl von Kontrollpunkten (Gelb) unterteilt und beschrieben werden. Dabei lassen sich Splinetypen wie etwa lineare, cubische, B-Spine, Akima auswählen. Die Splines (Grün) werden untereinander in einem gleichmäßigen Abstand angeordnet, wobei deren Distanz untereinander mittels einer Gesamtbreiteneingabe festgelegt werden kann. Dabei werden die Abstände der Splines (Grün) nicht auf die Normalen untereinander, sondern jeweils entlang der zwei Grundsplines (Rot) gemessen. Sind diese zwei Grundsplines beispielsweise länger als die eingegeben Breiten, dann wird pro Spline (bsp. Uferzone) eine zufällige Positionierung der Splines gewählt, wobei die Splineabstände untereinander an den Anfangs- und Endpunkten konstant bleiben.



Bemerkung

Die Verwendung einer solchen Grundmatrix hat sich als geschickt herausgestellt. Es wird versucht, im gesamten Prozess eine möglichst gleichmäßige Rasterung zu erhalten, da die Knetfläche so etwa auch zur Definition von signifikanten Geometriepunkten oder auch zur Festlegung von Schnittlinienführung etc. dienen kann. Durch diese Methode lässt sich die Brückengeometrie grundsätzlich mit einer klaren übersichtlichen regelmäßigen Anzahl von Punkten beschreiben.

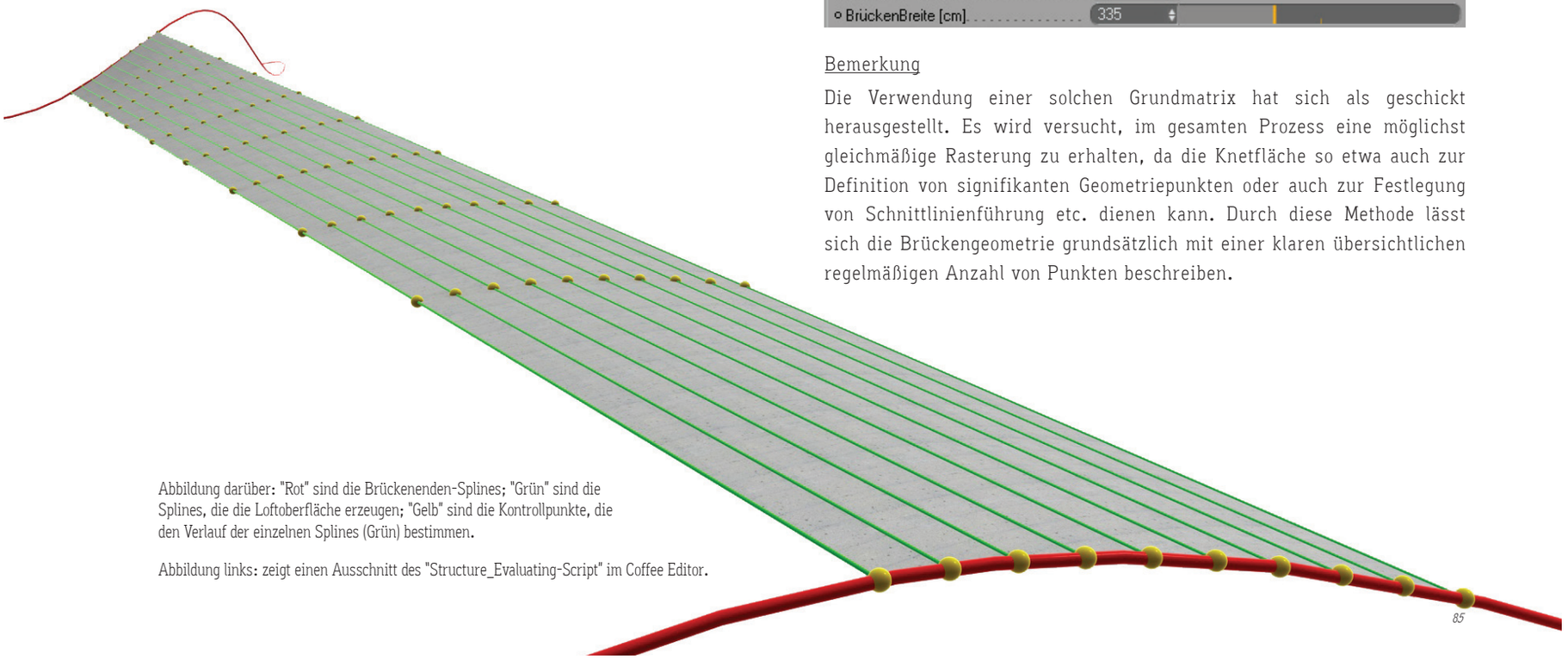
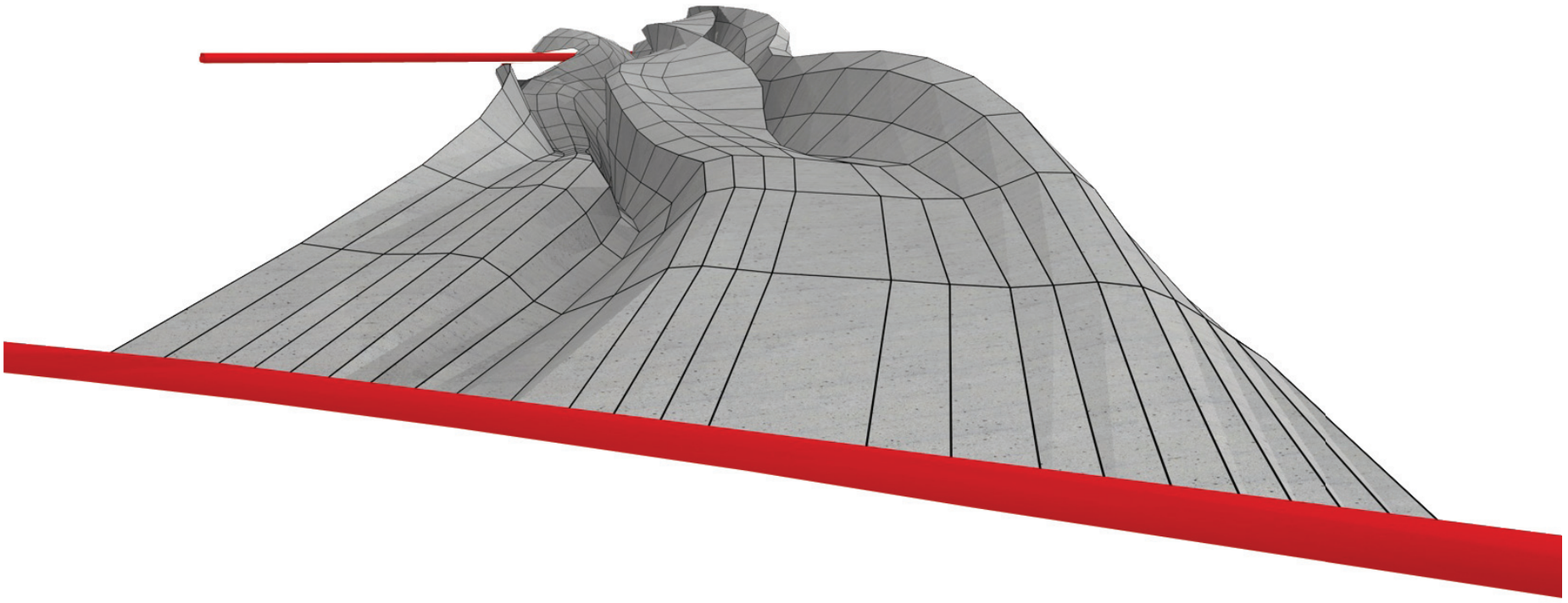
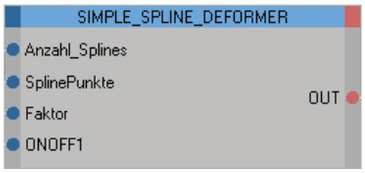


Abbildung darüber: "Rot" sind die Brückenenden-Splines; "Grün" sind die Splines, die die Loftoberfläche erzeugen; "Gelb" sind die Kontrollpunkte, die den Verlauf der einzelnen Splines (Grün) bestimmen.

Abbildung links: zeigt einen Ausschnitt des "Structure_Evaluating-Script" im Coffee Editor.



Simple_Spline_Deformer[]

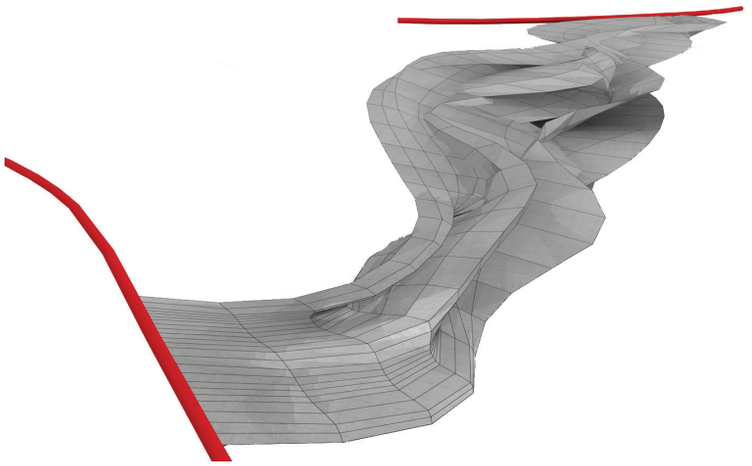


Allgemeine Beschreibung

Mittels dieser Funktionsbox kann eine bestimmte Anzahl von Splines mittels Zufallsgenerator verzerrt werden, wobei die Stärke der Verzerrung der einzelnen Kontrollpunkte der Splines im Raum über einen Faktor bestimmt werden kann.

Motivation

Dieses Script wurde geschrieben, um andere Module schneller testen bzw. überprüfen zu können. Aus diesem Grund kann die manuelle Verzerrung einzelner Splinekontrollpunkte durch eine automatisierte ersetzt werden.



Im Entwurf

Dieses Modul ist nicht zur Verwendung im eigentlichen Enwurfsprozess gedacht, da es zum Testen von verschiedensten Funktionen entwickelt wurde.

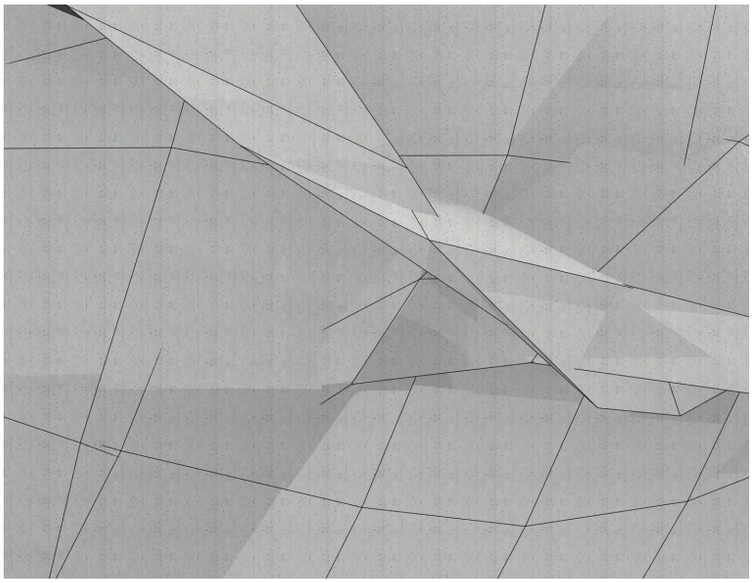
Detail

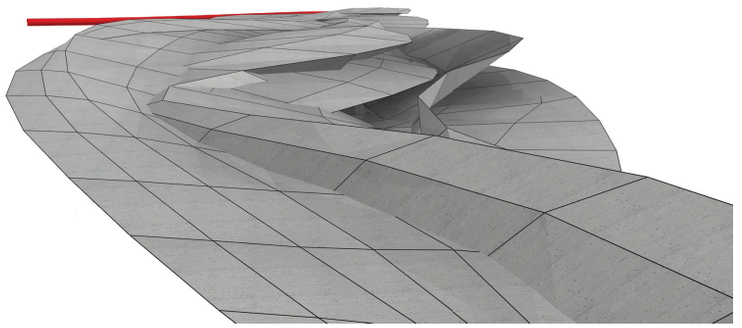
Die zu verzerrenden Splines müssen einen bestimmten Grundnamen aufweisen, der innerhalb der Programme festgelegt wurde und sich nur in ihrem Suffix unterscheiden darf (beispielsweise O_Spline.1, O_Spline.2, O_Spline.3).

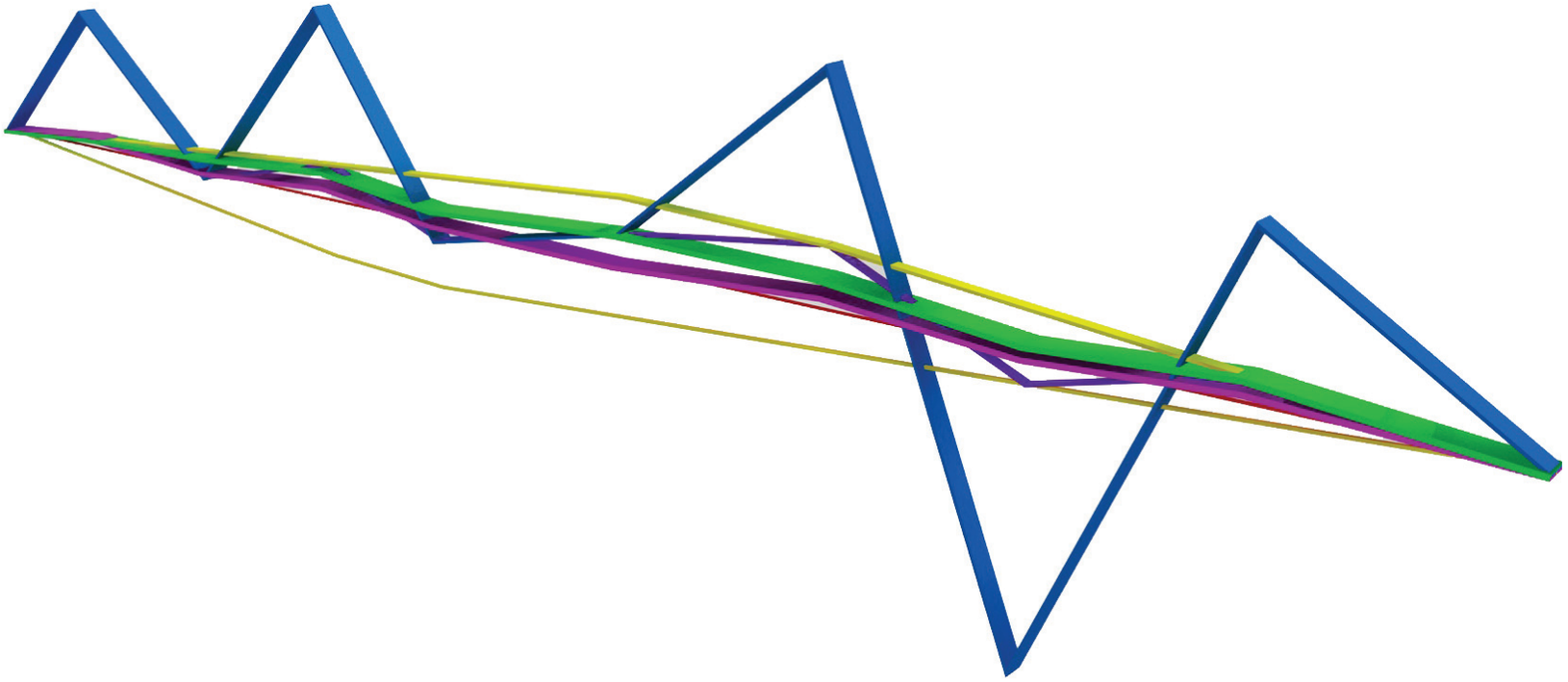
Bemerkungen

Der Funktions-Node könnte erweitert werden, indem man beispielsweise diesem extern bestimmte Namen von zu verzerrenden Objekten übergeben kann.

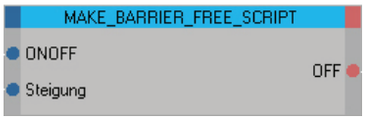
Die Ergebnisse mit diesen Modulen haben dazu geführt, die Entscheidung zu treffen, dass die Punkte der Knetfläche sich nicht überlappen dürfen, denn dies führt zu unkontrollierbaren Überschneidungen. Außerdem wurde angenommen, dass solche Überlappungen und Verschneidungen von Flächen schnell zu einer unnzutzbaren Brückenoberfläche führt.

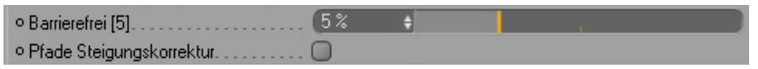






Make_Barrier_Free_Path[]





Allgemeine Beschreibung

Diesem Node können einzelne Splines übergeben werden, wobei diese darauf überprüft werden, ob die von dem/der BenutzerIn definierte maximal zulässige Steigung überschritten wurde. Im Falle einer Überschreitung werden die Splines korrigiert.

Motivation

Dieses Modul wurde entwickelt, um den Entwurf in einem realisierbaren Rahmen zu halten, indem die Brücke für verschiedenste Nutzer in vorgesehenen Wegbereichen nützbar gehalten werden kann.

Im Entwurf

Im Entwurf kann durch dieses Modul garantiert werden, dass die Fußgänger- und Radfahrwege sowie vor allem der barrierefeie Weg ihre maximal zulässigen Steigungen nicht überschreiten und dadurch für die Nutzergruppen benutzbar bleiben.

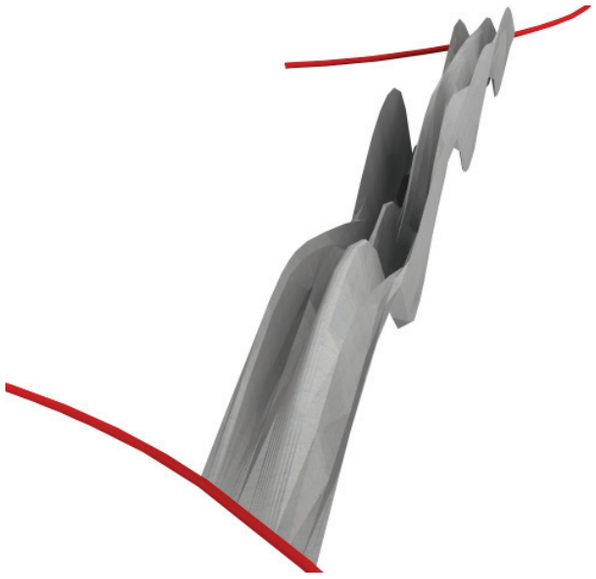
Detail

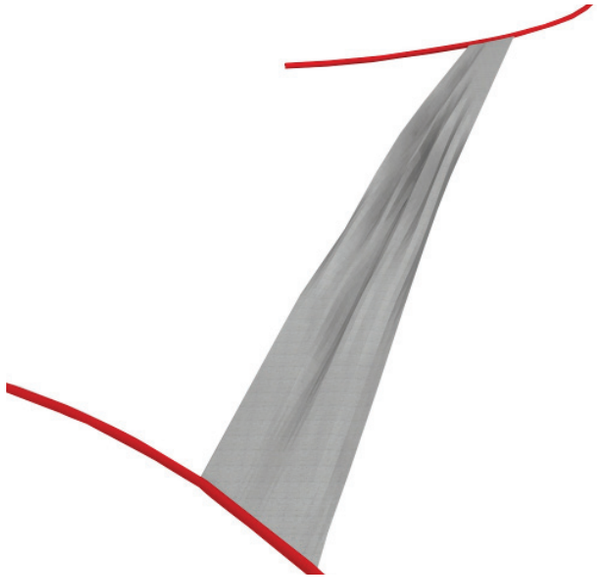
Ein maximal zulässiger Steigungswert kann von dem/der BenutzerIn in Prozent eingegeben werden. Dieser definiert einen maximal möglichen Bereich (gelbe Linien), in dem sich die jeweilige Spline unter diesen Einschränkungen befinden muss. Bei der Veränderung gibt es grundsätzlich zwei Modi, wie die Original-Spline (Blau) verzerrt werden kann. In einer ersten Variante wird die Original-Spline (Blau) zunächst als Ganzes in den zulässigen Bereich (Gelb) skaliert und erst dann geglättet. Dabei wird der Grundcharakter der Original-Spline erhalten (Hellviolett), wobei der Nachteil ensteht, dass der Verlauf meist sehr flach wird. Im zweiten Modus wird die Original-Spline (Blau) nicht skaliert, sondern es werden nur die einzelnen Punkte, die den zulässigen Bereich überschreiten, an die Grenzen verschoben. Danach erfolgt ebenfalls die Glättung. Insgesamt entstehen in der Regel Linienführungen mit markanteren Höhendifferenzen (Grün) als mittels der ersten Methode (hell Violett). Der ideale Verlauf dient als allgemeine Referenz und wird als direkte Verbindung zwischen dem Anfangs- und Endpunkt der einzelnen Splines verstanden.

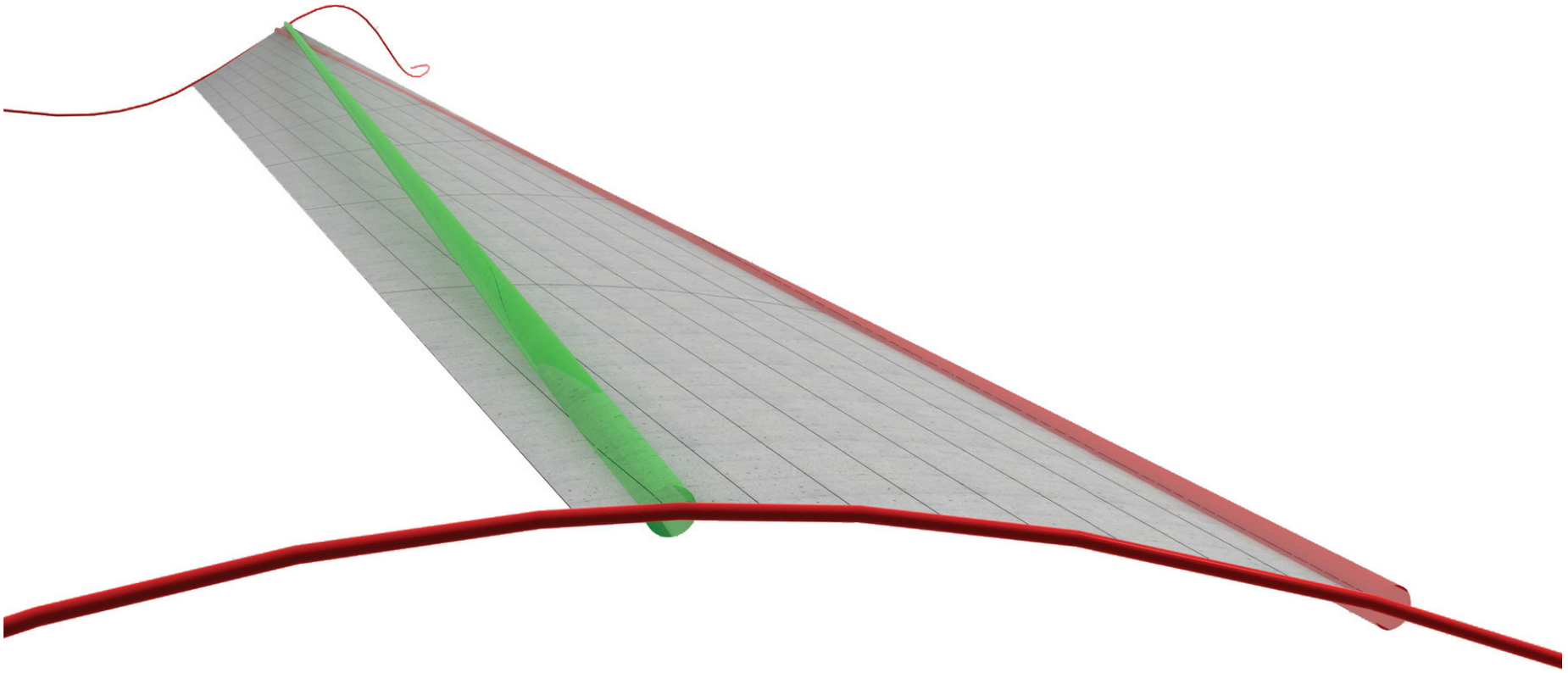
Bemerkungen

Die Modi müssen aktuell innerhalb des Moduls direkt im Script eingestellt werden. Diese Auswahl könnte man auch extern dem/der BenutzerIn zugänglich machen.

Weisen die Endpunkte der einzelnen Splines eine zu große Höhendifferenz auf, entstehen Fehler, da eine Korrektur ohne seitliches Ausweichen (im Grundriss) prinzipiell nicht möglich ist. In einem solchem Fall entstehen Rampen, die am Ende einen Knick aufweisen. Weiters sei an dieser Stelle erwähnt, dass in der Entwicklungsphase zur Überprüfung die Steigungen der einzelnen Splines der Knetflächen getestet wurden. Als Ergebnis resultierten zwar einzelne geglättete Splines, doch die gesamte Oberfläche erschien zerknittert und war daher nicht begehbar. Um definieren zu können, in welchem Bereich die Brückenoberfläche eben verlaufen soll, mussten daher zunächst die Wege erzeugt werden, damit in einem nächsten Schritt das Weglichtenüberprüfungsprogramm entwickelt werden konnte.







Make_Path_Modules[]

Allgemeine Beschreibung

Mit Hilfe der Pfadgenerierung-Nodes kann jeweils eine Spline mit einer frei definierbaren Anzahl an Kontrollpunkten zwischen den Brückenenden generiert werden.

Motivation

Um die Pfade nicht für jeden Test oder auch während des Entwerfens manuell neu eingeben zu müssen, können die Pfade automatisch generiert werden. Die Pfade werden in diesem Entwurfskonzept auch erzeugt, um später für die Defintion der Geometrie sowie für das Bestimmen der Position von Nutzlasten verwendet werden zu können.

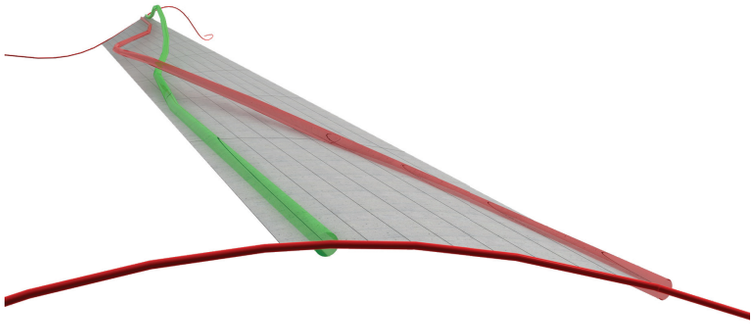


Abbildung 71: Die Wege (Rot und Grün) werden mit diesen Modulen lediglich generiert und führen durch ihr Vorhandensein nicht sofort zu einer Verformung der Knetfläche bzw. Brückenoberfläche.

Im Entwurf

Die Geometrie der Brücke wird wesentlich von der Wegeführung bestimmt. Um die Wege nicht bei jedem Entwurf manuell neu zeichen zu müssen, können diese automatisch erzeugt werden. Der Pfad kann nach der Generierung beliebig manipuliert und weiterbearbeitet werden. Auch die Anfangspositionen können jederzeit verändert werden.

Detail

Dabei wird die Spline zwischen den Brückenenden innerhalb des Knetmassenbereichs generiert. Die Wege-Splines werden intern als B-Splinetypen definert. Als Vorschlag wird je Pfad ein zufälliger Anfangs -und Endpunkt vergeben, der natürlich nach Generierung direkt manuell verändert werden kann.

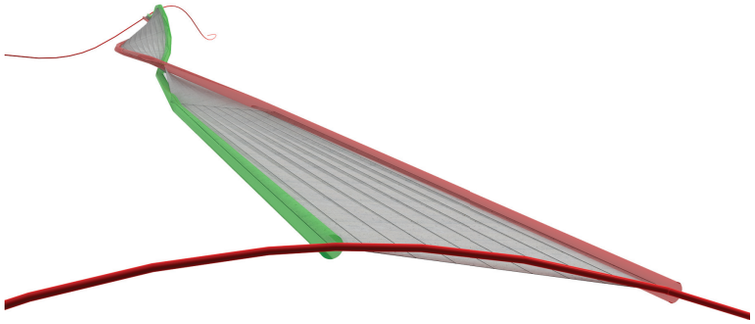


Abbildung 72: Achtung! Um die Brückengeometrie dem Verlauf der einzelnen Wege anzupassen, muss erst das "Knead_to_Paths-Modul" aktiviert werden.

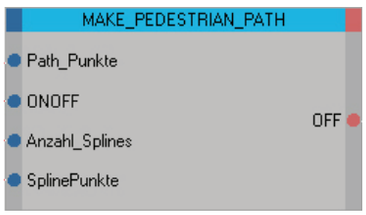
Bemerkung

Im Entwurfsprozess werden diese Pfadegenerierungsmodule zweimal für die Erzeugung von zwei Wegen annähernd ident verwendet. Der einzige Unterschied besteht darin, dass für die jeweilig generierten Pfade unterschiedliche Namen vergeben werden.

Als Modulserweiterung könnte die Splinetypwahl dem/der BenutzerIn extern über das User-Interface zugängig gemacht werden.

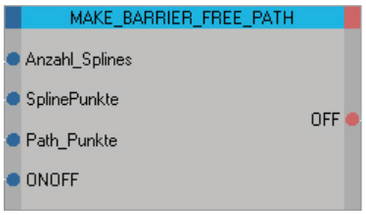
Make_Pedestrian_Path[]

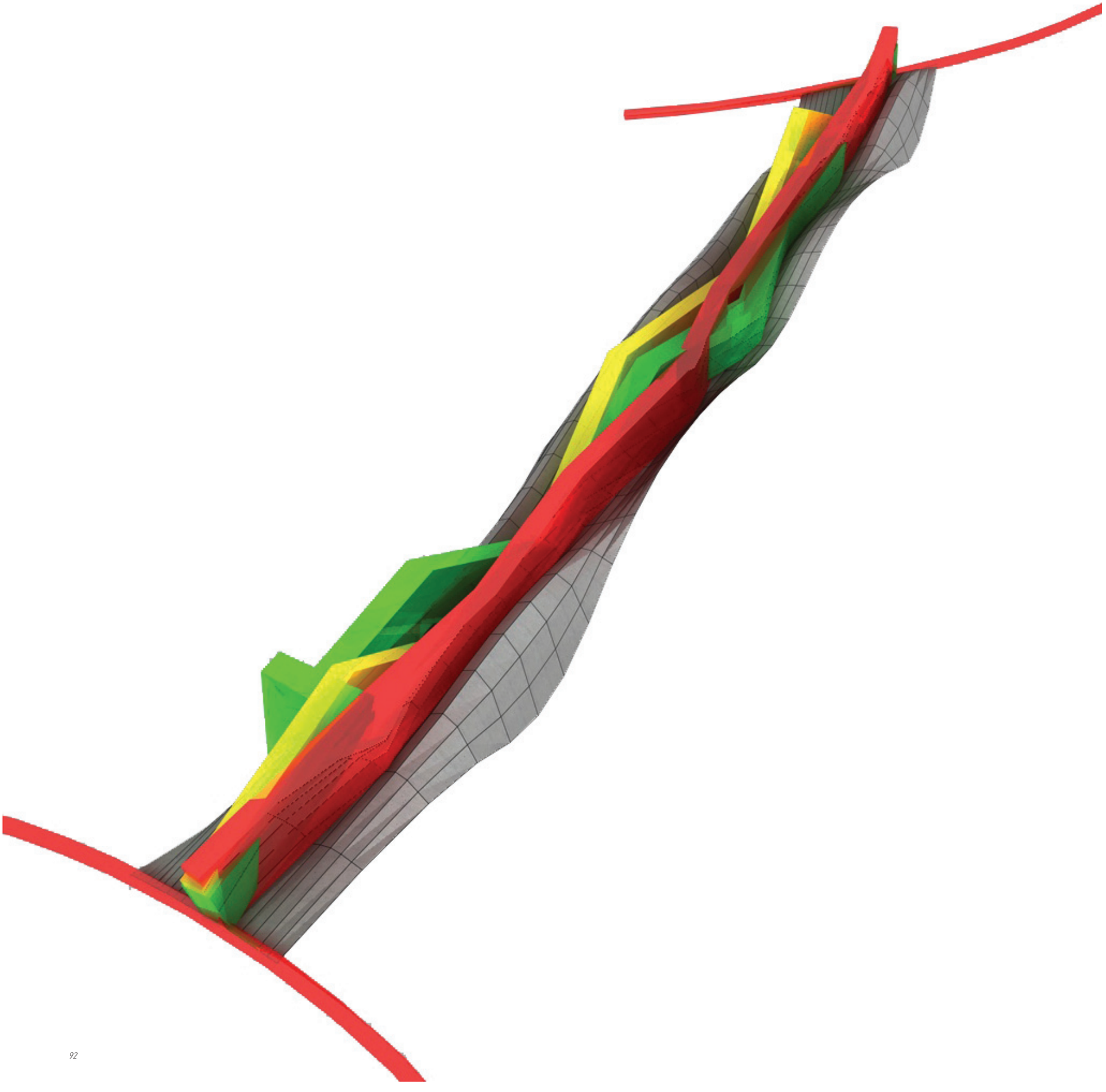
Erzeugt ein Spline Objekt mit dem Namen "Pedestrian_Path_Spline".



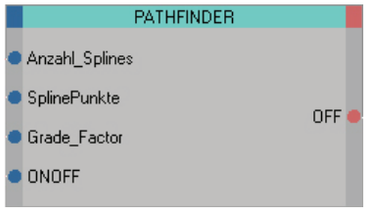
Make_Barrier_Free_Path[]

Erzeugt ein Spline Objekt mit dem Namen "Barrier_Free_Path_Spline".





Pathfinder[]



Allgemeine Beschreibung

Mit dem "Pathfinder-Modul" kann eine Spline generiert werden, die den kürzesten Weg zwischen einem definierten Anfangs- und Endpunkt in einer Punktematrix widerspiegelt. Mit einem Faktor kann die Empfindlichkeit gegenüber Höhenveränderungen eingestellt werden.

Motivation

Beim Entwurf freier Brückengeometrien kann es manchmal schwierig sein, einen günstigen Weg für unterschiedliche Nutzergruppen über die Brücke zu finden und dabei manuell zu konstruieren. Mit der Pathfinder-Funktion wird dieser Vorgang automatisiert.

Im Entwurf

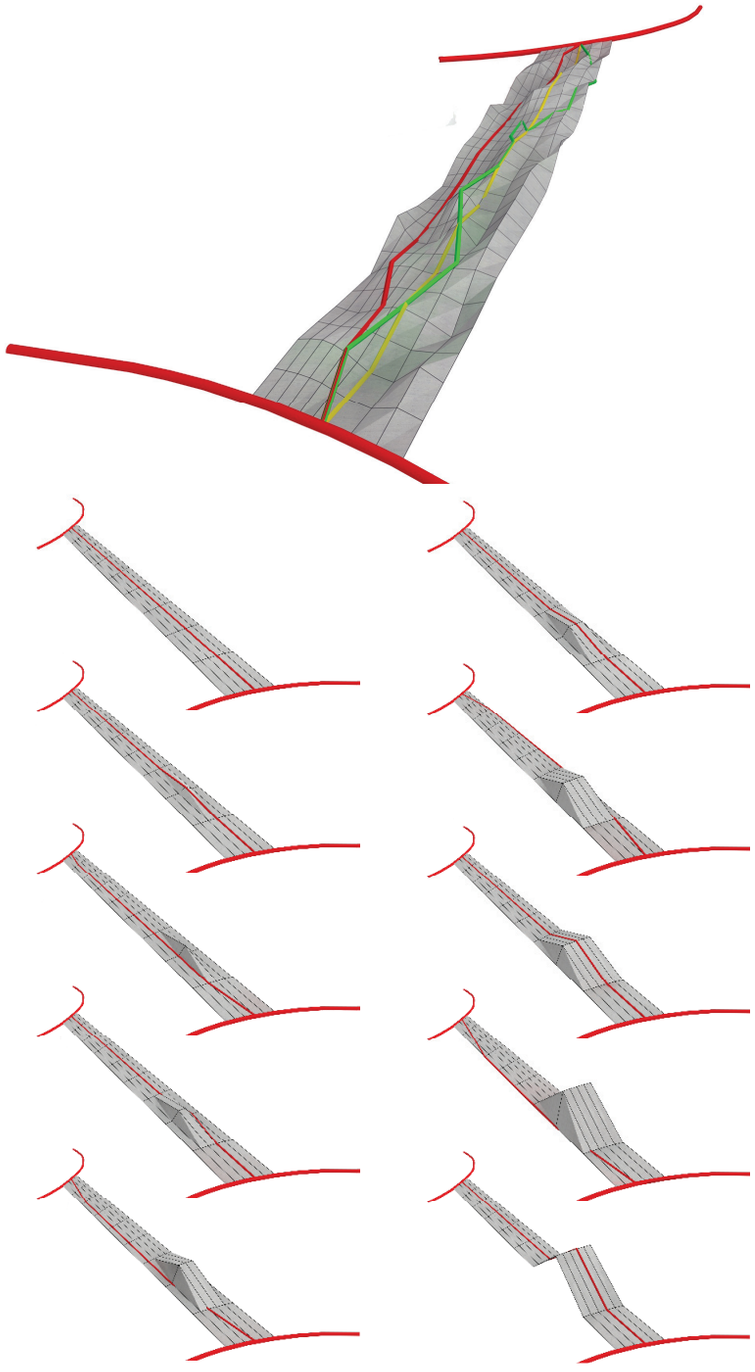
Im Entwurf wird dieses Modul genützt, um den günstigsten Radfahr- oder auch barrierfreien Weg über die Brückenoberfläche zu finden. Es werden zwei Punkte auf der Brücke gewählt, um danach den kürzesten Weg über die entworfene Oberfläche zu berechnen und diesen als Pfad-Objekt darzustellen. Je nachdem, wie der Höhenempfindlichkeitsfaktor eingestellt wird, ergibt sich ein eher direkter oder indirekter Pfadverlauf. So kann beispielsweise der günstigste Verlauf für einen barrierefreien Weg gesucht werden. In einem weiteren Schritt kann dieser Weg mit dem "Make_Barrier_Free_Path-Modul" auf die maximal zulässige Steigung korrigiert werden, um danach einen ebenen Verlauf entlang des Pfades mit einer definierten Lichte mittels des "Path_Clearance-Modul" generieren zu können.

Detail

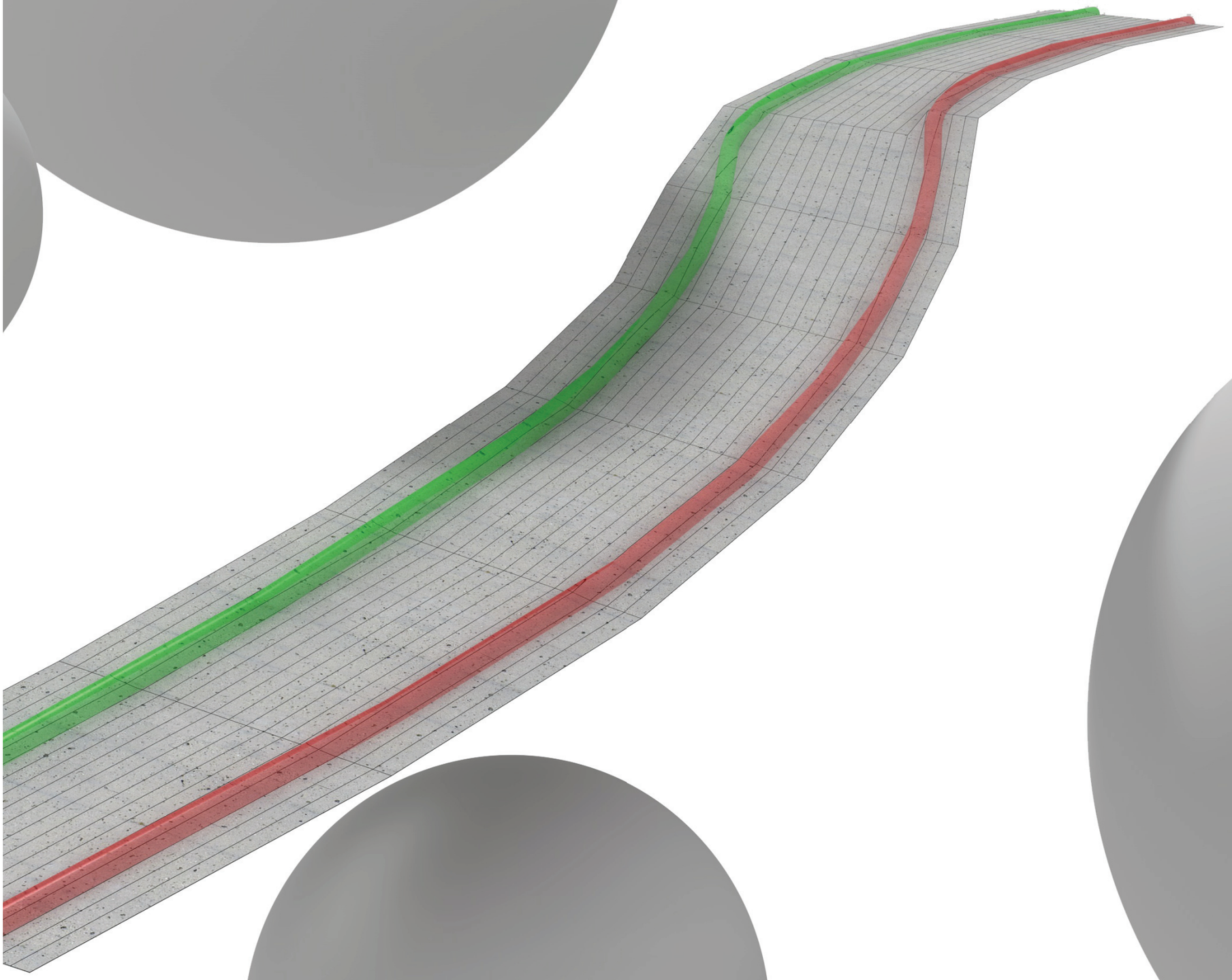
Die zwei Punkte, zwischen denen der kürzeste Weg berechnet wird, werden im Modul-Script festgelegt. Diese Pfadanfangs- bzw. Pfadendpunkte werden mittels Zufallsgeneratoren an den Brückenenden festgelegt. Der Weg wird mittels des sogenannten "Dijkstra Agorithmus" berechnet, welcher eine gängige Methode für diese Art von Problemstellung darstellt. Grundsätzlich werden mittels dieses Algorithmus' einzelne Strecken bewertet bzw. gewichtet, weshalb man dieses Vorgehen der Graphentheorie zuordnet.

Bemerkung

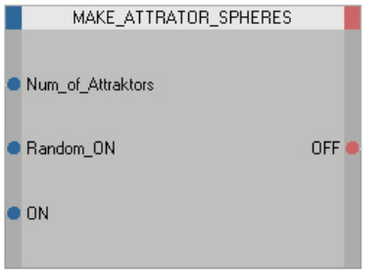
Die Defintion der zwei Zielpunkte könnten dem/der NutzerIn extern über das User-Interface zugänglich gemacht werden.



Die einzelnen Pfade in den Abbildungen wurden mit diesem Modul automatisch generiert. Dabei wurde der "gelbe Pfad" mit einem niedrigen Höheneinflussfaktor generiert, weshalb dieser einen relativ geradlinigen Verlauf über die Oberfläche einnimmt, da dieser gegen Höhensprünge relativ unempfindlich ist. Der "grüne Pfad" wurde mit einer hohen Empfindlichkeit berechnet, weshalb dieser einen Zickzackkurs wählt, um Höhenüberwindungen zu vermeiden. Man erkennt in den Abbildungen, dass alle Pfade aus denselben Endpunkten enspringen, sich jedoch unterschiedlich über die Oberfläche entwickeln.



Make_Attractor_Spheres[]



Allgemeine Beschreibung

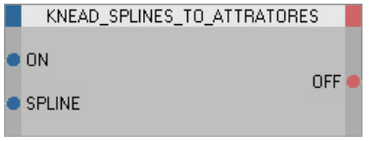
Mit diesem Modul werden Kugelobjekte mit zwei unterschiedlichen Namensgebungen wie etwa "POSITIVE_ATTRACTOR_Sphere13" und NEGATIV_ATTRACTOR_Sphere13" erzeugt. Bei der Generierung kann die Anzahl der Kugeln bestimmt und ein Zufallsmodus aktiviert oder deaktiviert werden.

Motivation

Um im Verlauf der Pfade bestimmte Qualitäten des Ortes reflektieren zu können, sollte es ermöglicht werden, diesen mittels anziehenden und abstoßenden Attraktoren auszustatten, die zunächst verschiedenst interpretiert werden können.

Im Entwurf

Das Entwurfskonzept sieht vor, dass ein Pfad im Verlauf besonders Rücksicht auf die Qualitäten der Umgebung nimmt. In der Wegeverlaufsfindungsphase können die Kugeln als Magneten verstanden werden, die je nach Radius und Bezeichnung eine mehr oder weniger stark anziehende oder abstoßende Wirkung auf den jeweils selektierten Pfad besitzen. Mittels diesem Modul kann zunächst die Entwurfsumgebung mit negativen und positiven Attraktoren bewertet werden.



Detail

Bei der Generierung kann der Zufallsmodus, der unterschiedliche Radien und Positionen im Entwurfsraum vergibt, aktiviert oder deaktiviert werden. Ist der Zufallsmodus deaktiviert, werden alle Kugeln mit dem gleichen Radius in derselben Position erzeugt, um dann manuell verändert zu werden. Bei der Attraktorentypenbestimmung bzw. der Namensgebung der Kugeln wird die gewünschte Anzahl halbiert, wobei eine Hälfte den Präfix "NEGATIV_" und die andere "POSITIV_" vergeben bekommt. Bei einer ungeraden Anzahl wird zugunsten der Erzeugung von positiven Attraktoren aufgerundet.

Knead_Splines_To_Attractors[]

Allgemeine Beschreibung

Mit diesem Node kann eine Spline selektiert werden, die auf die Attraktorkugeln in der Entwurfsszene reagiert.

Im Entwurf

Zusammen mit dem vorhergehenden Modul wird dieser Funktionsknoten wie bereits erwähnt für die Wegeformfindung eingesetzt. Es kann ein Pfad selektiert werden, der auf die Attraktoren im Entwurfsraum mit einer Verformung reagiert.

Motivation

Die vergebenen Attraktoren sollten Auswirkungen auf die Wegeführung haben, weshalb dieses Wegeausrichtungmodul entwickelt wurde.

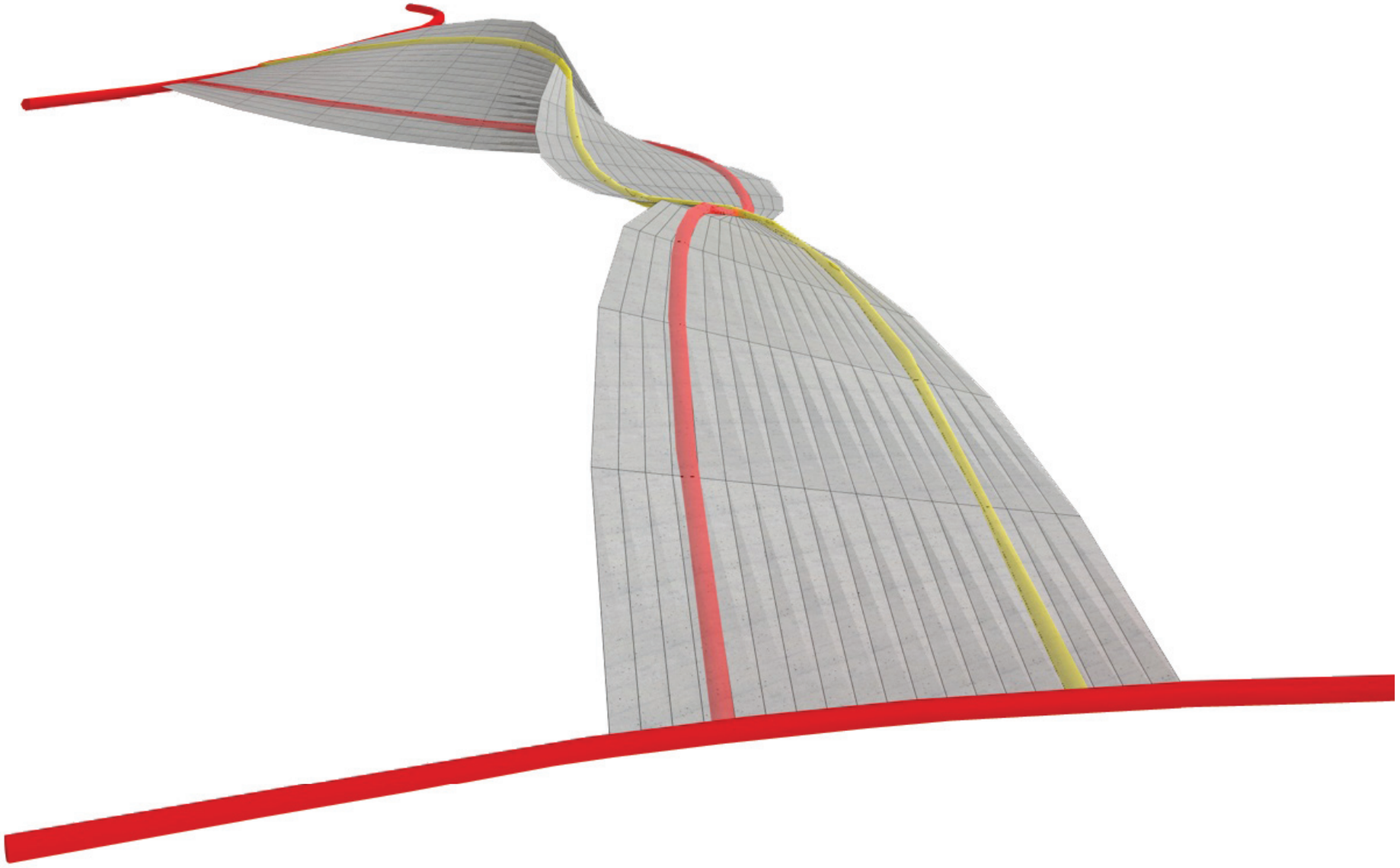
Detail

Genau genommen wirken sich die Attraktoren auf die einzelnen Kontrollpunkte der Splines aus. Der Ausrichtenbefehl kann mehrmals ausgeführt werden. Bei mehrmaliger Befehlausführung pendelt relativ rasch eine optimale Ausrichtung ein und es finden keine sichtbaren Positionsverschiebungen der Kontrollpunkte statt.

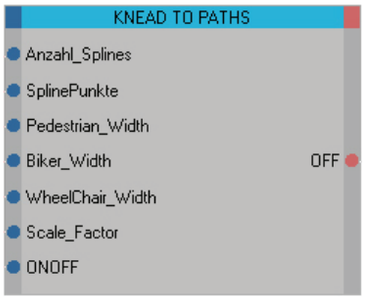
Bemerkungen

Eigentlich könnte man diese zwei Module in einem zusammenfassen.

Diese zwei Funktionen wurden zwar im anfänglichen Entwurfsstadium als Inspirations- und Testzwecke verwendet, wurden aber bei der Generierung beim Entwerfen der zwei Fallbeispiele nicht verwendet. Es hat sich herausgestellt, dass es sehr schwierig ist, eine objektive Bewertung der Umgebung vorzunehmen. Das Problem ist weniger die Position des Attraktors, als vielmehr die Evaluierung seiner Stärke gegenüber anderen.



Knead_To_Paths[]



Allgemeine Beschreibung

Das "Knead_To Path-Modul" spannt die Grundmatrix zwischen zwei Splines auf, wobei eingestellte Pfadbreiten und ein zusätzlicher Faktor für die Überlappung berücksichtigt werden können.

Motivation

Wie bereits mehrfach erwähnt ist die Grundform der Brücke von der Wegeführung abhängig, weshalb diese Knet-Funktion für ein schnelles Ausprobieren von Geometrien entwickelt wurde.

Im Entwurf

Nach dem Entwurfskonzept wird die Grundform der Brückenoberfläche wesentlich durch die Wegeführung zwischen den Uferzonen bestimmt. Es ist vorgesehen, dass die Oberflächegeometrie direkt von der Linienführung zweier Wege abhängig gemacht wird. Im Entwurfsprozess kann die Brücke beispielsweise bei Veränderung von Wegbreiten oder Verläufen jederzeit neu aufgespannt werden, wodurch ein schnelles intuitives Ausprobieren von Formen ermöglicht wird.

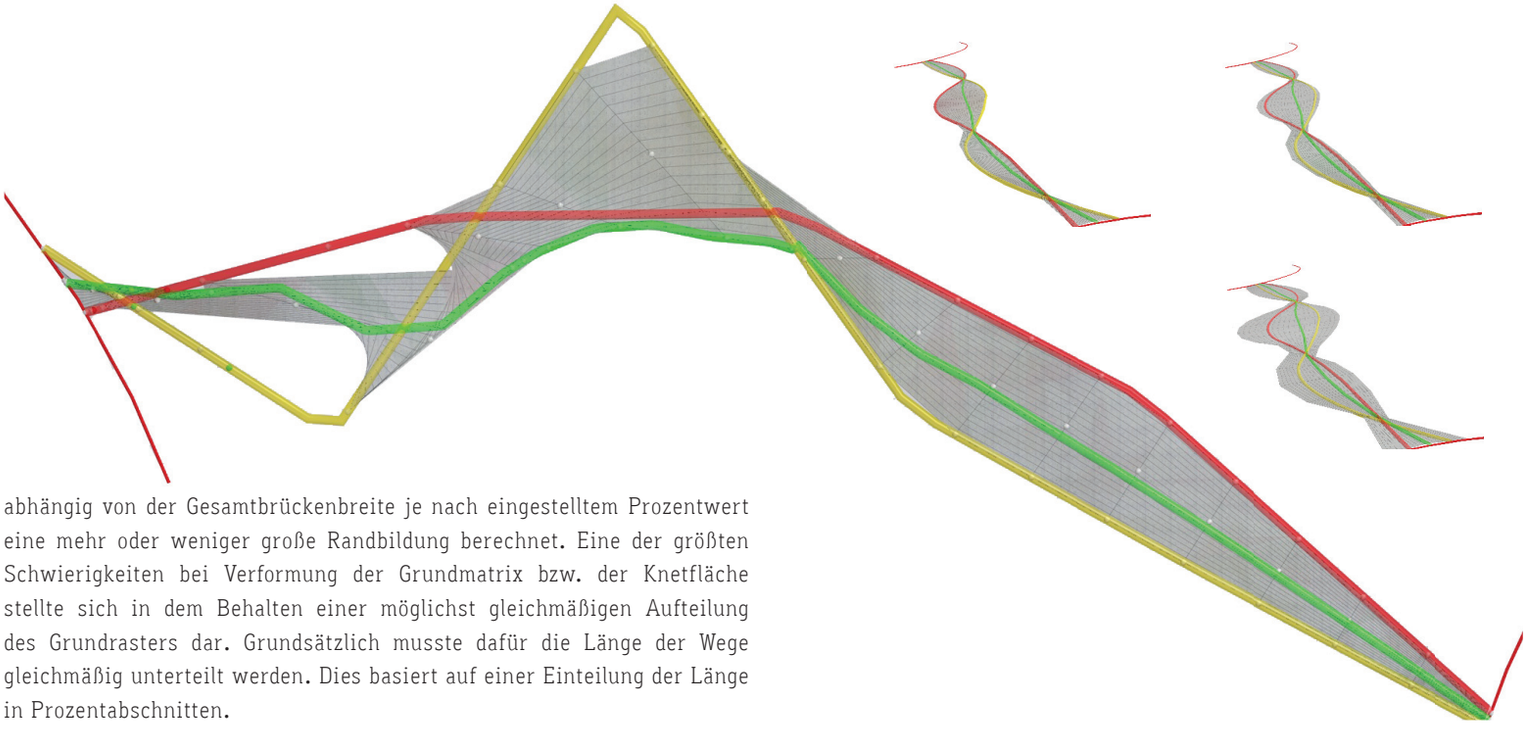
Detail

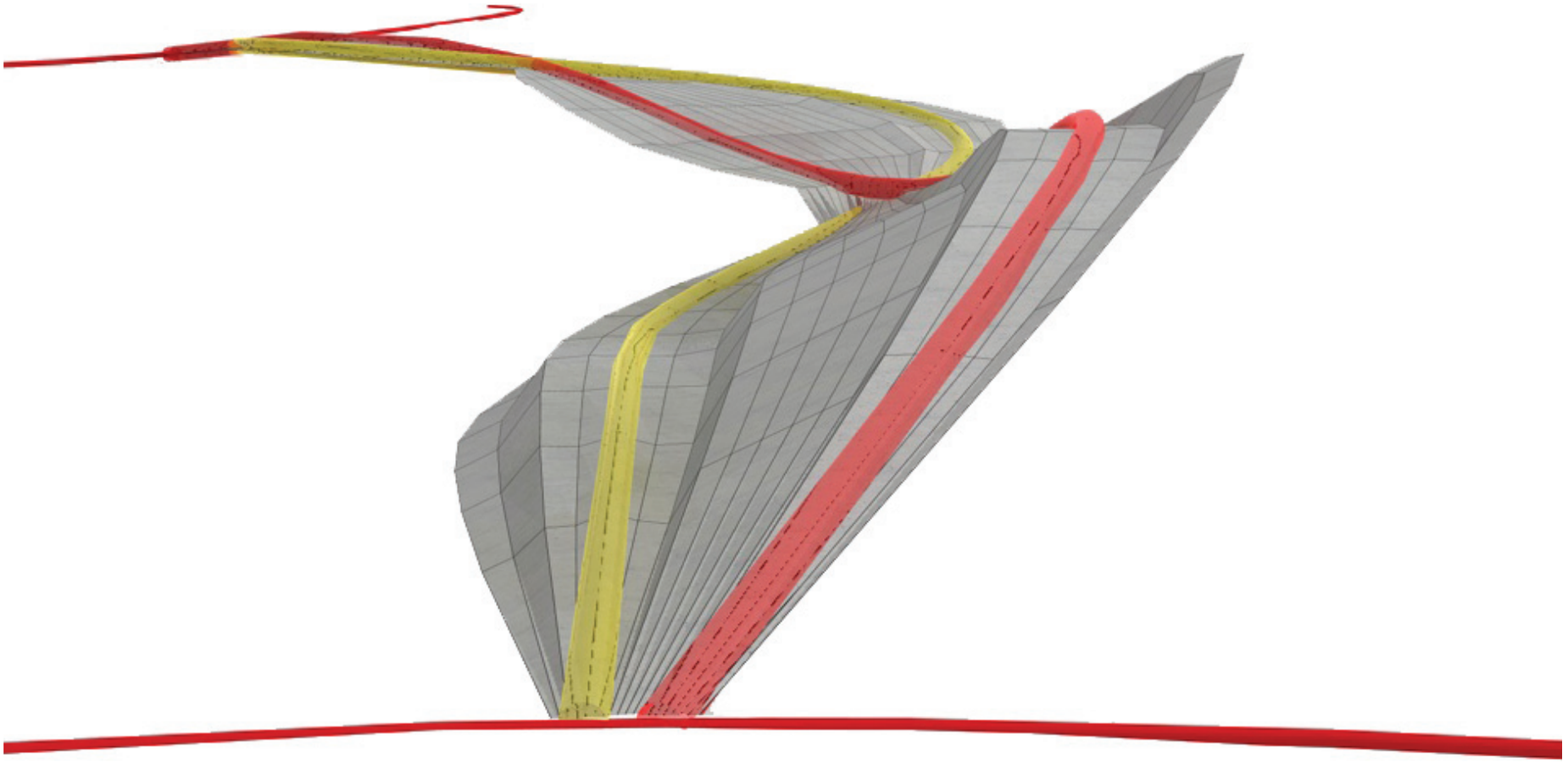
Da Splines als Linienobjekte selbst keine Breite besitzen, definieren sie die Mittelachsen der jeweiligen Wege. Beim Kneten der Fläche muss die Lichte der Wege berücksichtigt werden, indem jeweils die halbe Wegbreite normal auf die Spline in der aufgespannten Ebene liegend hinzugefügt wird. Für den Fall, dass die Wege nicht direkt mit dem Brückenrand abschließen sollen, wird ein zusätzlicher Faktor bereitgestellt, der abhängig von der Gesamtbrückenbreite je nach eingestelltem Prozentwert eine mehr oder weniger große Randbildung berechnet. Eine der größten Schwierigkeiten bei Verformung der Grundmatrix bzw. der Knetfläche stellte sich in dem Behalten einer möglichst gleichmäßigen Aufteilung des Grundrasters dar. Grundsätzlich musste dafür die Länge der Wege gleichmäßig unterteilt werden. Dies basiert auf einer Einteilung der Länge in Prozentabschnitten.

Bemerkungen

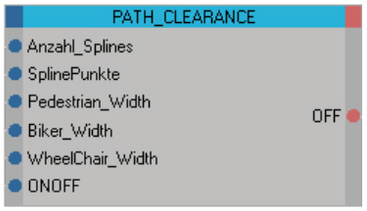
Diese geknetete Oberfläche muss nicht automatisch begehbar sein. Wenn beispielsweise zwar die einzelnen Steigungen der Wege vor dem Kneten korrigiert wurden, kann es trotzdem vorkommen, dass durch den Unterschied der Wegeführung untereinander eine zu großer Querneigung ensteht. Dieses Problem kann mit dem Path_Clearence-Modul behoben werden.

Weiters wurde anfangs auch versucht, das Programm so zu gestalten, dass die Fläche immer zwischen den äußersten Pfaden ausgerichtet wird. Da zu diesem Zeitpunkt aufgrund des Mangels der Kenntnis einer programminternen Unterscheidung zwischen zwei Prozentrechnungsbefehlen immer wieder Fehler auftauchten (siehe Abbildung), wurde die Ausrichtung auf zwei Pfade beschränkt. Dies könnte in neuen Versionen wieder eingeführt werden, wurde jedoch hier aus Zeitgründen nicht wieder integriert.





Path_Clearance[]

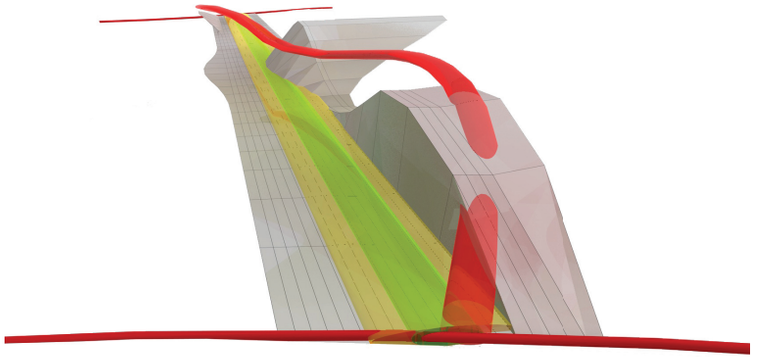


Allgemeine Beschreibung

Mittels der "Path_Clearance"-Funktion können die Höhenkoordinaten der einzelnen Punkte einer Matrix, die sich innerhalb eines definierten Normalabstandes links und rechts neben einer Spline befinden, dem Höhenverlauf der Spline angeglichen werden.

Motivation

Um einen Freiformbrückenentwurf realisierbar bzw. benutzbar zu machen, muss zumindest in den vorgesehenen Verkehrszonen wie etwa Radfahrer- und Fußgängerbereich eine relativ ebene Fläche garantiert werden. Um diese ebenen Zonen rasch automatisch generieren zu können, wurde die sogenannte "Path_Clearance"-Funktion entworfen bzw. das "Weglichte_machen_Script" geschrieben.



Im Entwurf

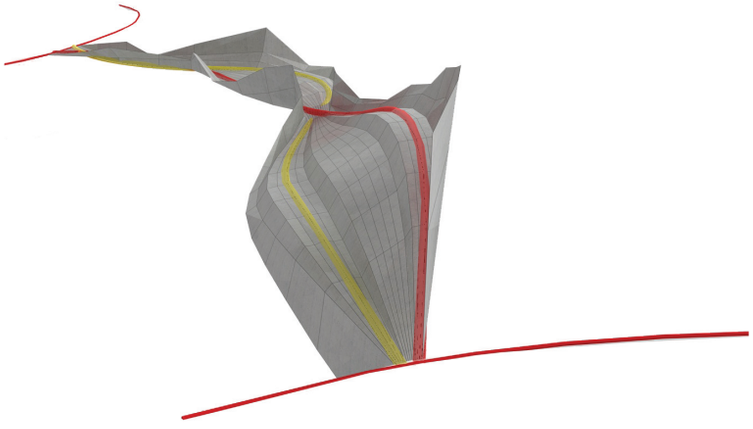
Um eine eventuell äußerst freigeformte Brückengeometrie begehbar zu machen, kann mittels dieser Funktion eine freizuhaltende Lichte jederzeit geschaffen werden, indem die Flächen links und rechts der Wegachse eingeebnet werden können. Dadurch entstehen in den Wegbereichen in der Brückenoberfläche stufenförmige Einschnitte.

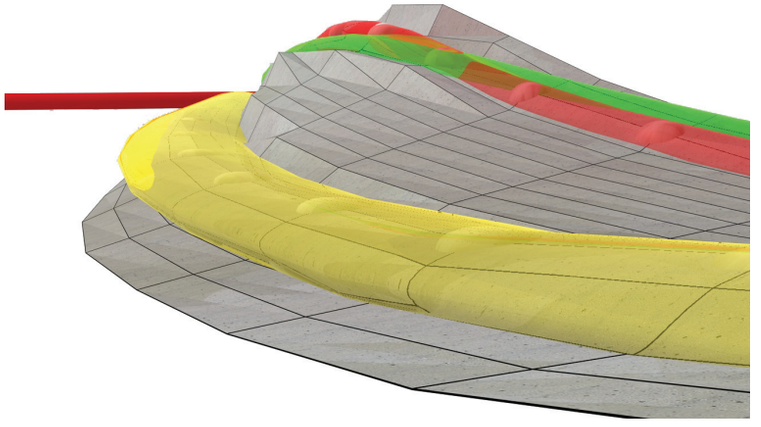
Detail

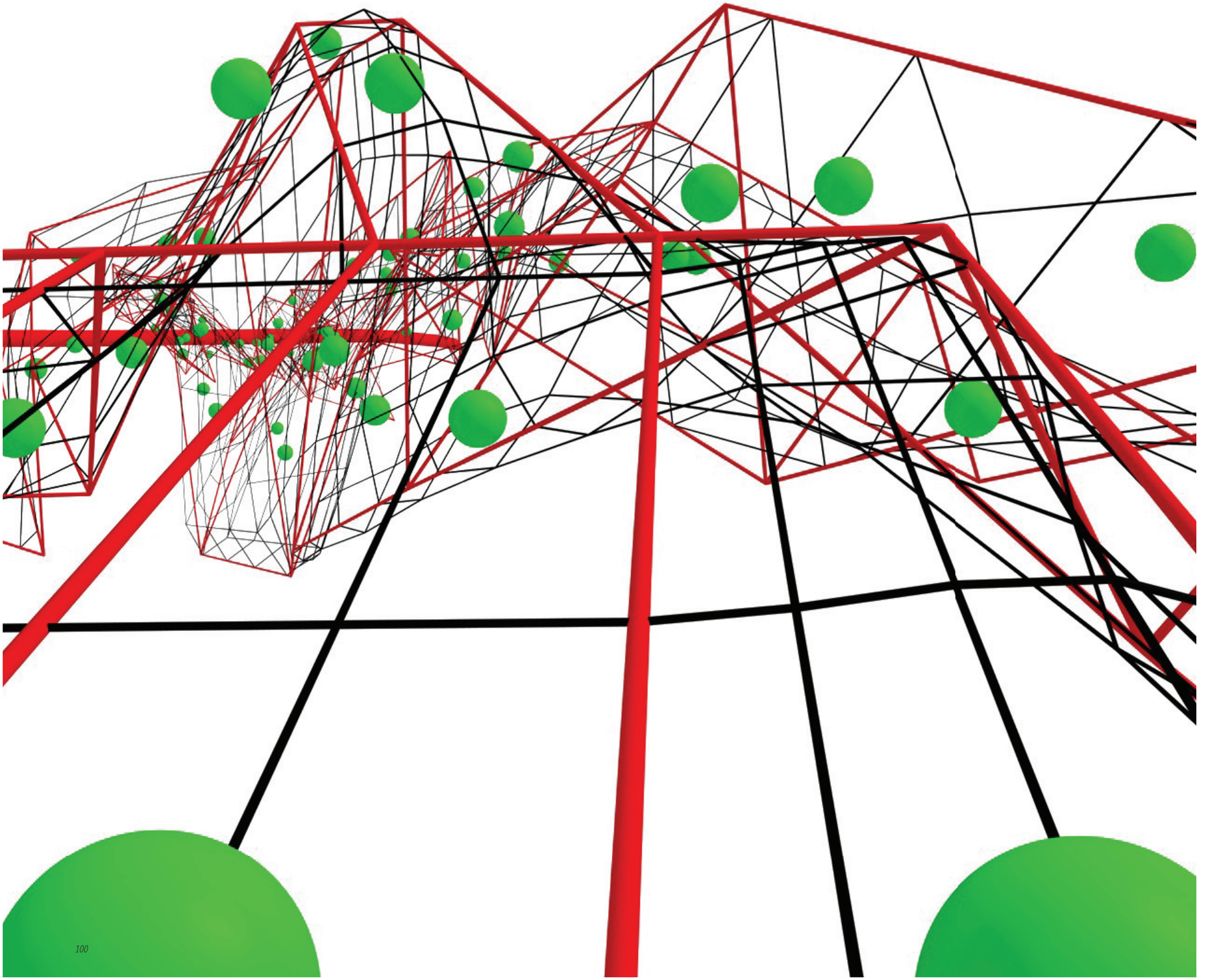
Grundsätzlich sollte aufgrund eines möglichst hohen Freiheitsgrades beim Entwerfen das Kreuzen von Wegen zugelassen werden. Dabei wurde etwa die Frage aufgeworfen, nach welchem der Pfade nun geebnet werden soll. Etwa nach dem niedrigsten, höchsten Pfad? Die Frage wurde mit einer Prioritätenvergabe beantwortet, die sich danach richtete, welche Wege mehr oder weniger Flexibilität gegenüber Höhenveränderungen aufweisen. Oberste Priorität erhielt der barrierefreie Weg, da er die meisten Einschänkungen besitzt. Dies bedeutet, die beiden anderen Pfade müssen sich bei einem Zusammentreffen bzw. Überlappen in ihrer Höhenentwicklung nach diesem Pfad richten. Als zweitwichtigster Pfad wurde der Radfahrerweg eingestuft. Der Fußgängerweg, welcher den größten Freiheitsgrad gegenüber Veränderungen besitzt, muss sich nach dieser Prioritätensetzung allen anderen Pfaden angleichen.
Weiters ergibt sich das Problem, dass nach dem Angleichen der Pfade untereinander deren maximal zulässige Steigungen eventuell nicht mehr eingehalten werden. Bei einer neuerlichen Steigungskorrektur enstehen wieder Pfade, die sich in unterschiedlichen Höhen überlappen. Dadurch kann eine Endlosschleife entstehen. Dieses Problem wurde minimiert, indem das "Path_Clearance"-Script die Änderungen mit einer kleinen Toleranz ausführt, wodurch sich im Regelfall ein Optimum einpendeln kann.

Bemerkung

Als Erweiterung könnte man beispielsweise das Weglichtenprofil etwa über eine grafische Eingabe definierbar machen.







Structure_Evaluating[]

Allgemeine Beschreibung

Das "Structure_Evaluating"-Modul generiert für eine aktuelle Brückenoberfläche mehrere Kubatur- sowie auch Brückenquerschnittsvorschläge, wobei einige Daten der Berechnungen schriftlich im Consolen-Fenster der Animationssoftware sowie auch grafisch am Brückenobjekt sichtbar gemacht werden.

Motivation

Wie bereits in der Aufgabenstellung erwähnt wurde dieses Modul mit dem Ziel entwickelt, eine Vorbemessung für von der Norm abweichende Formen zu entwickeln und dabei Material- und Konstruktionsüberlegungen zu berücksichtigen.

Im Entwurf

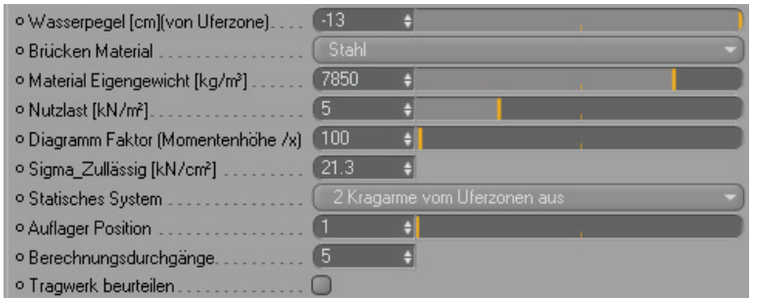
Da standardmäßig als Vorschlag bereits einige Daten im User-Interface vorläufig festgelegt sind, muss als Vorraussetzung für eine Berechnung lediglich die Grundmatrix bzw. die sogenannte Knetfläche generiert worden sein. Grundsätzlich kann jeweils zwischen zwei gängigen statischen Systemen und Materialien gewählt werden. Weiters können beispielsweise das Materialgewicht, die Nutzlast, die maximal zulässige Spannung (Sigma) sowie verschiedene Auflagerpositionen extern vom Benutzer über das User-Interface eingegeben werden. Weiters können mehrere Berechnungsdurchgänge gefordert werden. Bis zum vierten Berechnungsdurchgang wird versucht, für die aktuelle Brückenoberfläche vier unterschiedliche Kubatur- und Querschnittsvarianten zu erzeugen. Danach schlägt das Script Veränderungen der gesamten Brückenform vor, indem es selbstständig auf die anderen Module zugreift und beispielsweise die einzelnen Wegeführungen etc. zu manipulieren beginnt.

Da sich der Entwurfsort laut Wettbewerbsausschreibungen in einem hochwassergefährdeten Gebiet befindet, wird versucht, einen Wasserpegel zu berücksichtigen, in den die Brücke nicht eintauchen sollte. Der Wasserpegel kann im User-Interface eingetragen werden, wobei dieser vom tiefsten Brückenoberflächenpunkt gemessen wird. Taucht nun die vorgeschlagene Brückenkonstruktion in das Wasser ein, wird versucht, die Brücke in diesem Bereich aus dem Wasser anzuheben.

Als Ergebnis wird jede Brückenvariante als dreidimensionales Objekt mitsamt den wichtigsten Daten in einer Ordnerstruktur in der Entwurfsszene archiviert, um die einzelnen Berechnungsschritte nachvollziehen zu können. Es können beispielsweise jederzeit Veränderungen manuell oder automatisch über das Aktivieren anderer Module getätigt werden und eine erneute Berechnung gestartet werden.

Detail

Als statisches System kann zwischen einem Kragarm und einem Einfeldträgersystem gewählt werden. Beim Kragarmsystem kann zusätzlich



zwischen zwei unterschiedlichen Kragarmsverläufen gewählt werden. Es kann entschieden werden, ob zwei Kragarme an der veränderbaren Auflagerposition eingespannt sind oder ob diese vom Ufer ausgehen und die im User-Interface einstellbare Auflagerposition lediglich die Spitzen der Kragarme definiert. Wird nun etwa im User-Interface die Auflagerposition in der Mitte der gesamten Überbrückung gewählt, werden nun im ersteren Fall zwei Kragarme beispielsweise in der Mitte eines Flusses auflagern und mit der Spitze in Richtung der beiden Uferenden zeigen. Im letzteren Fall würde dies den umgekehrten Fall bedeuten, dass in diesem Modus die Auflagerposition im User-Interface eigentlich die Brückenspitze bezeichnet. So würden sich in diesem Fall die Auflager an Uferzonen befinden und die zwei Kragarme mit deren Spitzen zur Flussmitte zeigen. Wird für die Auflagerposition die erste oder letzte mögliche Position gewählt, dann wird nur ein Kragarm generiert.

Weiters kann entweder Beton oder Stahl für die Brückenkonstruktion gewählt werden. Dies hat vor allem Auswirkungen auf die unterschiedlichen Grundparameter bei der Definition der Brückenquerschnitte und der Tragfähigkeits- und Gebrauchstauglichkeitsprüfung.

Vereinfachte Beschreibung des Tragwerksfindungsprozess'

Der Ablauf einer Tragwerksbemessung wurde bereits im Kapitel "Entwerfen mit Kräften - Grundkenntnisse" von Seite 59 bis 61 allgemein dargestellt.

Kurz:

1. Statisches System bestimmen
2. Lasten ermitteln und Auflagerreaktionen berechnen
3. Schnittkraftermittlung
4. Tragfähigkeit überprüfen
5. Gebrauchstauglichkeit nachweisen

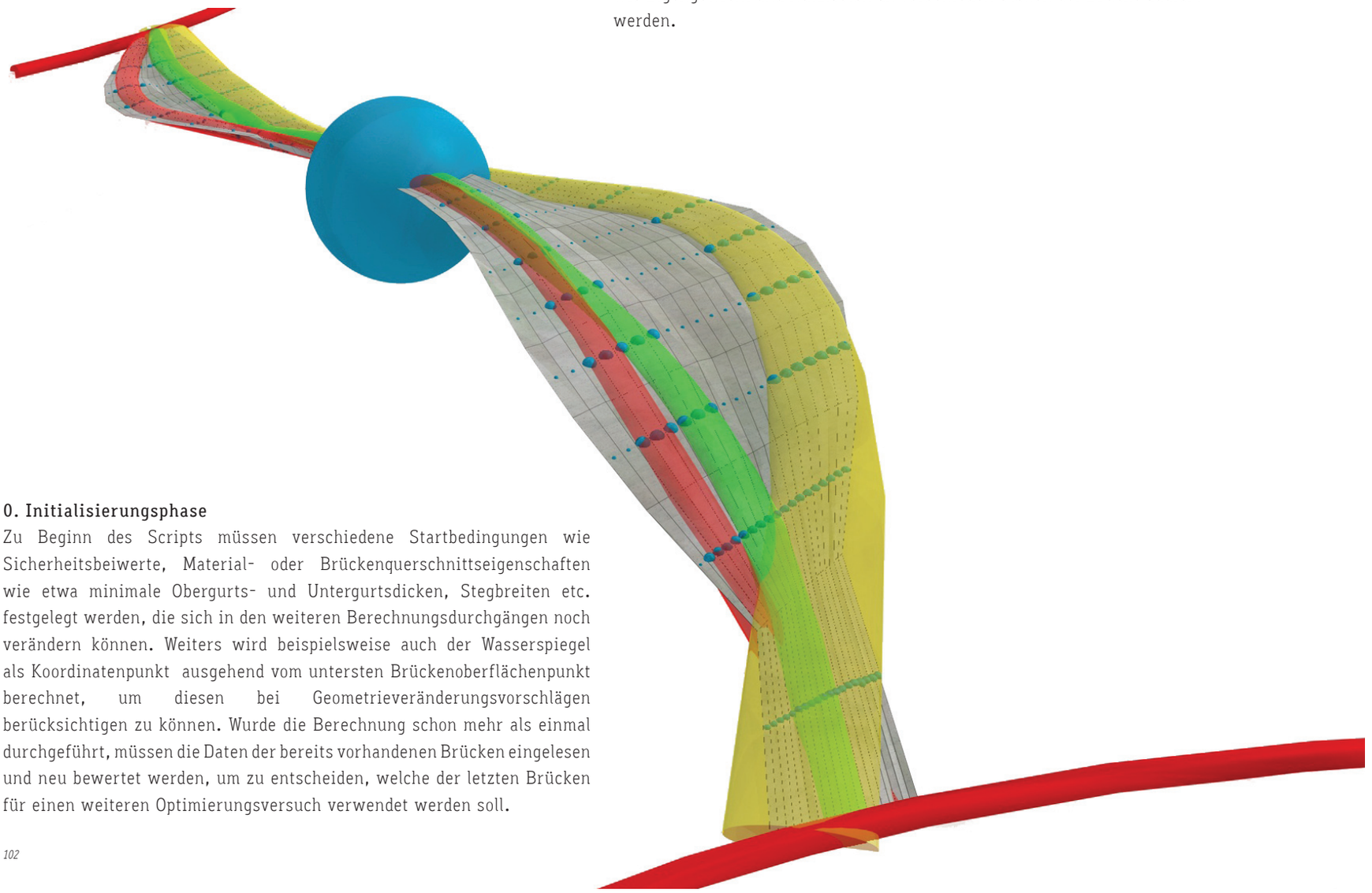
Im Folgenden wird in groben Zügen versucht darzustellen, wie die Brückenkubatur bzw. die Querschnittsfindung im Script aufgebaut wurde:

1. Statisches System bestimmen

Das statische System wurde bereits im User-Interface festgelegt und wird später in der Momenten- und Durchbiegungsberechnung berücksichtigt.

2. Lasten ermitteln und Auflagerreaktionen berechnen

Um die Nutzlasten berechnen zu können, muss zunächst die Brückenoberfläche ermittelt werden. In Bereichen, in denen sich Verkehrswege befinden, können zusätzliche Lasten angenommen werden. Laut "EUROCODE 1" müssen, wie auch in der Wettbewerbsausschreibung angeführt wurde, die Lasten bei Fuß- und Radfahrstegen mit $5kN/m^2$ angenommen werden. Um das Eigengewicht als Last zu bestimmen, wird danach das Brückenvolumen berechnet. Dazu müssen zuvor die Brückenquerschnittsprofile definiert werden. In den ersten zwei Durchläufen werden über die gesamte Brückenlänge Vollquerschnitte vergeben. Weiters wird in der ersten Berechnung die Brückenhöhe über die gesamte Systemlänge konstant angenommen, da es lediglich darum geht, einen ersten Momentenverlauf zu berechnen, der die Variation der Brückenbreite und die damit veränderte Nutzlast über die gesamte Oberfläche berücksichtigt. Nun können Lasten wie Eigengewicht und Nutzlasten an verschieden Stellen der Brücke addiert werden.



0. Initialisierungsphase

Zu Beginn des Scripts müssen verschiedene Startbedingungen wie Sicherheitsbeiwerte, Material- oder Brückenquerschnittseigenschaften wie etwa minimale Obergurts- und Untergurtsdicken, Stegbreiten etc. festgelegt werden, die sich in den weiteren Berechnungsdurchgängen noch verändern können. Weiters wird beispielsweise auch der Wasserspiegel als Koordinatenpunkt ausgehend vom untersten Brückenoberflächenpunkt berechnet, um diesen bei Geometrieveränderungsvorschlägen berücksichtigen zu können. Wurde die Berechnung schon mehr als einmal durchgeführt, müssen die Daten der bereits vorhandenen Brücken eingelesen und neu bewertet werden, um zu entscheiden, welche der letzten Brücken für einen weiteren Optimierungsversuch verwendet werden soll.

3. Schnittkraftermittlung

In einem nächsten Schritt kann je nach dem gewählten statischen System und den Lastannahmen an den verschiedenen Brückenstellen der Momentenverlauf berechnet werden.

Beim ersten Berechnungsdurchgang wird je nach Tragwerkssystem mit einer Vorbemessungsformel festgelegt, welche Höhe der Querschnitt an der Stelle mit dem größten Biegemoment besitzen müsste. Mit diesen Daten wird ein erster Idealträger generiert, indem das "maximale Schnittmoment" mit der aus der Vorbemessungsformel resultierenden "Maximalhöhe " gleichgesetzt wird. Danach wird in Orientierung an der "Maximalhöhe" der vorläufige Höhenverlauf über die Brückenlänge anhand der Momentenlinie interpoliert. Bei dieser groben Vorbemessung fließen zusätzlich zwei eigens überlegte Korrekturfaktoren ein. Diese Korrekturfaktoren bewerten die Brückengeometrie an verschiedenen Stellen auf günstige oder ungünstige Querschnittsproportion und stärken oder schwächen die geschätzte Querschnittshöhe an den jeweiligen Stellen zusätzlich. Beim ersten Berechnungsdurchgang wird hier abgebrochen, um die Brücken mit dieser ersten geschätzten Kubatur neu zu berechen.

4. Tragfähigkeit überprüfen

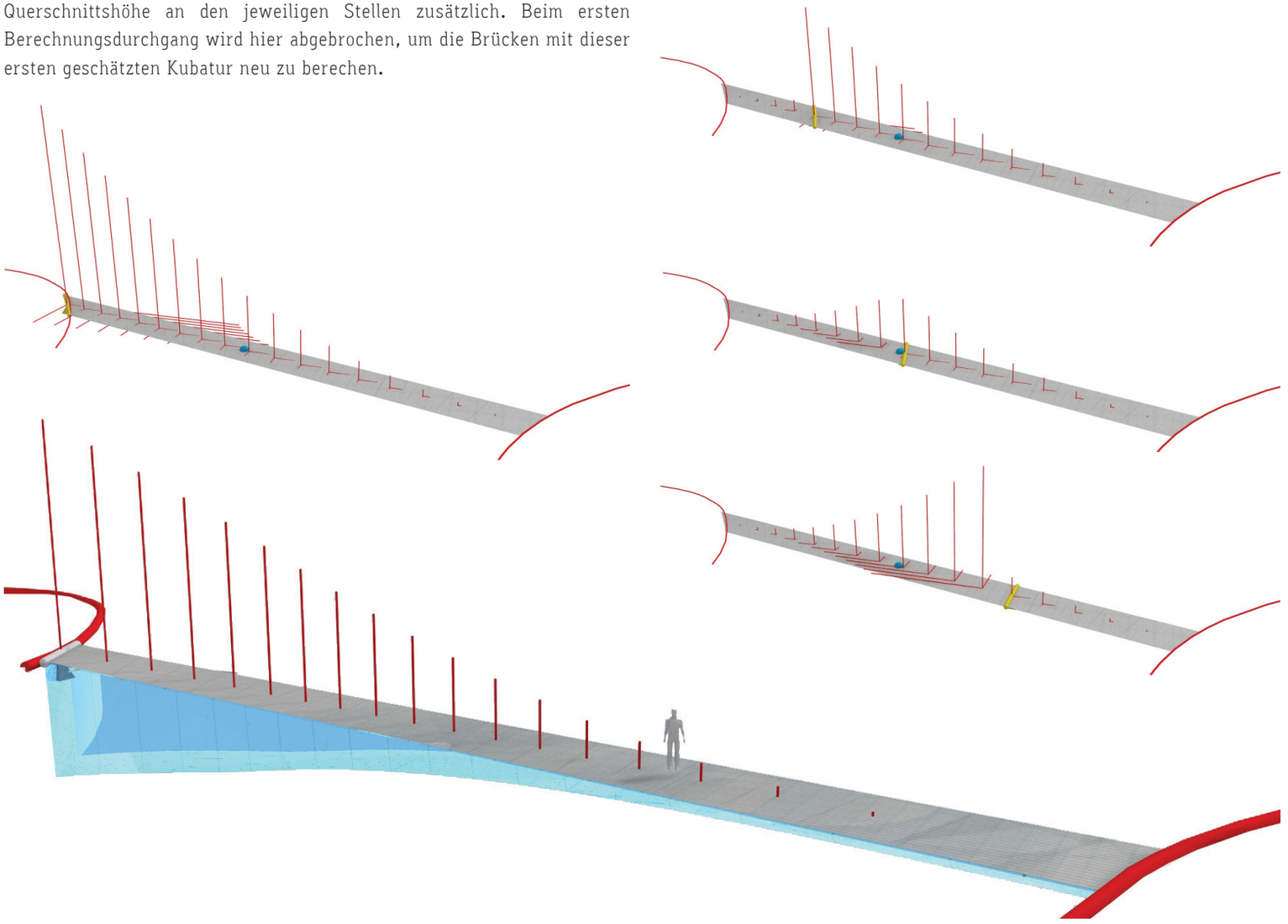
Nachdem der Schnittmomentenverlauf berechnet wurde, werden nun die Flächenträgheitsmomente (I) und die Widerstandsmomente (W) der Brückenprofile an verschiedenen Schnittstellen über die Brücke berechnet. In der Folge werden mit den Schnittmomenten und den Widerstandswerten an den verschiedenen Brückenstellen die Spannungsauslastungen gegenüber den maximal zulässigen Spannungen berechnet und überprüft.

5. Gebrauchstauglichkeit nachweisen

Mit der "Winkelgewichtentheorie" wird schließlich die Gesamtdurchbiegung festgestellt, indem die Verformungen an den einzelnen Brückenstellen ermittelt werden. Wie dies im Detail funktioniert, soll hier nicht weiter erläutert werden.

6. Brückeneigenschaften bewerten und abspeichern

Nun werden die Ergebnisse bewertet und abgespeichert.



Bemerkungen

Wasserpegel:
Die Wasserpegelkontrollfunktion könnte als Extramodul vom Structure_Evaluating-Modul ausgekoppelt werden.

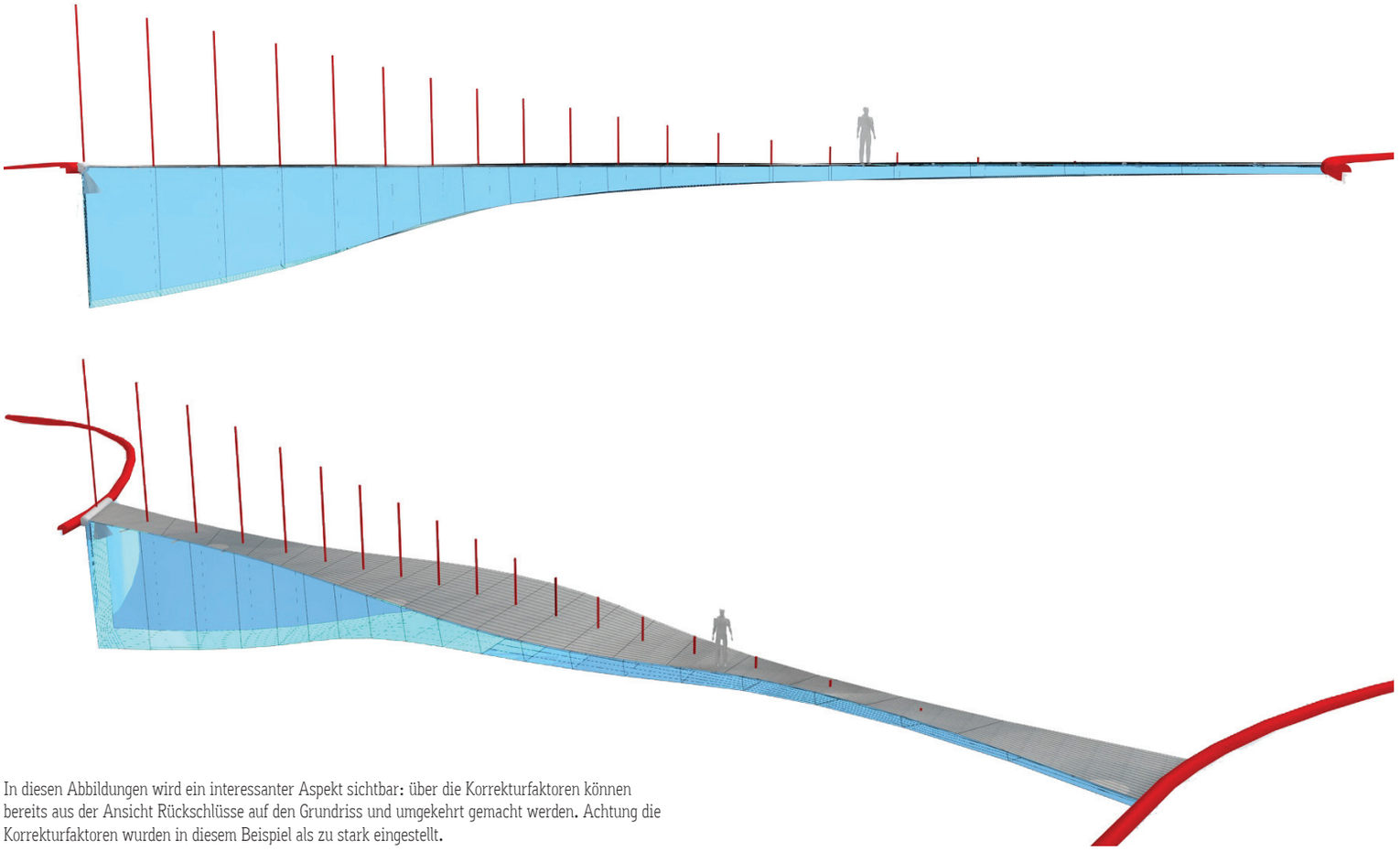
Optimierung:
Die Optimierung des Tragwerkes war in dem Entwurfskonzept sekundär, da es vorrangig um das Parametrisieren einer Tragwerksbemessung für eine Vorbemessung ging. Dabei wurde versucht, über eigens entworfene Korrekturfaktoren, welche die Brückenproportionen auf günstig und ungünstig zu bewerten versuchen, die gängigen Vorbemessungsformeln zu ergänzen.

Für eine weitere Optimierung können aufgrund der Ergebnisse automatisch Veränderungen an der Brücke vorgenommen werden, wobei der Vorschlag für das Hauptkriterium lautet: Mit wenig Masse bei zulässigen Spannungen und wenig Durchbiegung die Distanz überbrücken.

Dafür gibt es im Grunde zwei Möglichkeiten:

Zum Einen kann der Brückenquerschnitt mehr oder weniger ausgehöhlt werden, worduch das Eigengewicht, die Schnittmomente und damit auch die Querschnittbelastungen ab- oder zunehmen. Natürlich nimmt bei zunehmender Aushöhlung auch die Widerstandsleistung des Querschnittes ab. Es wird sich irgendwann ein optimaler Aushöhlgrad einpendeln, wobei allerdings nie die anfangs angenommene Brückenhöhe verändert wurde. Es kann allerdings auch zu gar keinem brauchbaren Ergebnis kommen, wenn bei der fix vorgegebenen Höhe kein Aushöhlungsgrad gefunden werden kann, bei dem die Brücke die zulässigen Werte einhalten kann.

Als zweite Variante kann der Höhenverlauf der Brücke je nach Berechnungsergebnissen variiert werden. Dabei nimmt ebenfalls das Eigengewicht und die Querschnittsleistung zu oder ab und es pendelt sich eine optimale Querschnittshöhe ein. Bei dieser Variante wird allerdings der Aushöhlungsgrad nie verändert, wodurch es passieren kann, dass aufgrund der geringen Hohlräume eine sehr schwere Brücke mit einer extremen Querschnittshöhe gewählt werden muss, um den auftretenden Spannungen genügend Widerstand entgegensetzen zu können. Auch hier kann es dazu kommen, dass keine Lösung möglich ist, da dass Brückengewicht durch die eventuell enorme notwendige Höhe zu schwer ist und das Material den auftretenden Spannungen nicht standhält.



In diesen Abbildungen wird ein interessanter Aspekt sichtbar: über die Korrekturfaktoren können bereits aus der Ansicht Rückschlüsse auf den Grundriss und umgekehrt gemacht werden. Achtung die Korrekturfaktoren wurden in diesem Beispiel als zu stark eingestellt.

Als Lösung wird eine Mischung in Form einer evolutionären Optimierungsstrategie angestrebt. Es wird immer der letzte günstigste Träger ausgewählt und per Zufall entweder im Aushöhlungsgrad oder in der Trägerhöhe variiert. Wenn sich die Werte verbessern, dient nun diese neue Brücke als Vorbild für den nächsten Mutationsversuch. Verschlechtern sich hingegen die Werte, wird der Vorgänger erneut verändert.

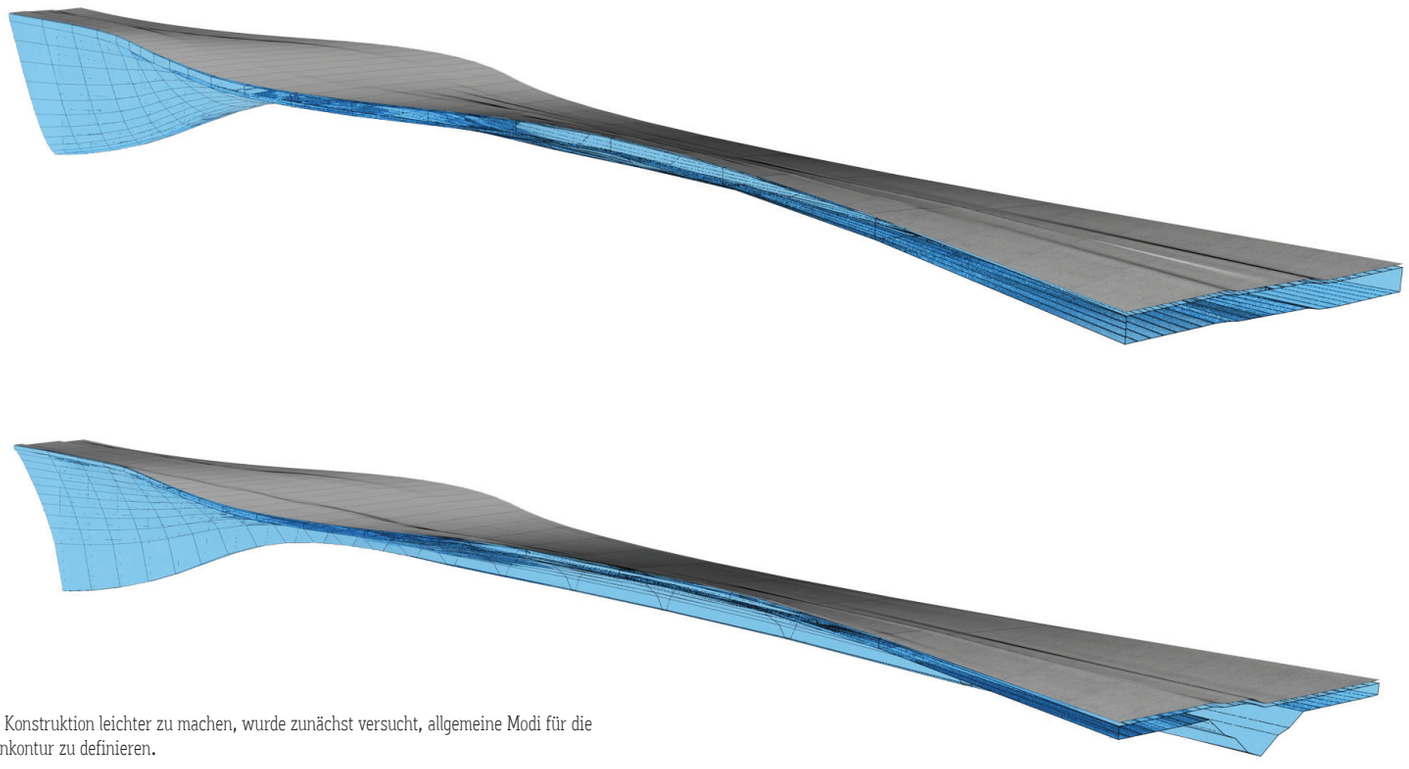
Achtung! Aufgrund der Probleme, die sich bei der Übergabe von Eigenschaftswerten in den nächsten Berechnungsdurchgang ergeben, wurde diese Optimierung nur teilweise realisiert. Diese Probleme resultieren aus den Eigenschaften der gängigen "Mashup-Tools", die bei der Vernetzung der einzelnen Module keine Schleifenbildungen ermöglichen. Dies bedeutet, es wird nicht erlaubt, dass ein Modul ein zweites Modul ansteuert und dabei der Ausgang des zweiten Moduls wieder mit dem Eingang des ersten Moduls verbunden wird.

Dieses Problem wurde in dieser Arbeit nach langen Überlegungen insofern gelöst, als dass alle Berechnungsergebnisse in den Eigenschaften unsichtbarer dreidimensionaler Objekte abgespeichert werden. Aus diesem Grund wurde auch die Option der Berechnungsdurchgänge geschaffen, um mit einem Iterator-Modul das gesamte Netzwerk mehrmals auszuführen. So müssen nun in jedem Durchgang die Ergebnisse außerhalb der Schaltung extern aus den Objekten ausgelesen werden. Dies stellt einen erheblichen Scriptingaufwand dar, weshalb die vorgeschlagene Optimierung nur teilweise realisiert wurde.

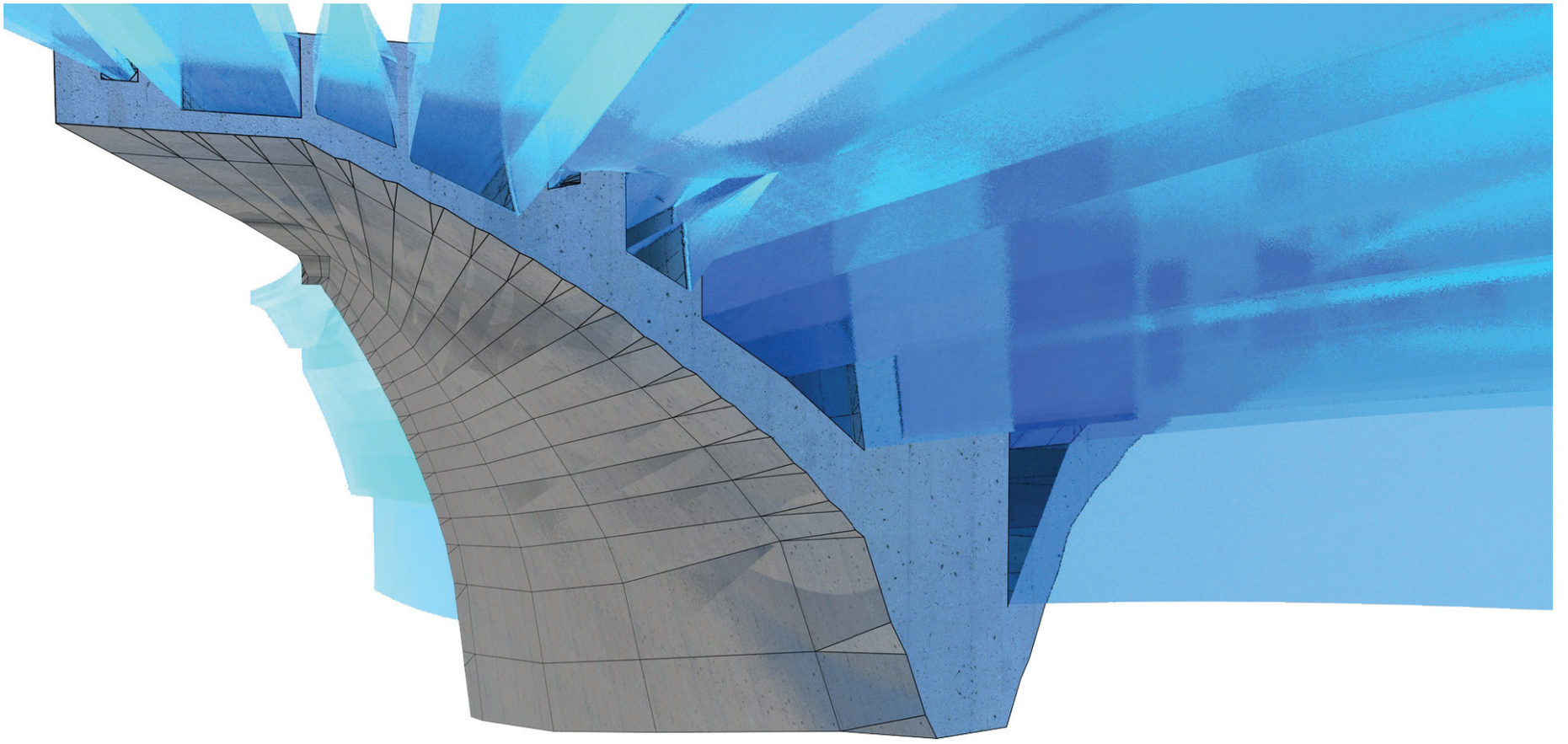
Der Benutzer ist gezungen, aus den generierten Brückenvarianten selbst die günstigste auszuwählen. Da die Werte der Brücken jeweils dargestellt werden, stellt dies keinen großen Aufwand dar.

Flächen und Volumsberechnung:
Da die Kubatur von Freiformen mittels NURBS bzw. gekrümmten Flächen beschrieben werden und eine genaue Flächen- sowie Volumsberechnung aufwändig ist, wurde die Brückenoberfläche in einzelne schiefwinklige Dreiecke sowie auch das Volumen in Pyramiden aufgelöst und für die Gesamtflächen und Volumsermittlung addiert. Dabei enstehen auch im Detail einige Probleme. So stellt sich die Oberfläche beispielsweise als Raster aus verzerrten Vierecken dar, die bei genauerer Betrachtung jeweils hyper-parabolische Flächen beschreibt. Diese Flächen müssten eigentlich mittels Intergral berechnet werden. Wie bereits erwähnt wurden diese Flächen für eine vereinfachte Berechnung in jeweils zwei schiefwinkelige Dreiecke zerlegt, wobei die Entscheidung getroffen werden musste, zwischen welchen Eckpunkten der einzelnen HP-Flächen die Diagonale bei der Aufteilung verlaufen soll. Es hat sich dabei herausgestellt, dass die Art von Annäherung ausreichend ist und der Fehler meist im Promillbereich liegt.

Wasserpegel:
Die Wasserpegelkontrollfunktion könnte als Extramodul vom Structure_ Evaluating-Modul ausgekoppelt werden.

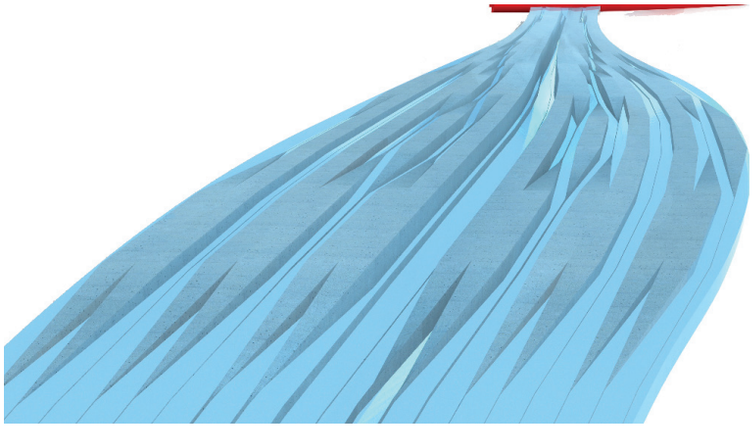


Um die Konstruktion leichter zu machen, wurde zunächst versucht, allgemeine Modi für die Brückenkontur zu definieren.



Materialwahl:
Als zu wählende Konstruktionsmaterialien werden Beton und Stahl zur Verfügung gestellt. Diese wurden etwa in der Nachweisberechnung und in dem Aushöhlungs-Algorithmus differenziert behandelt.

Aushöhlungs-Algorithmus:
Der Aushöhlungsalgorithmus entstand aus der Notwendigkeit, die gesamte Tragwerkskonstruktion leichter zu machen, da diese sich ansonsten aufgrund ihres Eigengewichts zu stark verformt. Die meisten Überlegungen bezüglich der Konstruktion der Brücke sind in die Definition der Querschnitte eingeflossen. Es musste überlegt werden, welche Art von Querschnitt für solche Brücken am geeignetsten erscheint, wobei dieser nach architektonischem Konzept ein homogenes Erscheinungsbild ermöglichen sollte.



Wie bereits im allgemeinen Teil (S.62) hingewiesen wurde, sind geschlossene mehrzellige Querschnittstypen als Grundkonzept gewählt worden. Der Hauptgrund liegt darin, dass bei Freiformen die Tragwerke meist auch auf Torsion beansprucht werden und mehrzellige Querschnitte gegen diese Art von Beanspruchung eine hohe Stabilität aufweisen. Weiters sind diese Querschnitte weitverbreitet; die Hohlräume können beispielsweise für Leitungsführungen oder auch externen Vorspannungen genützt sowie bei großen Projekten für Personen zu Wartungszwecken begehbar gemacht werden.
Es wurde versucht, diese Querschnitte so allgmein wie möglich zu definieren, um dabei material- und konstruktionsspezifische Eigenschaften berücksichtigen zu können. So werden je nach Material unterschiedliche Mindestbauteildicken vergeben. Diese Dicken sind etwa aus herstellungstechnischen Umständen begründet. Dementsprechend können die Profilwände bei der Stahlkonstruktion um einiges zarter als bei einer Betonstruktur dimensioniert werden. Weiters wird beim Aushöhlen der Querschnitte darauf geachtet, dass die Stege aus konstruktiven Gründen im Schnitt exakt senkrecht verlaufen. Dadurch lassen sich die Profile einfacher herstellen, da solche Stegeplatten sich mit wenig Aufwand gießen

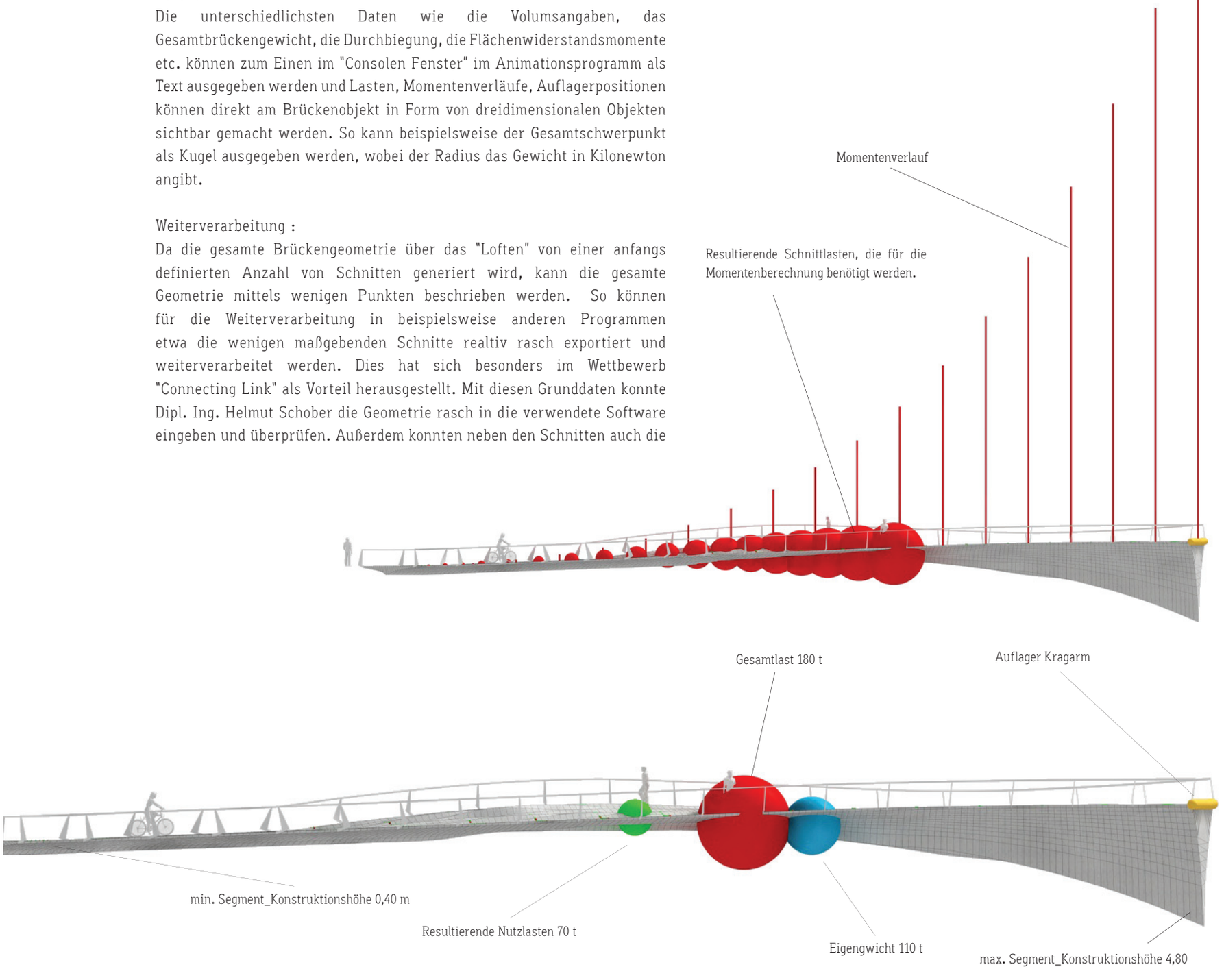
In den zwei Abbildungen wurden Bilder mit einer groben Aushöhlung gewählt, um diese plaktiv leichter sichtbar zu machen. Im Regelfall ist etwa besonders bei der Stahlkonstruktion ein sehr hoher Hohlanteil (Grau), denn die Stege können sehr zart ausgeführt werden.

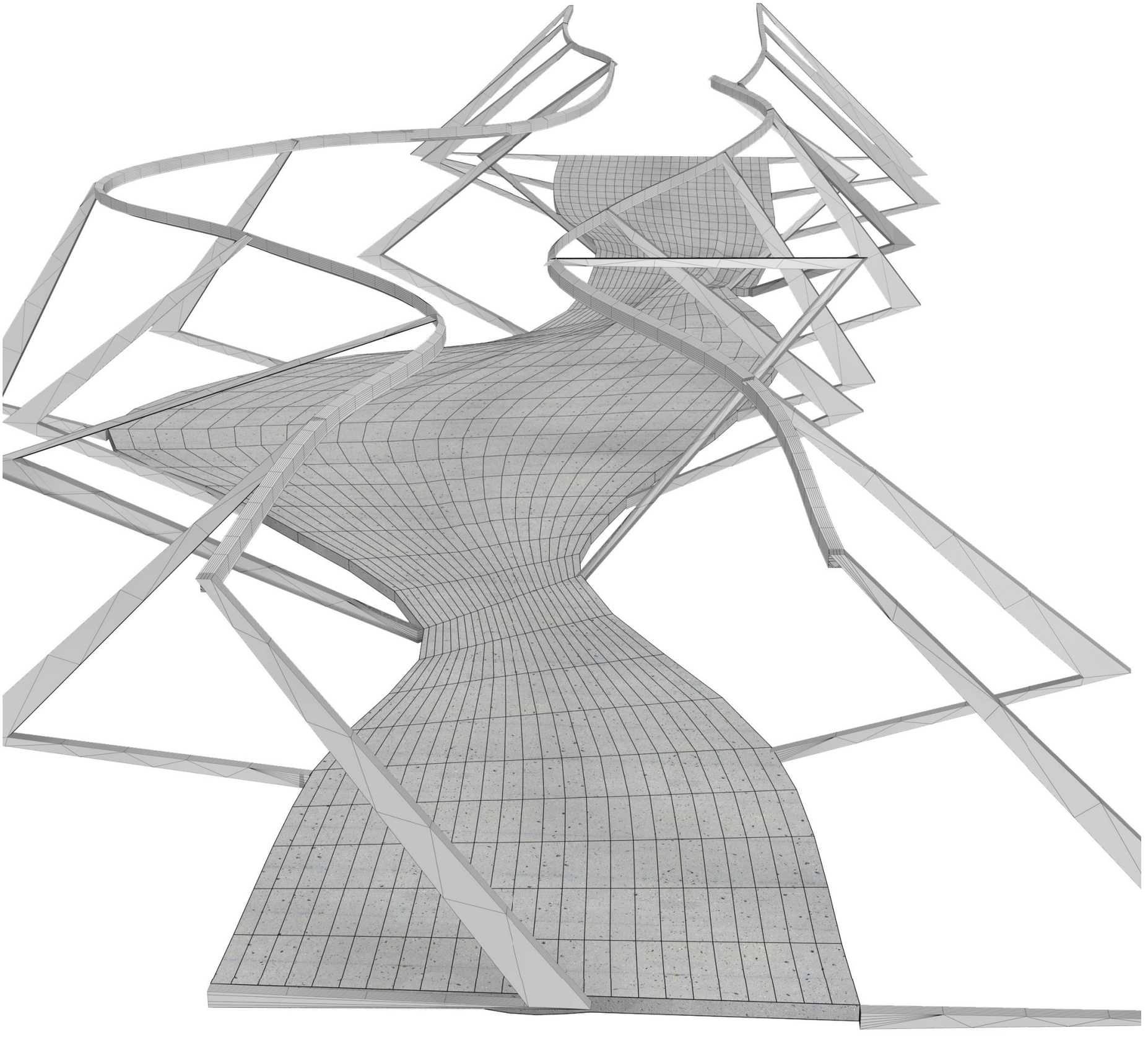
oder oder aus Metallplatten ausschneiden lassen. Außerdem wurden auch Mindesthohlraumbreiten und -höhen vergeben. Müssen diese unterschritten werden, da die gesamte Brücke an dieser Stelle sehr schmal oder zu niedrig wird, werden gleich Vollprofile vergeben, da sich die Herstellung von winzigen Hohlräumen als äußerst aufwändig darstellt. Alle diese Parameter lassen sich innerhalb des Scriptes auch verändern.

Daten:
Die unterschiedlichsten Daten wie die Volumsangaben, das Gesamtbrückengewicht, die Durchbiegung, die Flächenwiderstandsmomente etc. können zum Einen im "Consolen Fenster" im Animationsprogramm als Text ausgegeben werden und Lasten, Momentenverläufe, Auflagerpositionen können direkt am Brückenobjekt in Form von dreidimensionalen Objekten sichtbar gemacht werden. So kann beispielsweise der Gesamtschwerpunkt als Kugel ausgegeben werden, wobei der Radius das Gewicht in Kilonewton angibt.

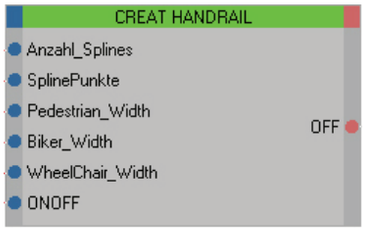
Weiterverarbeitung :
Da die gesamte Brückengeometrie über das "Loften" von einer anfangs definierten Anzahl von Schnitten generiert wird, kann die gesamte Geometrie mittels wenigen Punkten beschrieben werden. So können für die Weiterverarbeitung in beispielsweise anderen Programmen etwa die wenigen maßgebenden Schnitte realtiv rasch exportiert und weiterverarbeitet werden. Dies hat sich besonders im Wettbewerb "Connecting Link" als Vorteil herausgestellt. Mit diesen Grunddaten konnte Dipl. Ing. Helmut Schober die Geometrie rasch in die verwendete Software eingeben und überprüfen. Außerdem konnten neben den Schnitten auch die Systemachsen als Polygonlinie und der Gesamtschwerpunkt als Kugelobjekt etc. exportiert werden.

Natürlich kann die gesamte Brückenoberfläche für die Weiterverarbeitung auch in ein Polygonnetz umgewandelt werden. Dadurch ensteht allerdings eine große Anzahl an Punkten, wodurch eine nachträgliche Bearbeitung der Brückengeometrie relativ aufwändig wird.





Create_Handrail[]



Allgemeine Beschreibung

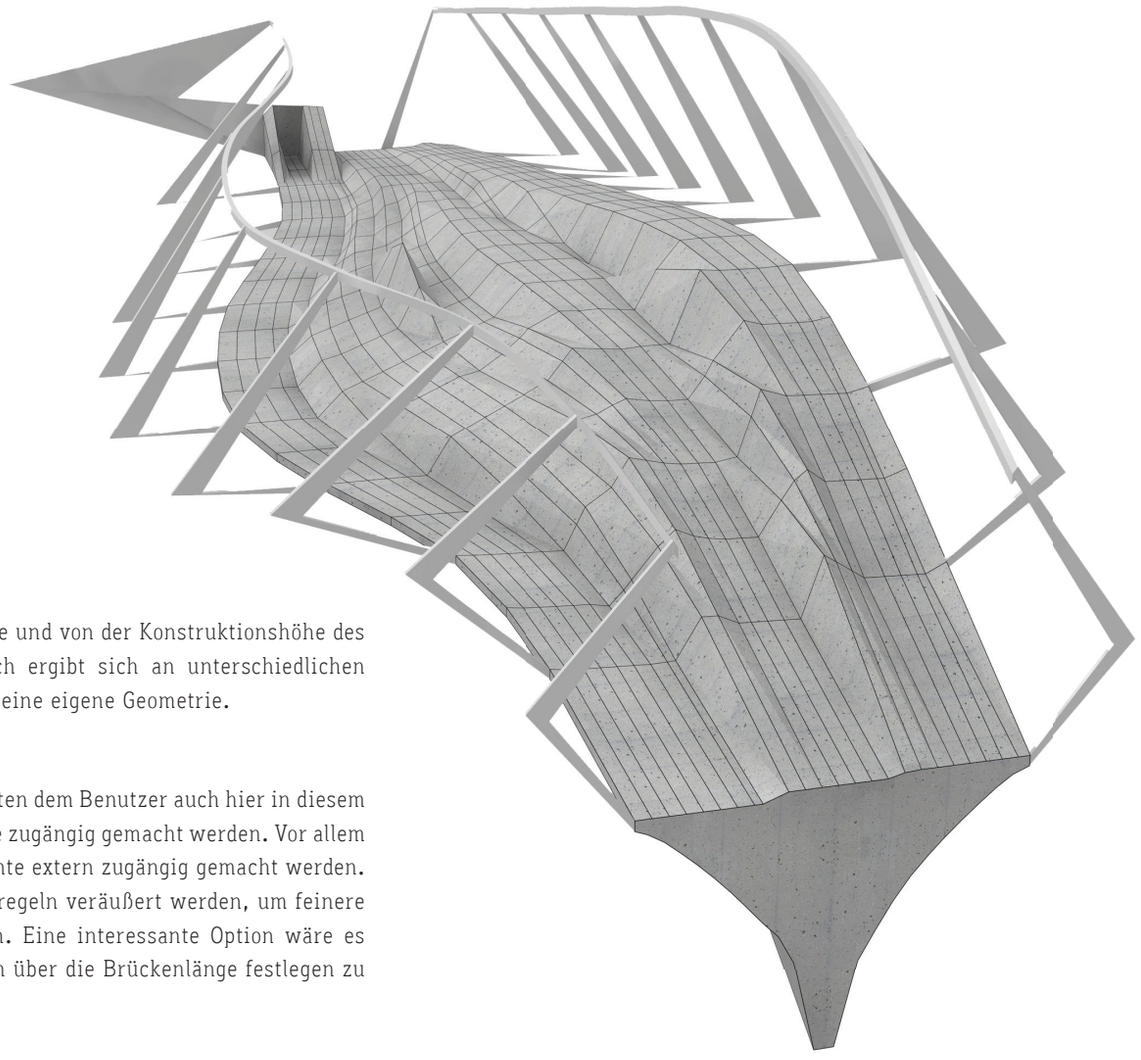
Diese Modul generiert eine von der Brückengeometrie abhängige Absturzsicherung.

Im Entwurf

Da die gesamte Geländergeometrie von der Brückenkubatur abhängig ist, kann diese erst nach der Volumsgenerierung durch das "Structure_ Evaluating-Modul" ausgeführt werden.

Motivation

Damit nicht bei jeder Brückenvariante ein neue Absturzsicherung manuell konstruiert werden muss, wurde ein Grundkonzept festgelegt, um auf Wunsch ein Vorschlag für ein Brückengländer automatisch generieren zu können. Außerdem ist besonders bei freieren Formen eine individuelle Geländeranpassung aufwendig.

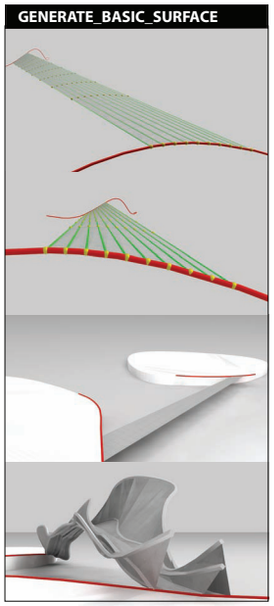


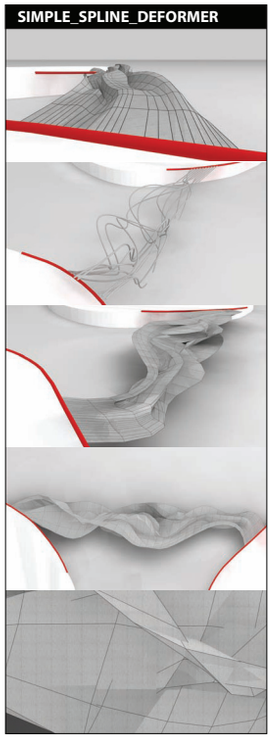
Detail

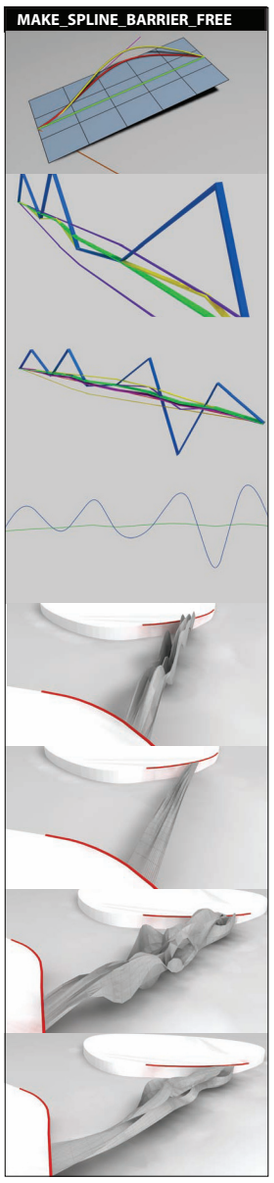
Das Geländer ist von der Brückenbreite und von der Konstruktionshöhe des Brückentragwerkes abhängig. Dadurch ergibt sich an unterschiedlichen Brückenstelle für jede Geländerstütze eine eigene Geometrie.

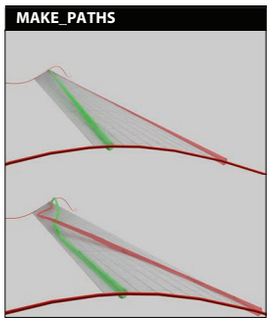
Bemerkung

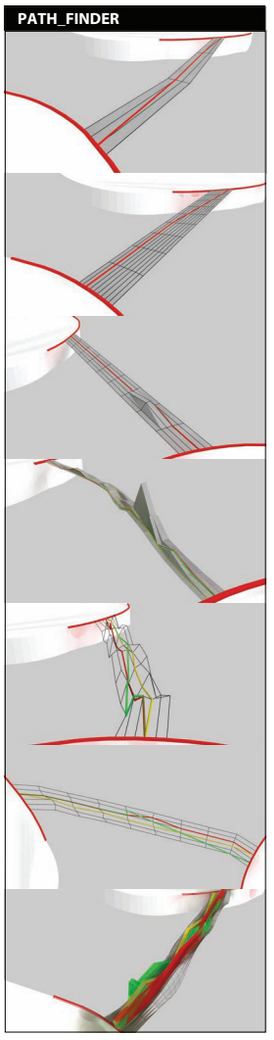
Wie in jedem der anderen Module könnten dem Benutzer auch hier in diesem Node mehr Variablen über das Userface zugängig gemacht werden. Vor allem die Einstellung der Handlaufhöhe könnte extern zugängig gemacht werden. Weiters könnten auch die Proportionsregeln veräußert werden, um feinere Abstimmungen vornehmen zu können. Eine interessante Option wäre es die Anzahl der Geländerunterteilungen über die Brückenlänge festlegen zu können.









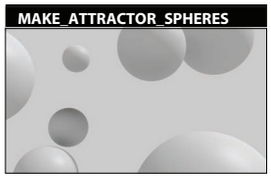


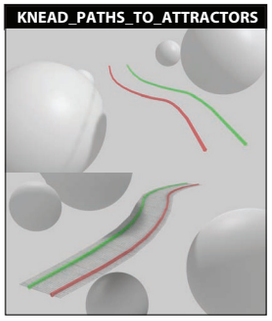
Entwicklungsüberblick

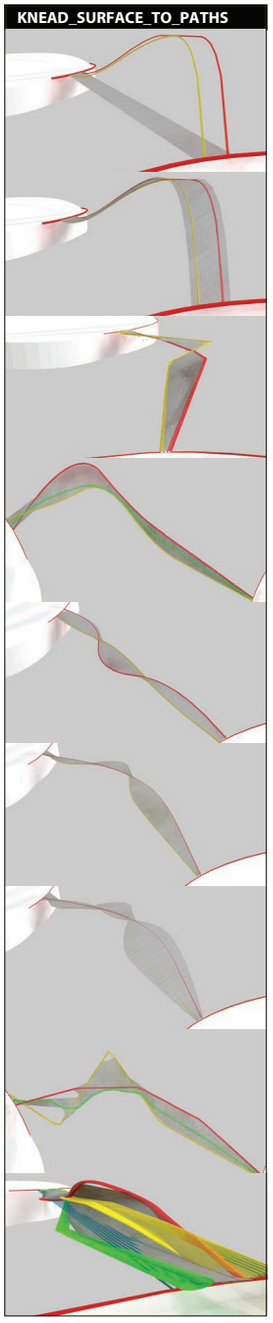
Um abschließend noch einen Überblick über die soeben beschriebenen Module und deren Entwicklung gewinnen zu können, wird an dieser Stelle versucht, anhand der obigen Grafiken skizzenhaft darzustellen, welchen ungefähren Verlauf dieser genommen hat. Dabei muss hinzugefügt werden, dass die Entstehung der einzelnen Module keinesfalls separat zu betrachten ist und nicht streng linear stattgefunden hat. Während des gesamten Prozess' wurden immer laufend wieder Veränderungen in Form von Korrekturen und Erweiterungen vorgenommen. So wurde beispielsweise das "Structure_Evaluating"-Modul bereits nach dem "Make_Barrier_Free"-Modul begonnen. Es wurde bereits der Teil für die Oberflächenberechnung entwickelt. Da jedoch noch keine Pfade für die Verkehrslastannahme vorhanden waren wurde an dieser Stelle pausiert und die zwei "Make_Paths" Module entworfen.

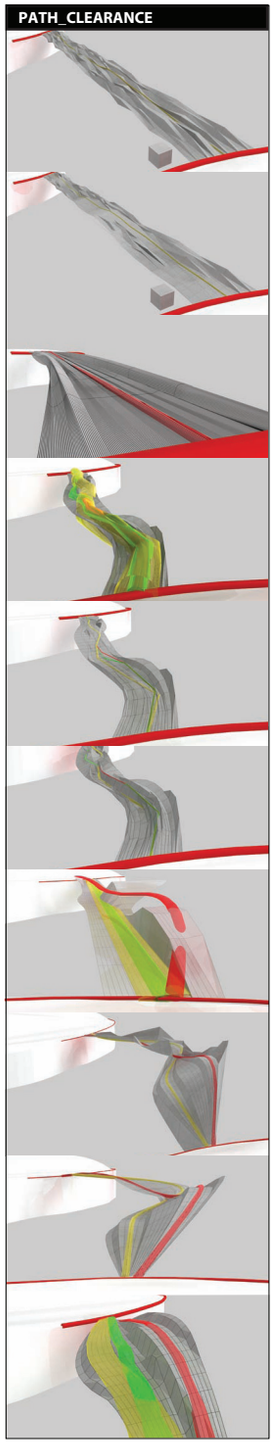
So wird mit den oben dargestellten Abbildungen, in welchen die einzelnen Module von links nach rechts angeführt werden, lediglich eine chronologische Entwicklungstendenz markiert. Dies bedeutet folglich jedoch noch nicht, dass die Module chronologisch dargestellt werden.
Weiters soll die vertikale Länge den Arbeits- bzw. Zeitaufwand andeuten. Einzig das "Structur-Evaluating"-Script konnte layouttechnisch aufgrund Platzmangels nur verkürzt dargestellt werden, da für dieses Modul im Gegensatz zu dem "Path_Clearance"- oder dem "Knead_Surface_To_Path"-Modul ungefähr der vier-, wenn nicht sogar der fünffache Aufwand anfiel.

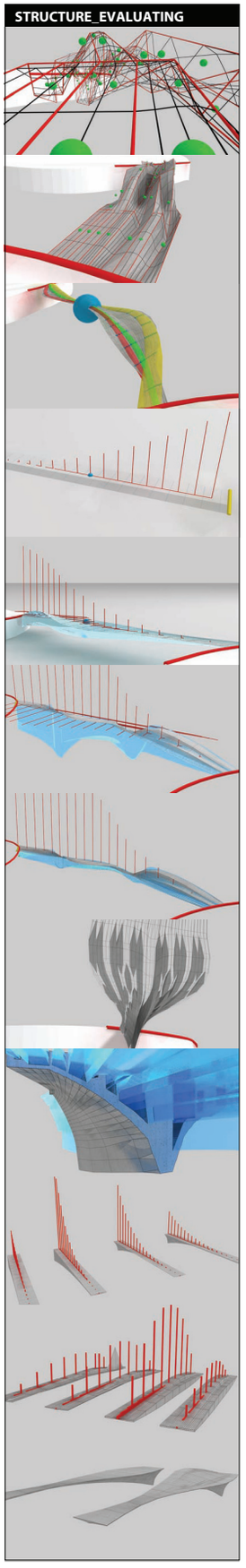
Weiters ist darauf hinzuweisen, dass die gesamte Entwicklung in dieser Arbeit als Experiment betrachtet wurde und daher der Prozess nicht streng determiniert war, sodass diese häufig durch mangelde Scriptingkenntnisse

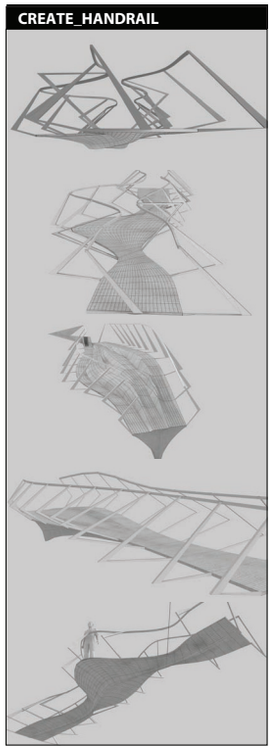




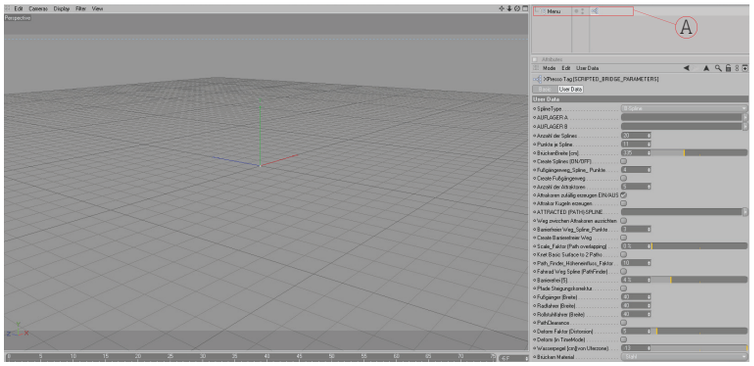




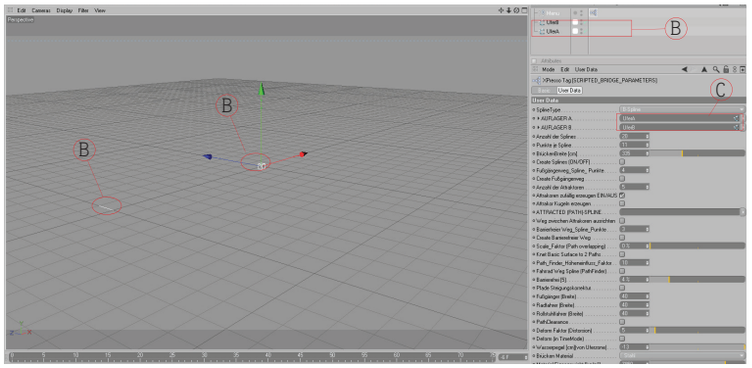




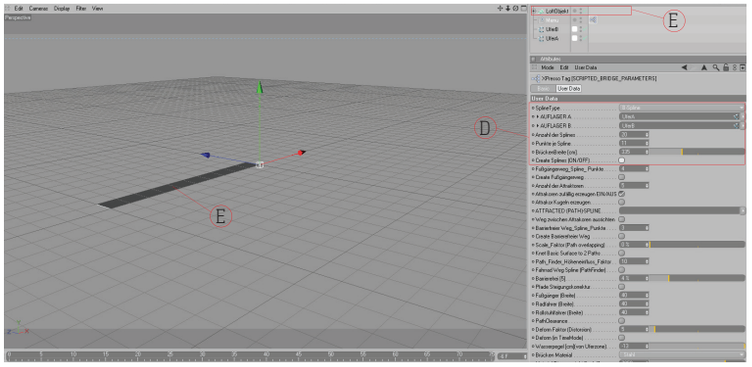
und dem Auftauchen unterschiedlichster Probleme zu einem stark dynamischen, offenen Prozess wurde. Wie sich demzufolge das gesamte Entwicklungskonzept während der Entwurfsphase verändert hat, wird im Appendix auf den Seiten 170 bis 175 dargestellt.



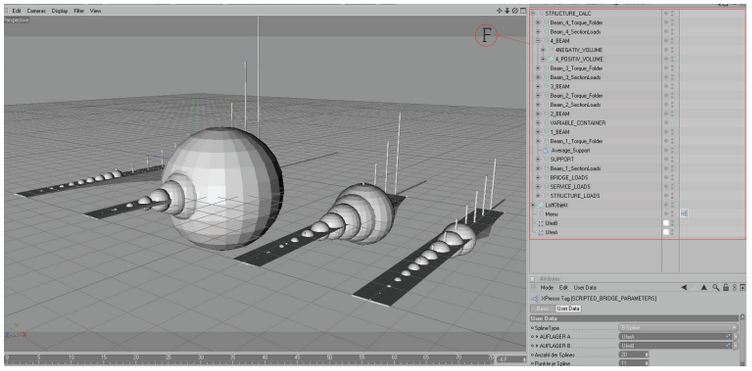
Die leere Szene enthält nur das Xpresso Tag.



Schritt 1: Brückenenden werden definiert.



Schritt 2: Oberfläche bzw. Grundmatrix wird gerneriert.



Schritt 3: Tragwerksberechnung mit vier Berechnungsschritten wurde gestartet.

Generieren einer "Scripted Bridge"

Scripted Bridge V1.0

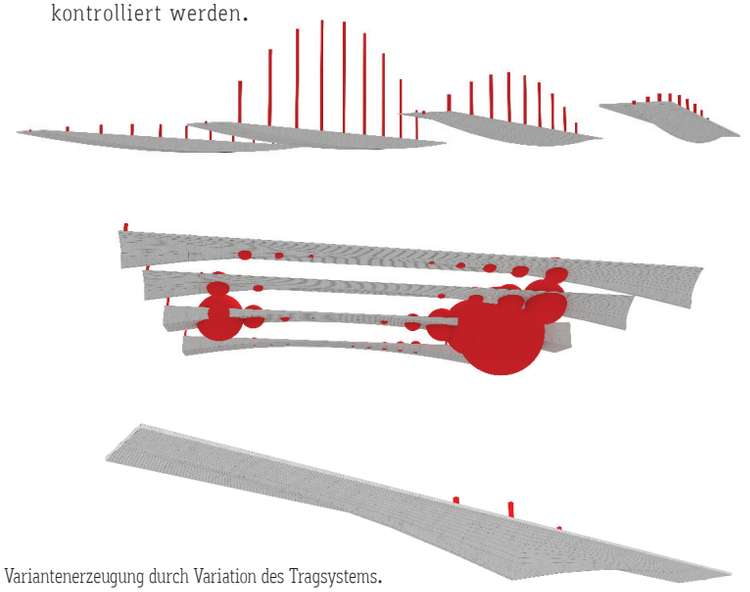
In diesem Beispiel soll der schnellste Weg für eine erste Tragwerksbemessung verdeutlicht werden.

- Als Startbedingung muss lediglich das "XPresso Tag" ("A") im gesamten Projekt vorhanden sein.

- In einem ersten Schritt müssen die Brückenenden definiert werden. In diesem Beispiel wurden die Splines für die Brückenenden in Form von dxf-Dateien aus einer CAD-Software importiert. Diese 3d-Objekte müssen nun mittels "Drag and Drop"-Funktion an die Stelle "C" im User-Interface eingefügt werden. Da in dem dargestellten Beispiel lediglich das Eigengwicht als Entwurfsbedingung der Konstruktion gewählt wird, müssen keine Wege generiert werden.

- Im nächsten Arbeitsschritt muss nur noch die Grundmatrix bzw. Grundknetfläche ("E") im User-Interface in den Punkten im Bereich "D" eingestellt werden.

- In einem letzten Schritt müssen noch die gewünschten Berechnungsschritte ins Interface eingetragen werden, um anschließend den Tragwerksevaluierungsprozess zu starten. Da im gewählten Beispiel vier Berechnungsschritte gewählt wurden, hat das Script nach Betätigen des Startbuttons in der Folge vier verschiedene Varianten von Brücken generiert. Dabei werden in der Entwurfsszene neben der reinen Brückentragwerksgeometrie auch eine Menge anderer 3d-Objekte "F" erzeugt. Weiters können die Berechnungsergebnisse auch in der sogenannten Console in Textform nachverfolgt bzw. kontrolliert werden.



Variantenerzeugung durch Variation des Tragsystems.

```
Das Schnittmoment am Schnitt[9] beträgt162.964264
Flächenträgheitmoment_Y[9]= 747.615295 cm^4.
Das Winkelgewicht an Schnitt[9] beträgt0.001389
Segmentlänge Schnitt[9]:100.000000
Durchbiegung Schnitt[9] : 1.735164cm.
--------------------------------------------------------------------------------
Das Schnittmoment am Schnitt[10] beträgt0.000000
Flächenträgheitmoment_Y[10]= 605.897095 cm^4.
Das Winkelgewicht an Schnitt[10] beträgt0.000213
Durchbiegung Schnitt[10] : 2.537500cm.

Max_Bending_of_Arm_1: 0.000000
Max_Bending_of_Arm_2: 6.666667
Current_Bending_of_Arm_1: 0
Current_Bending_of_Arm_2: 2.537500
Kragarm 1 Länge (Brückenlänge)= 0.000000 cm.
Kragarm 2 Länge (Brückenlänge)= 1000.000000 cm.
Träger 1 darf sich max. um 0.000000 cm durchbiegen!
Träger 2 darf sich max. um 6.666667 cm durchbiegen!
Träger 1 biegt sich 0cm durch!
Träger 2 biegt sich 2.537500cm durch!
Das E-Modul beträgt: 21000kN/cm².
Das durchschnittliche Flächenträgheitsmoment beträgt nilcm^4.
Bending_Arm_1_Efficiency: 0%
Bending_Arm_2_Efficiency: 38.062500%

AUF ZULÄSSIGE SPANNUNGEN ÜBERPRÜFEN
Obergurt ist zu 29.189257% ausgelastet!
Untergurt ist zu 14.415875% ausgelastet!
Obergurt ist zu 32.648113% ausgelastet!
```

```
Untergurt ist zu 14.415875% ausgelastet!
Obergurt ist zu 32.648113% ausgelastet!
Untergurt ist zu 15.865648% ausgelastet!
Obergurt ist zu 37.525627% ausgelastet!
Untergurt ist zu 17.656441% ausgelastet!
Obergurt ist zu 66.492416% ausgelastet!
Untergurt ist zu 21.941221% ausgelastet!
Obergurt ist zu 81.689423% ausgelastet!
Untergurt ist zu 25.508913% ausgelastet!
Obergurt ist zu 93.807907% ausgelastet!
Untergurt ist zu 27.607325% ausgelastet!
Obergurt ist zu 121.463074% ausgelastet!
Untergurt ist zu 31.804142% ausgelastet!
Obergurt ist zu 113.556885% ausgelastet!
Untergurt ist zu 34.090473% ausgelastet!
Obergurt ist zu 80.759880% ausgelastet!
Untergurt ist zu 31.404060% ausgelastet!
Obergurt ist zu 25.765440% ausgelastet!
Untergurt ist zu 14.163925% ausgelastet!
Obergurt ist zu 0.000000% ausgelastet!
Untergurt ist zu 0.000000% ausgelastet!

BRÜCKENGEOMETRIE VERÄNDERN!
Formula_Divident_Arm_1:20.000000 Bending_of_Arm_1_Efficiency: 0
Formula_Divident_Arm_2:20.000000 Bending_of_Arm_2_Efficiency: 0.380625
MODIFY_PATHS_ON=True

Die Wegeführung wird verändert!
KNEAD_TO_PATHS_ON=True
PATH_CLEARANCE_ON=True
```

Abbildungen XX: Kleiner Ausschnitt der Berechnungsergebnisse in der Console.

BRÜCKE

statisches System: Kragarmsystem
Material Stahl: 7850 kg/m3
Länge: 10 m
Breite: 1 m

Schritt 1:
Trägerhöhe wurde konstant angenommen, wodurch das Volumen für die erste Berechnung nicht korrekt ist.

Schritt 2:
Momente werden nun mit richtigem Volumen berechnet.

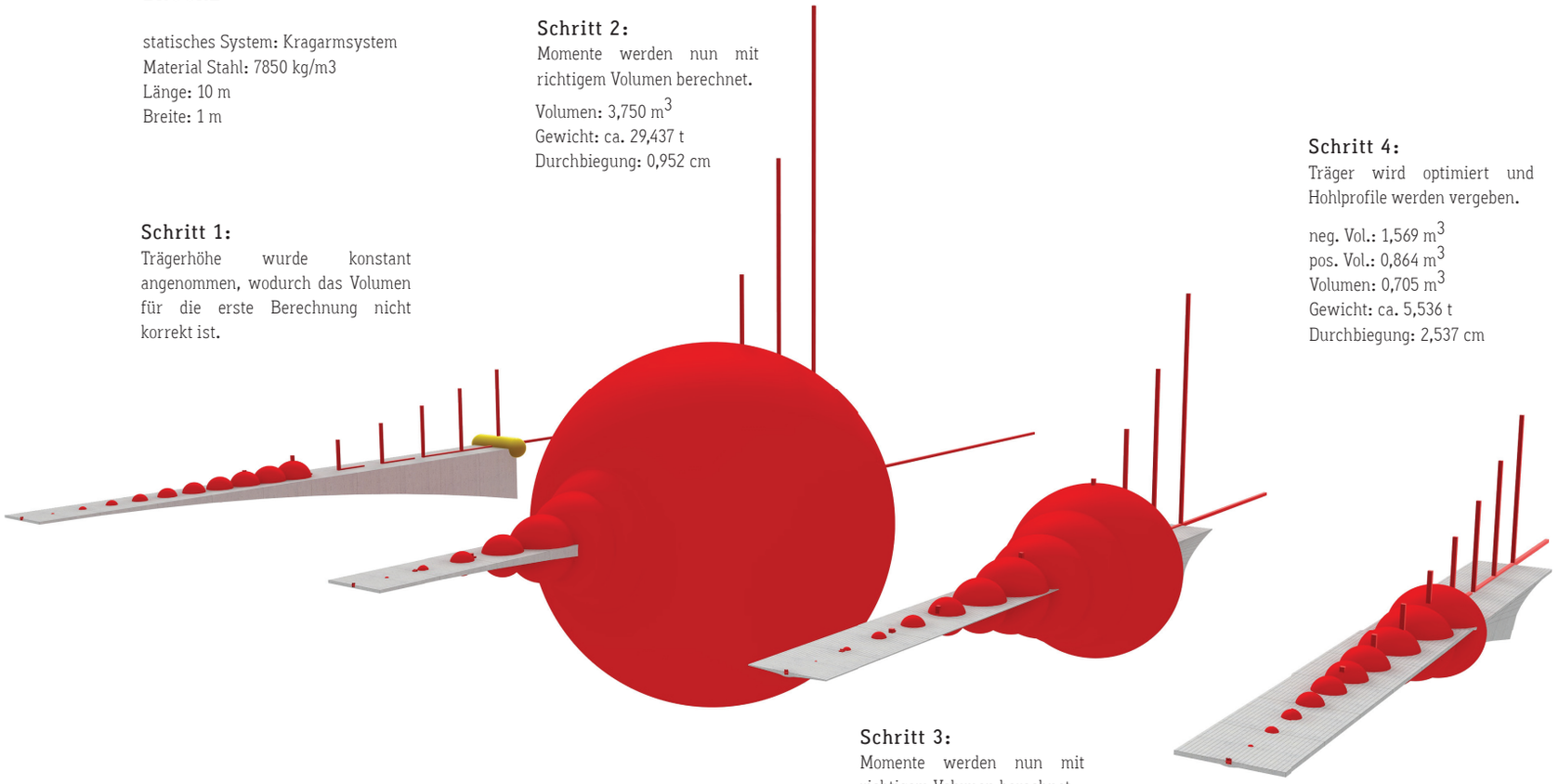
Volumen: 3,750 m^3
Gewicht: ca. 29,437 t
Durchbiegung: 0,952 cm

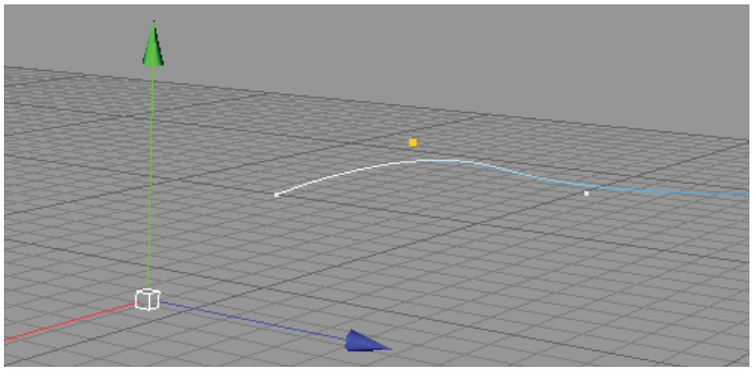
Schritt 3:
Momente werden nun mit richtigem Volumen berechnet.

Volumen: 1,569 m^3
Gewicht: ca. 11,942 t
Durchbiegung: 1,514 cm

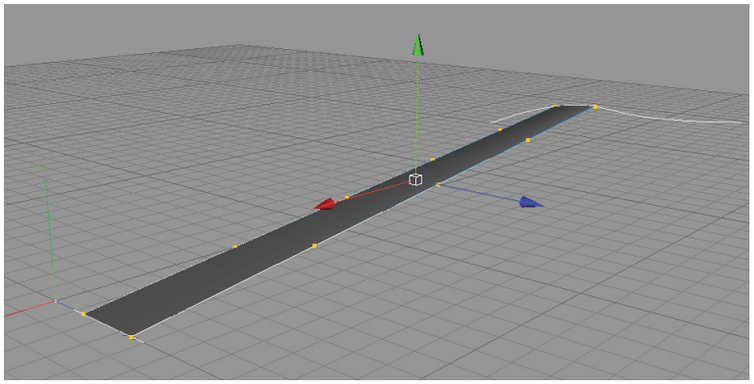
Schritt 4:
Träger wird optimiert und Hohlprofile werden vergeben.

neg. Vol.: 1,569 m^3
pos. Vol.: 0,864 m^3
Volumen: 0,705 m^3
Gewicht: ca. 5,536 t
Durchbiegung: 2,537 cm

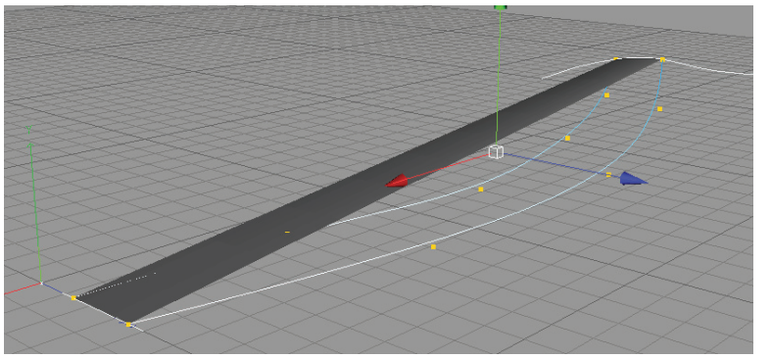




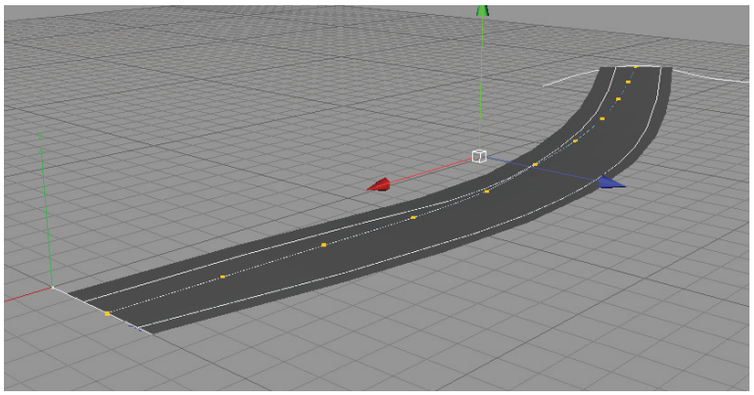
Schritt 1: Brückenenden werden manuell definiert.



Schritt 2: Nachdem Oberfläche bzw. Grundmatrix erzeugt wurde werden nun die Wege generiert.



Schritt 3: Manipulation der Kontrollpunkte der Wege.



Schritt 4: Nachdem die Oberfläche mit dem Knead_to_Path-Modul den Pfaden angepasst wurde, ist zusätzlich ein dritter Pfad, der den kürzesten Weg wischen den Brückenenden anzeigt automatisch erzeugt worden

Scripted Bridge V2.0

Im zweiten Beispiel soll bereits die Brückengeometrie mittels der einzelnen Wege verzerrt werden und mehrere Berechnungsschritte mit weiteren Optimierungsvorschlägen gerneriert werden.

- Wie schon im ersten Beispiel ist auch hier die Grundvoraussetzung das Vorhandensein des "XPresso Tag", welches die gesamten Module und Scripten enthält, in der Entwurfsszene.

- Im Unterschied zum ersten Beispiel wurden in diesem die Splines für die Brückenenden im Programm erstellt und verzerrt. Erst danach wurde die Grundmatrix erzeugt.

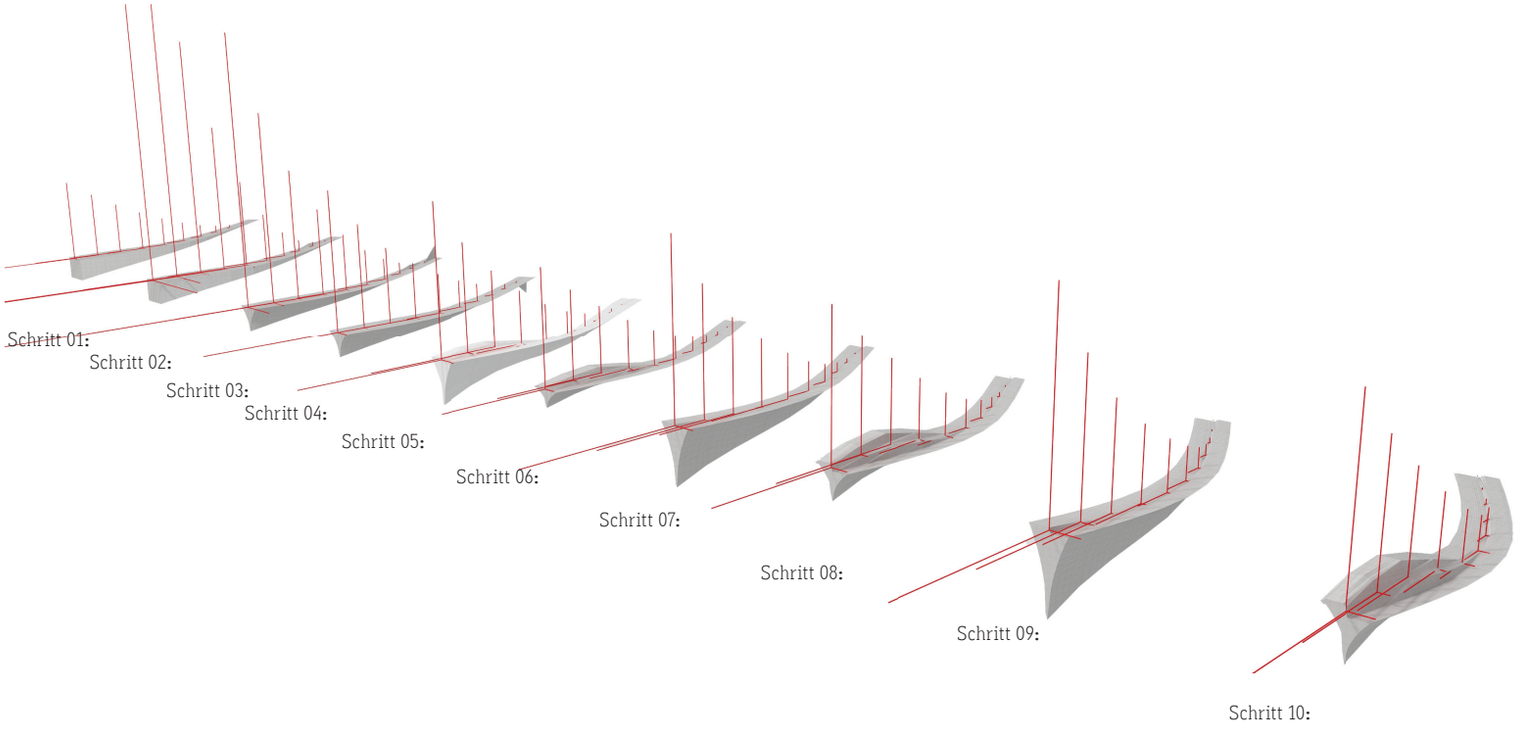
- Mittels Eingabe im User-Interface wurden nun zwei Wege (Splines) mit einer jeweils unterschiedlichen Anzahl an Kontrollpunkten generiert. Danach wurden diese zu einer Bogenform verzerrt.

- Um schließlich die Brückenoberfläche zwischen den aktuellen Wegesplines aufzuspannen, musste nun die Option "Knet Basic Surface to 2 Paths" (Knead_to_Paths-Modul) aktiviert werden.

- Anschließend wurde noch eine dritter Weg mit dem Pathfinder Modul generiert. Dieser gibt die kürzeste Distanz über zwischen den Brückenenden an.

- Zusätzlich wird diesmal auch ein Wasserspiegel nicht allzuweit unter den Auflagerpunkten angenommen.

- Daraufhin kann die Berechnung gestartet werden, wobei dafür zehn Berechnungsschritte gewählt wurden. Als Ergebnis werden einige Objekte erzeugt, wobei dieses Mal auch generelle Veränderungsvorschläge für die Brückengeometrie gemacht werden. So werden beispielsweise auch die Wegeführungen verändert, indem etwa versucht wird, die Brücke im Auflagerbereich - wenn möglich - anzuheben, da andernfalls die Brückenkonstruktion ins Wasser ragen würde. Aufgrund der Steigungsbeschränkungen der Startpositionen der Wege an den Brückenenden ist eine Anhebung der Konstruktion nicht direkt am Auflager möglich. In späteren Varianten schlägt das Programm auch andere Startpositionen vor.



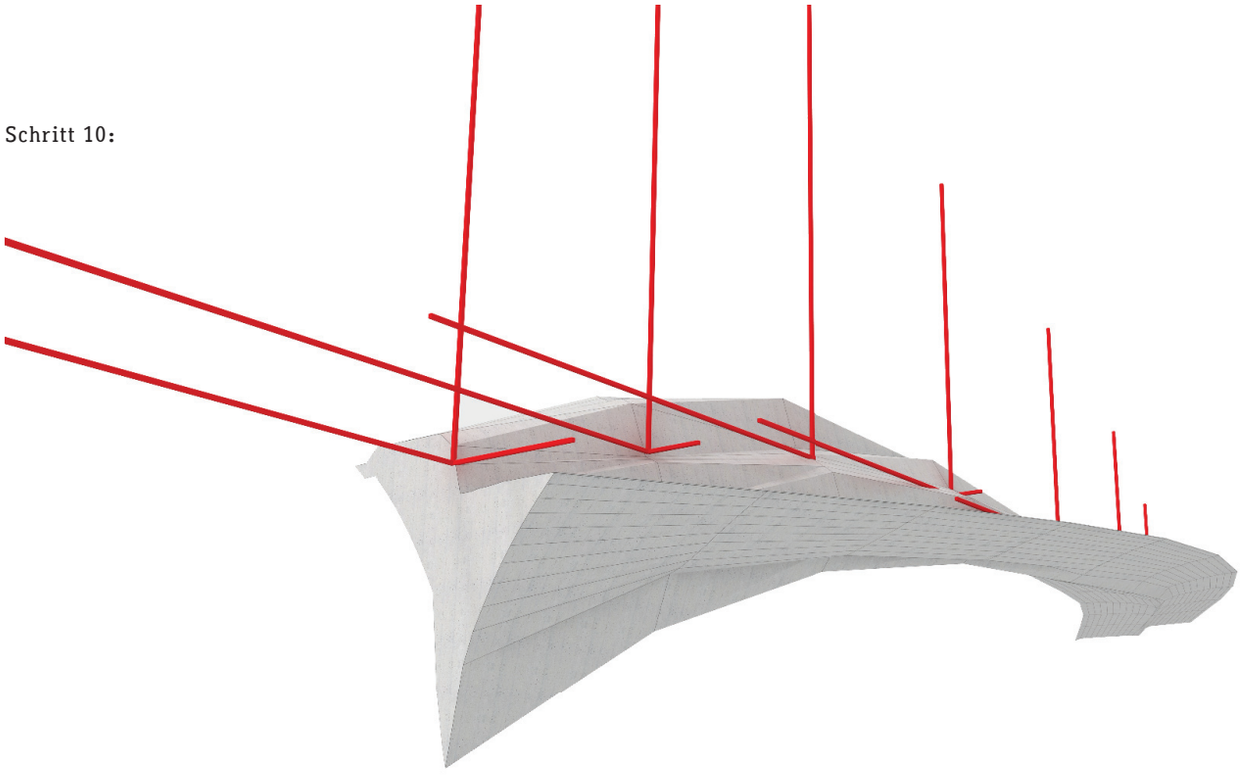
Schritt 07:
pos. Vol.: 66,726 m^3
neg. Vol.: 48,347 m^3
Volumen: 18,379 m^3
Gewicht: ca. 187,341 t
Durchbiegung: 7,610 cm (ca. 40,7%)

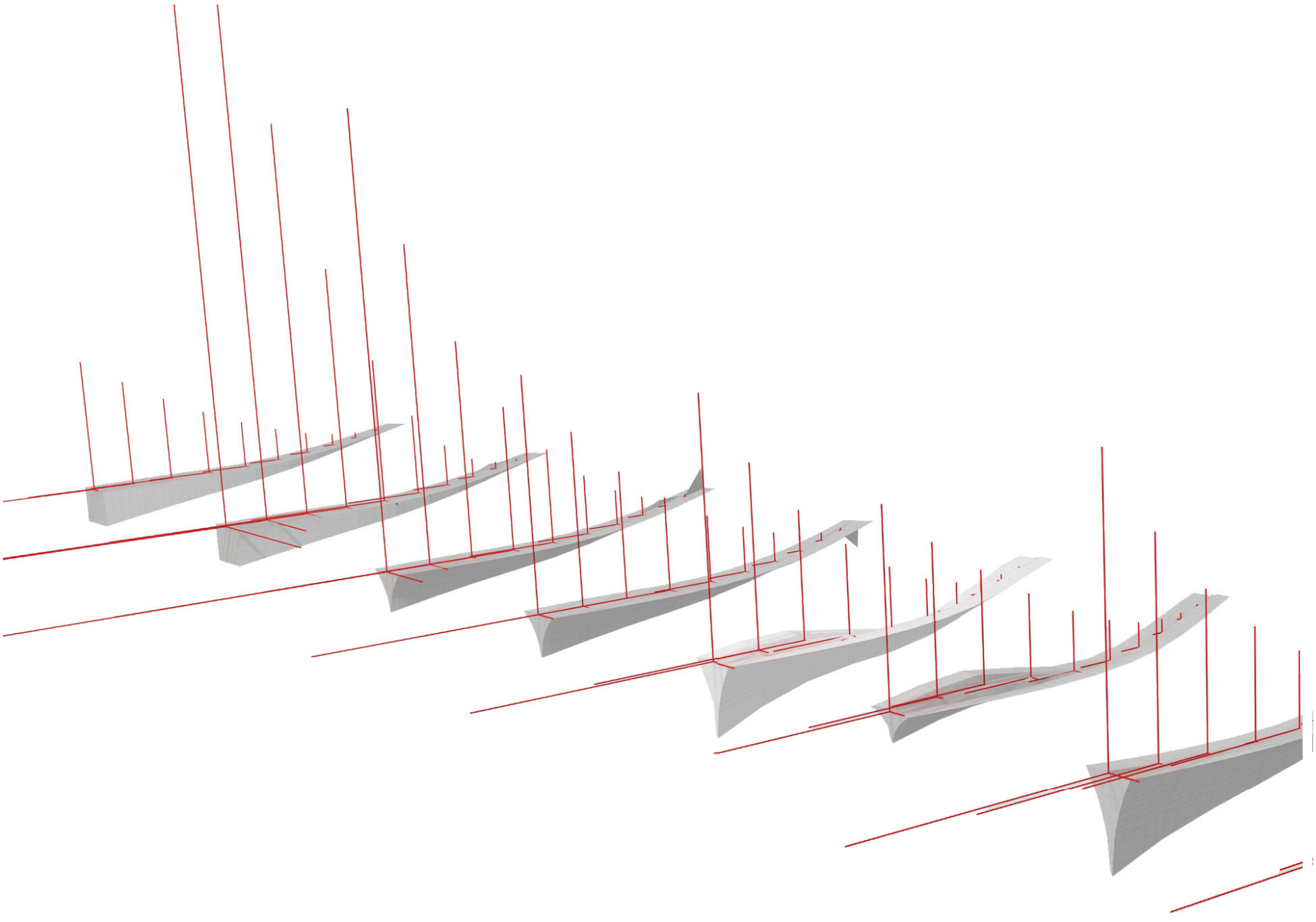
Schritt 08:
pos. Vol.: 25,959 m^3
neg. Vol.: 13,488 m^3
Volumen: 12,470 m^3
Gewicht: ca. 142,336 t
Durchbiegung: 45,301cm (ca. 242%)

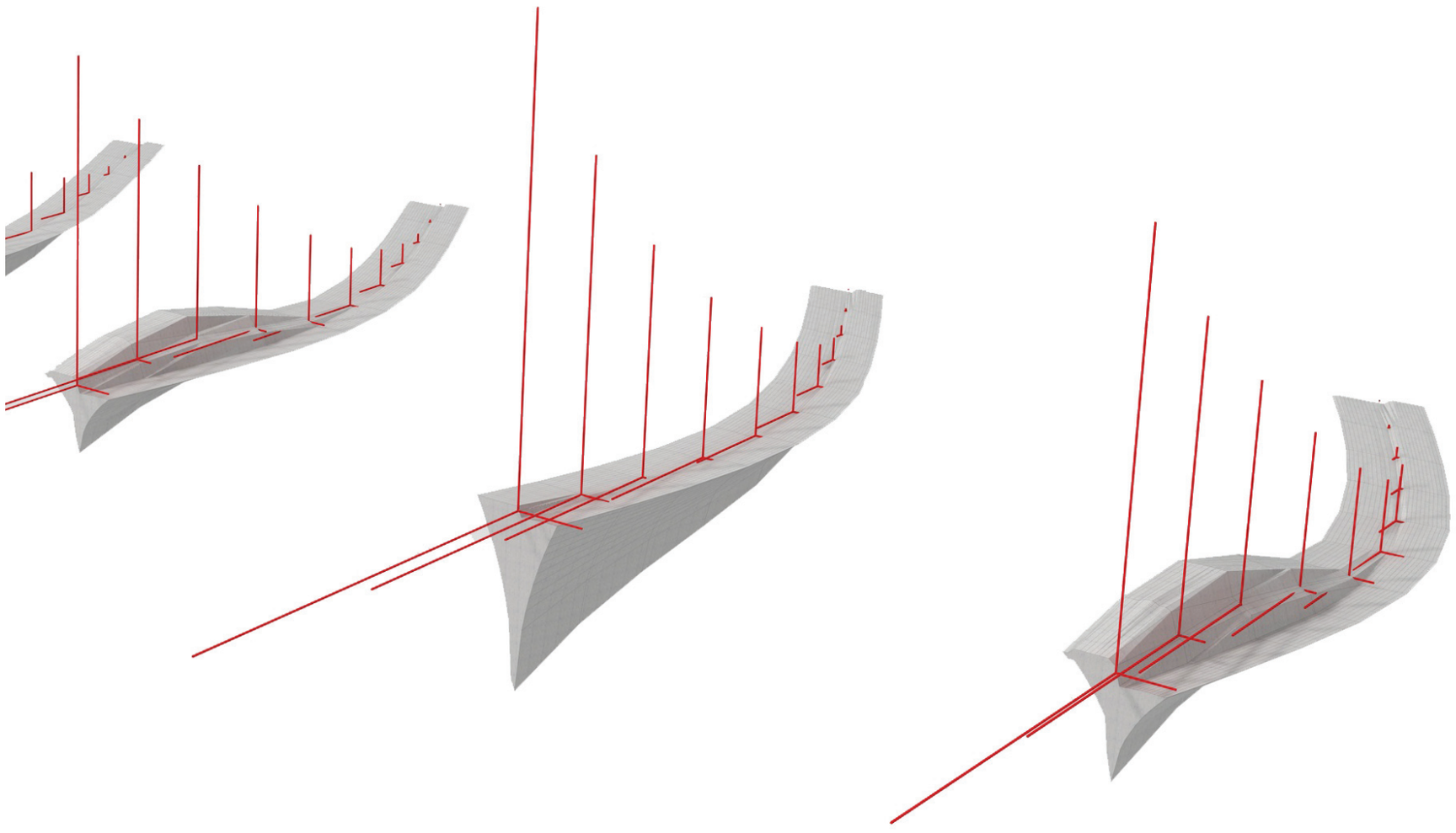
Schritt 09:
pos. Vol.: 76,361 m^3
neg. Vol.: 52,874 m^3
Volumen: 23,487288 m^3
Gewicht: ca. 227,244 t
Durchbiegung: 8,153 cm (ca. 43%)

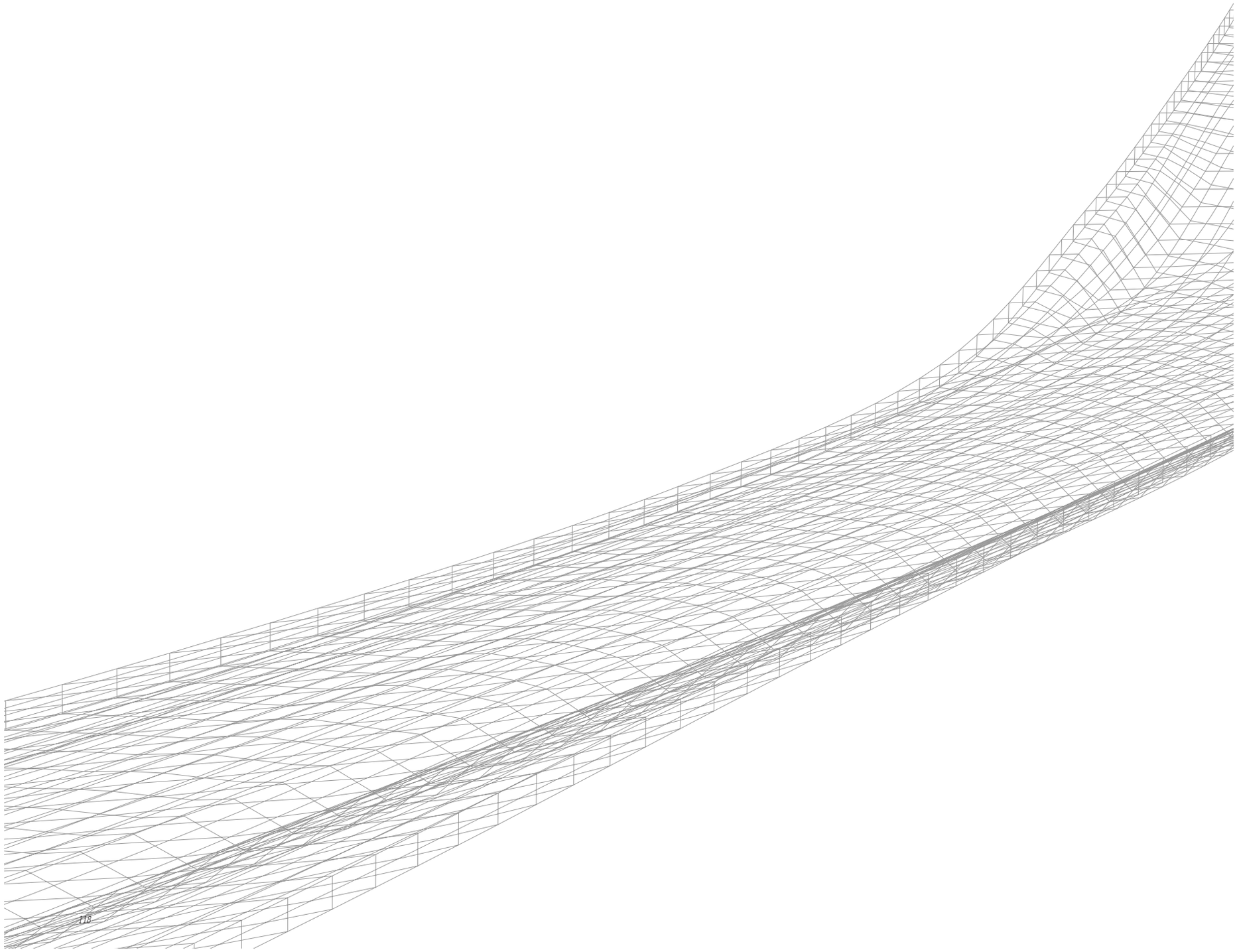
Schritt 10:
pos. Vol.: 30,061 m^3
neg. Vol.: 14,639 m^3
Volumen: 15,422 m^3
Gewicht: ca. 165,260 t
Durchbiegung: 40,354 cm (ca. 216%)

Schritt 10:

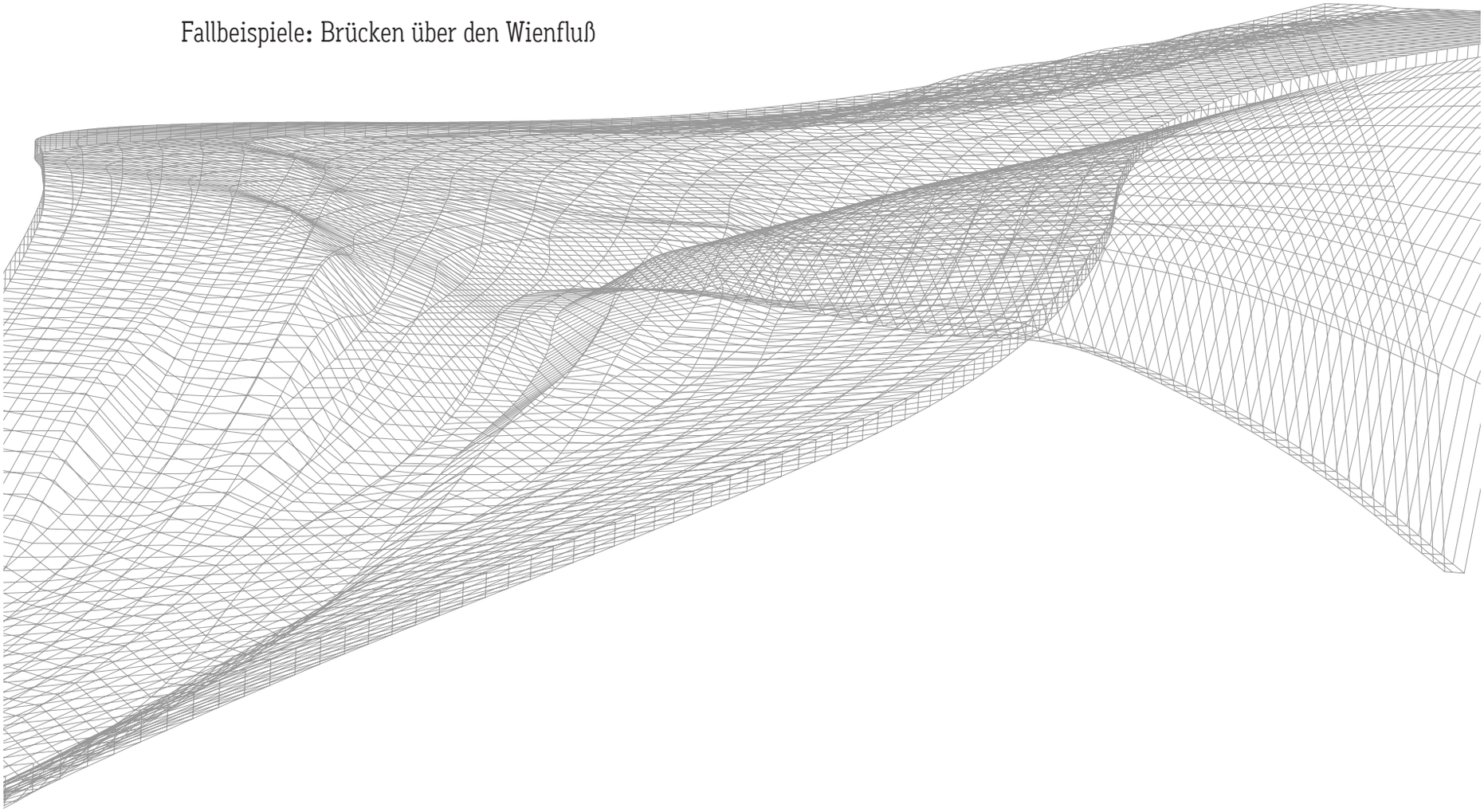








Fallbeispiele: Brücken über den Wienfluß




Uniqa Tower
Untere Donaustraße
Donaukanal
Aspernbrücke
Hermannpark
scripted bridge
Urania
Obere Weißgerberstra
Uraniastraße
Radezkybrücke
Julius Raab Platz
Wienfluß

LAGEPLAN Concrete Student Trophy
M 1:1000

Fallbeispiele: Brücke über den Wienfluss

Allgemeines

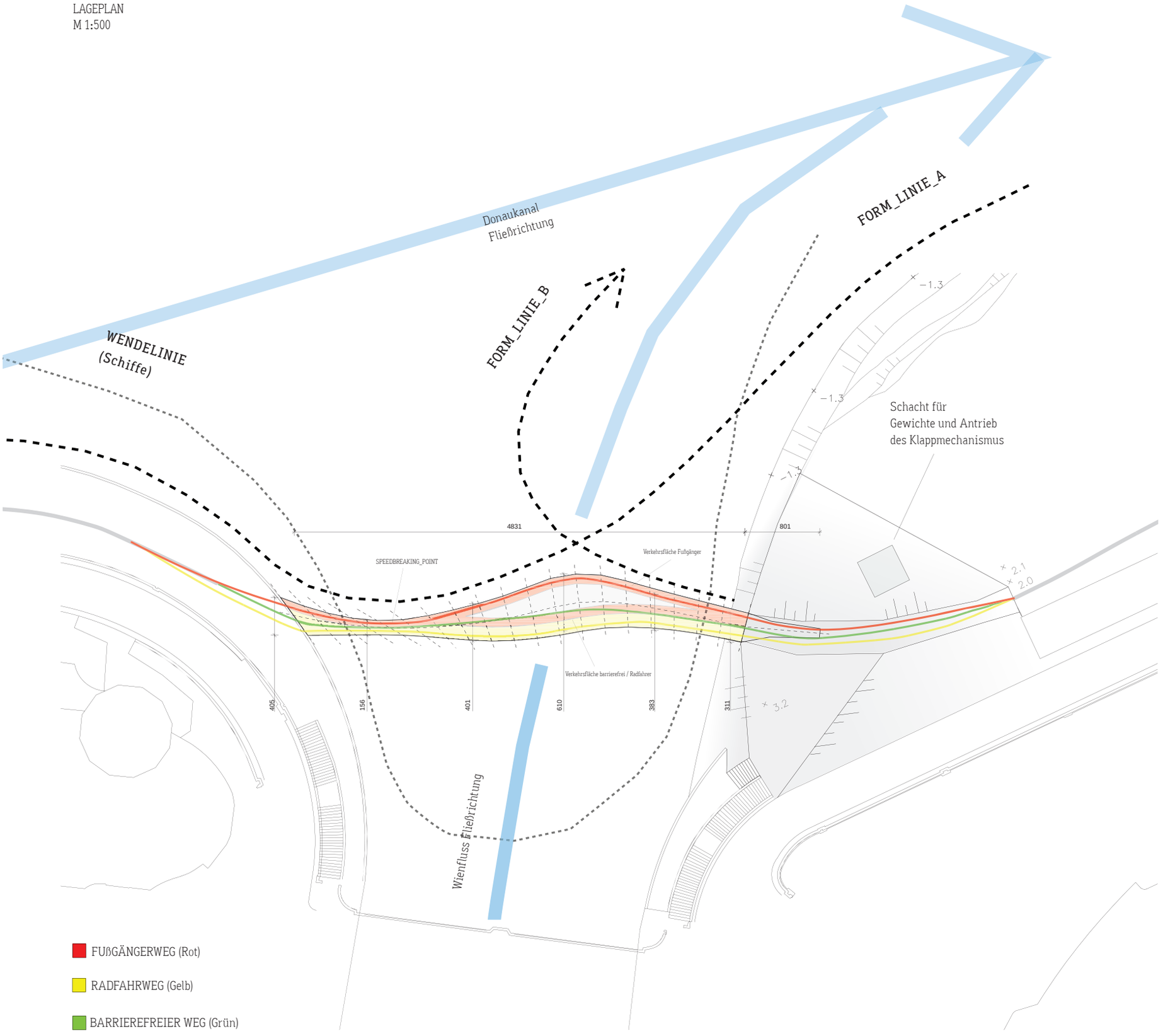
Es wurde bereits mehrfach ausgeführt, dass für die Entwicklung des parametergesteuerten Brückengenerierungsprozesses die Ausschreibungen zweier Wettbewerbe als Rahmenbedingungen verwendet wurde/-n. Die einzelnen Module bzw. die gesamte Schaltung wurden ständig in Bezug auf die Wettbewerbe getestet und weiterentwickelt. So wurden die Wettbewerbsbrücken mittels des entwickelten "Mashup-Systems" entworfen, wobei sich dieses jeweils in einem unterschiedlichen Entwicklungsstadium befand.

Die "Concrete Student Trophy 2009" wurde als erster Wettbewerb wesentlich für die Parameter- sowie Grundkonzeptfindung verwendet. Zwar wurden auch bereits Vorschläge für eine Brückenkubatur sowie für mögliche Brückenprofile gemacht, doch konnten noch keine Ergebnisse bezüglich der Querschnittswerte oder der auftretenden Spannungen und Durchbiegungen an verschiedenen Stellen des Systems ausgegeben werden.

Der Realisierungswettbewerb "Connectiong Link" diente folglich zur Weiterentwicklung der im Zuge des ersten Wettbewerbes nicht fertiggestellten Funktionen und darüber hinaus zu einer praktischen Überprüfung der Ergebnisse in einem Ingenieurbüro. Weiters konnte festgestellt werden, inwiefern sich die produzierten Daten für eine Weiterverarbeitung auch mit anderen CAD-Programmen eignen.

Das architektonische Entwurfs- und Formenkonzept für die Wettbewerbsentwürfe enwickelte sich einerseits aus den Ideen und Formvorschlägen, die während der Entwicklung des parametrisierten Prozesses auftauchten und andererseits aus der Abstrahierung örtlicher Gegebenheiten.

LAGEPLAN
M 1:500



Concrete Student Trophy

Allgemeines

Vorweg ist festzuhalten, dass das architektonische Entwurfskonzept grundsätzlich bei beiden Wettbewerben annähernd ident ist und deshalb im Folgenden für beide wiedergegeben wird. Bei den Ausführungen zum Wettbwerb "Connecting Link" sollen aus diesem Grund lediglich dessen Ergänzungen beschrieben werden.

Entwurfskonzept

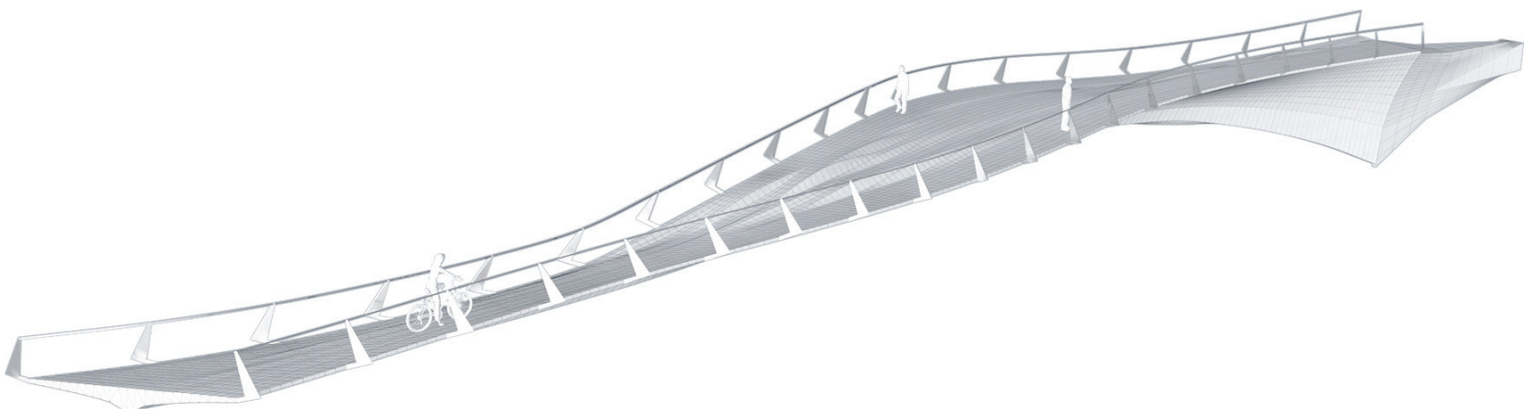
Der Standort befindet sich in einem historischen Umfeld, das aus Denkmalschutzgründen einen sensiblen Umgang mit dem Bestand erfordert. Durch die Fließgewässer, die unterschiedlichen Wasserpegel, die Klappbrücke sowie den Schiffsverkehr ist das Entwurfsareal durch Begriffe wie Veränderung, Bewegung und Transformation geprägt. Es herrscht ein Kontrast zwischen urbaner Dichte und der großzügigen Weite der Flussmündung.

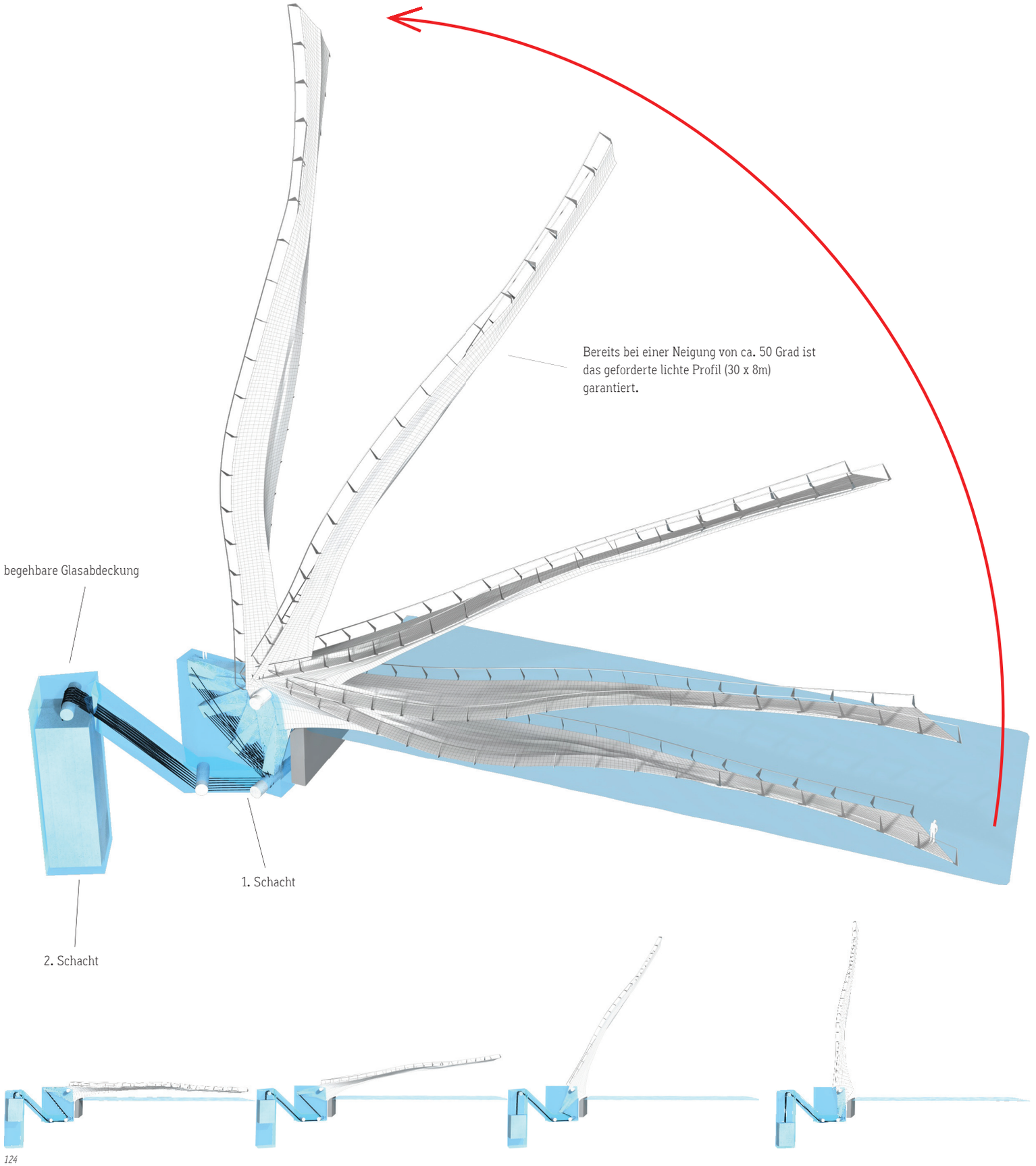
Diese Randbedingungen wurden in den Entwurfsgedanken aufgenommen und spiegeln sich in der Wahl einer dezenten, dynamischen Brückenformgebung wider. Im begehbaren Zustand stellt der Entwurf die Dominanz der Urania einerseits sowie der Radetzkybrücke andererseits nicht in Frage. Bewusst wird auf eine zweiteilige Ausführung des Brückenkörpers verzichtet, um Interventionen auf Uraniaseite möglichst gering zu halten. Sämtliche mechanische Einrichtungen zum Klappen der Brücke werden auf Hermannparkseite konzentriert (Denkmalschutz). Durch die Spannweite von annähernd 50m wird im aufgeklappten Zustand zumindest temporär mit der Höhenentwicklung und - was gleichbedeutend ist - mit den Wertigkeiten der umgebenden Gebäude (Urania, Uniqa-Tower, Finanzministerium) gespielt. Es entsteht eine temporäre Skulptur, die von ihrer Umgebung als Landmark wahrgenommen wird.

Formgebung

An der Kragarmeinspannseite (Hermannparkufer) neigt sich die Brücke in Fließrichtung des Wienflusses (FORM_LINIE_B), damit bei extremem Unwetter und Hochwasser kein Treibgut am Schwert (Kragarmsbeginn) hängenbleiben kann. Weiters soll dadurch das physikalische und ästhetische Gewicht betont werden, das vom Fluss zunnächst mitgerissen wird. In den restlichen zwei Dritteln der Brückenlänge wird mit der Fom ein gedachter weicher Mündungsverlauf nachempfunden (FORM_LINIE_A).

In der Seitensansicht hebt sich die Brücke zur Brückenmitte hin etwas weiter vom Wasser ab und nimmt eine flache Bogenform an. Dies hat zur Folge, dass während des Klappvorganges das geforderte Mindestraumprofil (Lichte) für die Schiffsdurchfahrt bereits bei einem Klappwinkel von 50 Grad erreicht wird. Dieser Effekt wird zusätzlich durch die annähernd konvexe Form des Kragarms verstärkt.




Bereits bei einer Neigung von ca. 50 Grad ist das geforderte lichte Profil (30 x 8m) garantiert.
begehbare Glasabdeckung
1. Schacht
2. Schacht

Klappmechanismus

Damit die Urania und die Radetzkybrücke in ihren Ansichten möglichst nicht durch Konstruktionen verdeckt werden, ist auf abgespannte Zugseilkonstruktionen und hohe Pylone verzichtet worden. Der gesamte Klappmechanismus befindet sich unterirdisch auf der Hermannparkseite. In Verlängerung der Brücke wurde unter der Erde ein Hebelarm ausgebildet, an dem über eine Seilkonstuktion ein Gegengewicht angreift. Über dieses Gewicht wird die Brücke im Gleichgewicht gehalten.

Der Vorteil dieser Art von Konstruktion liegt einerseits in seiner unterirdischen Durchführbarkeit und in der Unempfindlichkeit gegen mögliche Umwelteinflüsse (Wasser). Der Drehpunkt der Brücke ist so gewählt, dass bis zum höchsten schiffbaren Wasserstand kein Wasser in den 1. Schacht eindringen kann.

Es gibt 2 Schächte, die zu realisieren sind:

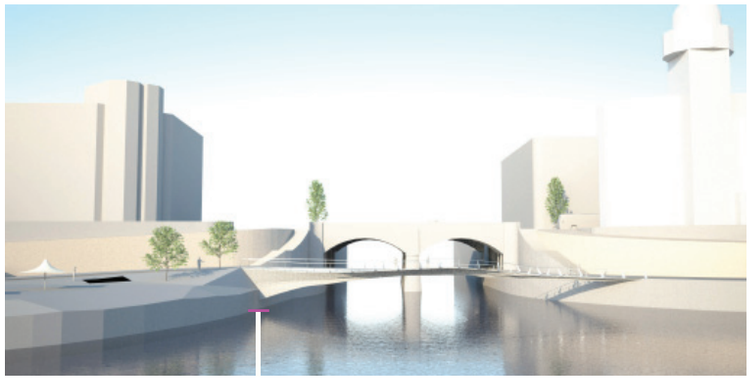
1.) Der erste tiefe schmale Schacht befindet sich unmittelbar hinter dem Auflager und ermöglicht die Drehbewegung der Brückenverlängerung (Hebelarm ca. 8m Länge). Dieser Schacht kann bei extremen Wasserständen ohne Konsequenzen auf den Mechanismus geflutet werden.

2.) Die Grundposition des 2. Schachtes ist flexibel und kann verändert werden. In diesem Schacht befindet sich das Gegengewicht und der Antriebsmotor.

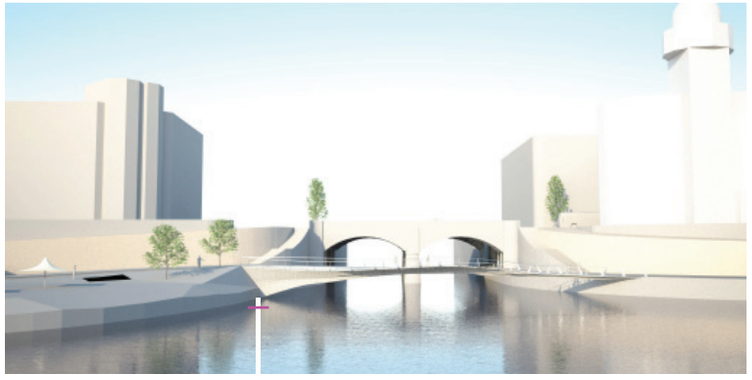
Das gesamte Raumprofil zwischen den Uferzonen wird freigehalten.

Durch einen mit einer begehbaren Glasabdeckung geschützten Schacht kann die Konstruktion eingesehen werden. Dies verstärkt die Erlebbarkeit des Bewegungsvorganges.

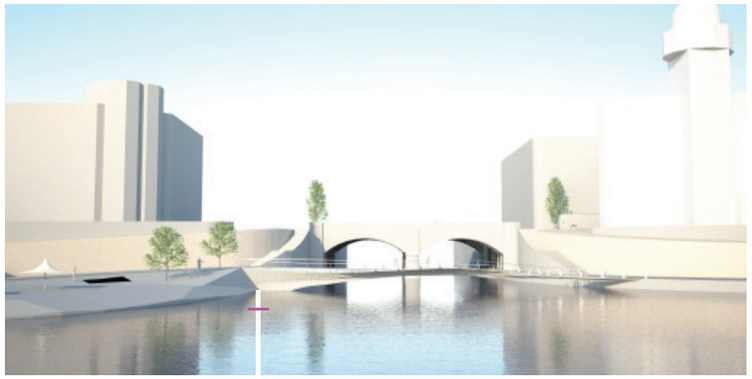
Bei 90 Grad aufgestellter Position wird die Brücke als temporäre Skulptur erkennbar, die gleichzeitig als Landmark wirksam wird.



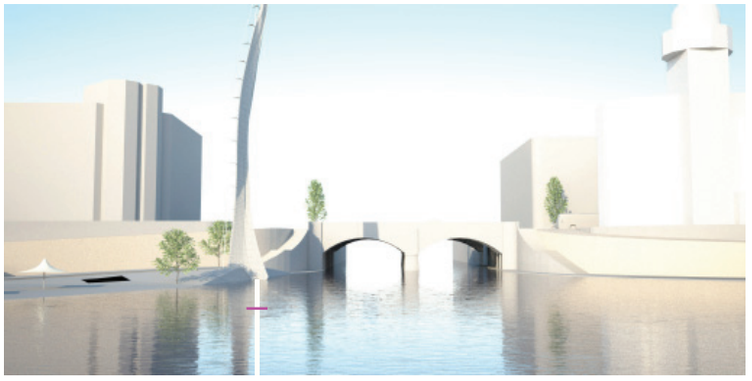
Regulierungswasserstand



Mittelwasser



höchster Schifffahrtswasserstand



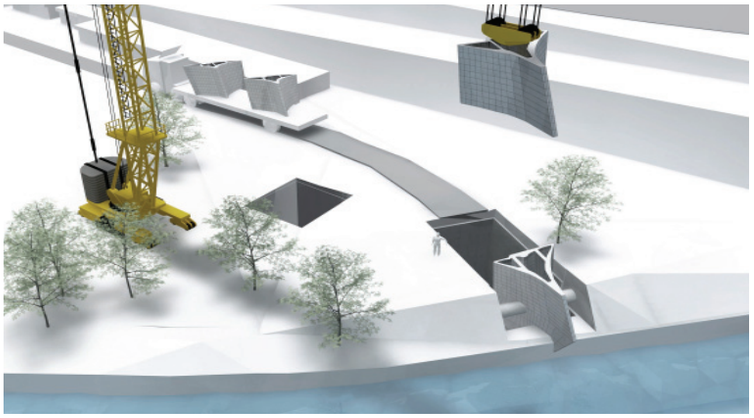
100-jähriges Hochwasser

Wasserstand

Die Uferzone auf der Herrmannparkseite wurde um 1.30 m angehoben, damit die Brücke im Regelfall nie im Wasser ist.

Brücke in horizontaler Position:
- Regulierungsniederwasser
- Mittelwasser
- Höchster Schifffahrtswasserstand

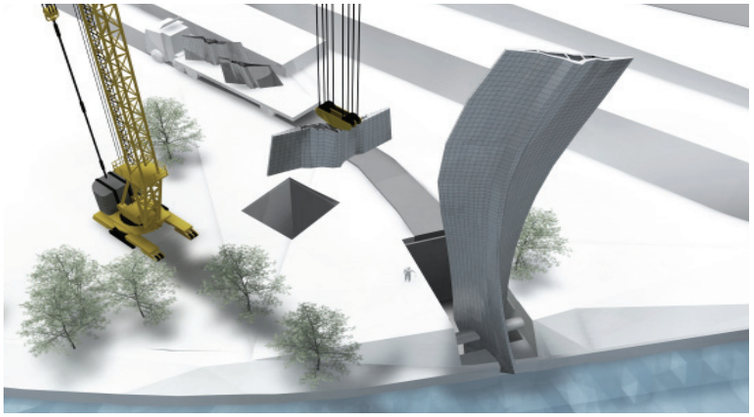
Brücke in aufgeklappter (vertikaler) Position:
- 100-jährliches Hochwasser
- Projekthochwasser



Bauablauf

Die Brücke besteht aus 20 einzelnen Segmenten, die alle in einer Fabrik vorgefertigt werden können. Jedes dieser Elemente verfügt über Schubrippen, womit ein Schubverbund zwischen den Segmenten hergestellt werden kann (siehe 1:50 Schnittdetails).

Grundsätzlich wird die Brücke in aufgestellter Position (vertikale Stellung) montiert. Dies hat den Vorteil, dass nicht über dem Gewässer gearbeitet werden muss und die Zukräfte in den Fugen der Konstruktion vermieden werden können.



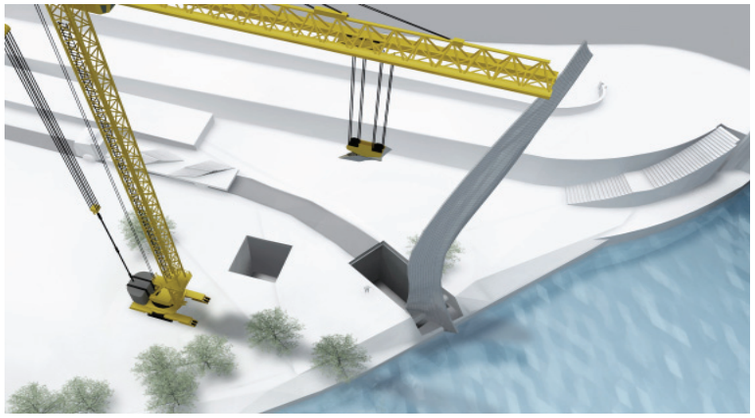
Die einzelnen Bauteile können mittels Baustelleinrichtungen relativ unkompliziert schrittweise übereinandergeschlichtet werden. Die Schubrippen helfen bei der korrekten Positionierung.

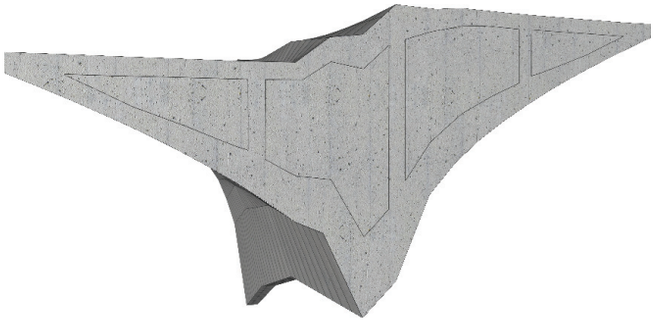
Die Spannglieder werden mit den Elementen gleichzeitig in die Segmenthohlräume eingezogen. In den einzelnen Hohlräumen sind Führungskonstruktionen für den Verlauf der Spannglieder vorgesehen.

Die Konstruktion wird während des Aufbauens vorgespannt und immer wieder nachjustiert.

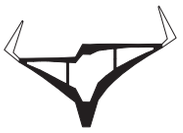
Nun kann die Brücke in die horizontale Position gebracht und die Oberfläche finalisiert werden.

Schlussendlich kann das Geländer auf den fertig eingelassenen Konsolen befestigt werden.





Schnitt 4



Schnitt 3

Schnitt 2

Schnitt 1

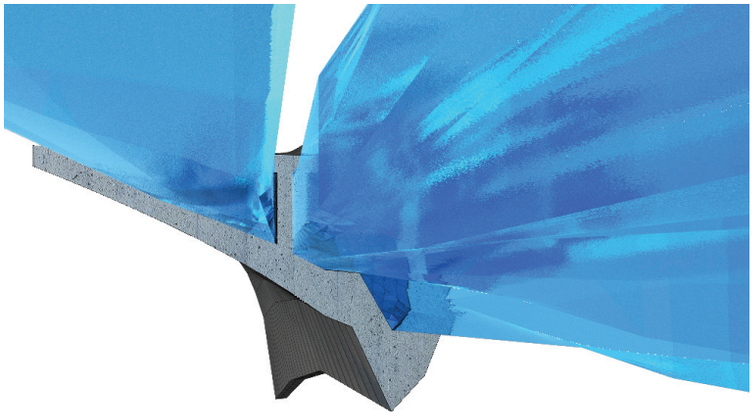
Schnitt 7

Schnitt 6

Schnitt 5

Schnitt 9

Schnitt 8



Schnitt 12

Schnitt 11

Schnitt 10

Schnitt 15

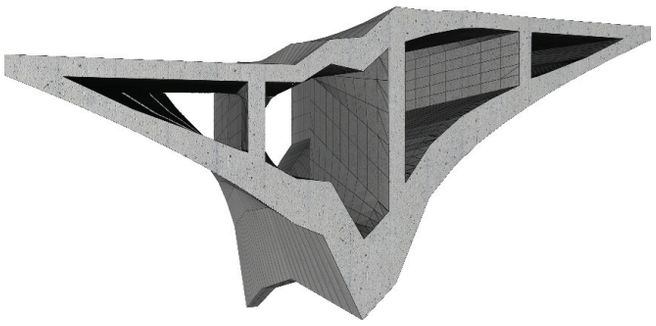
Schnitt 14

Schnitt 13

Schnitt 18

Schnitt 17

Schnitt 16



Schnitt 20

Schnitt 19

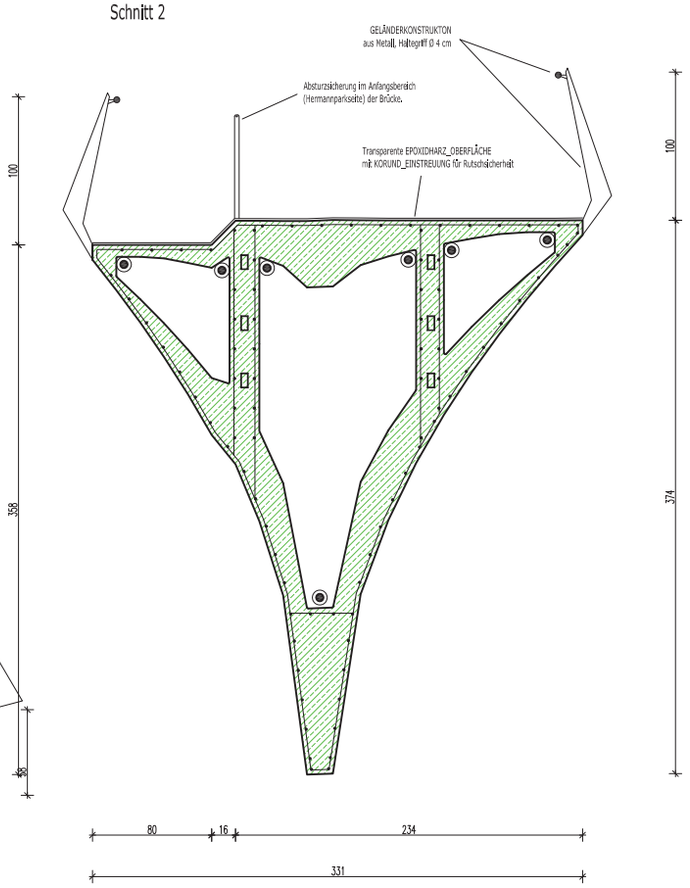
Brückenquerschnitte M 1:200

Die Systemquerschnitte werden automatisch über den Aushöhlungs-Algorithmus generiert und können relativ einfach in einem CAD-Programm weiterverarbeitet werden. Es sei jedoch darauf hingewiesen, dass diese Schnitte lediglich Vorschläge sind und diese selbstverständlich noch weiterbearbeitet bzw. vereinfacht werden können.

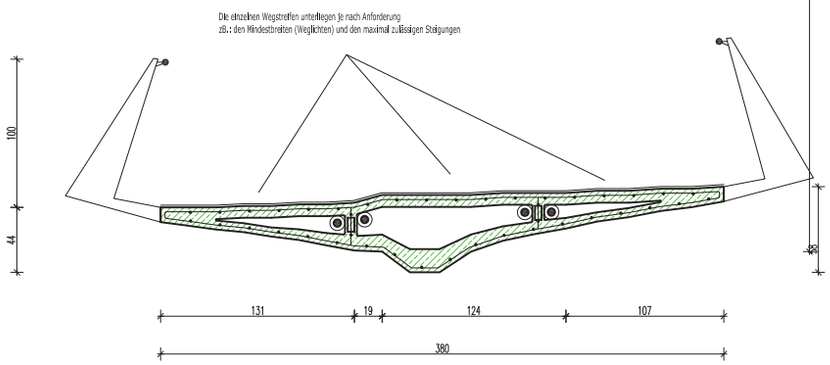
Detailschnitte M 1:50

In dieser Detailierung wurden die vom Script vorgeschlagenen Querschnitte nicht mehr idealisiert, sondern direkt weiterverarbeitet.

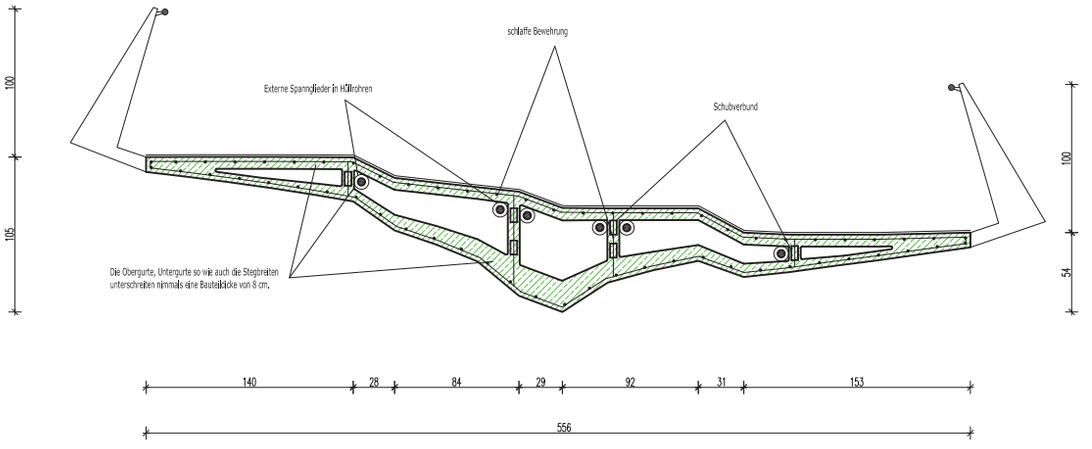
Schnitt 2

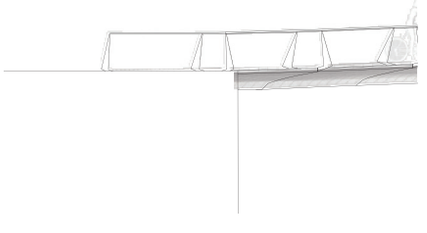


Schnitt 18

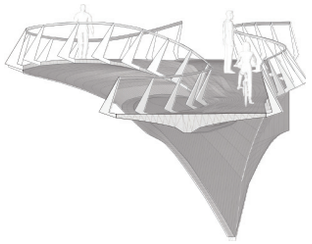


Schnitt 10

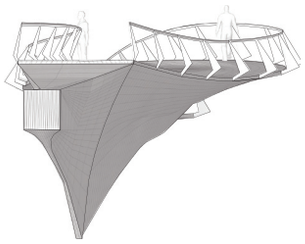




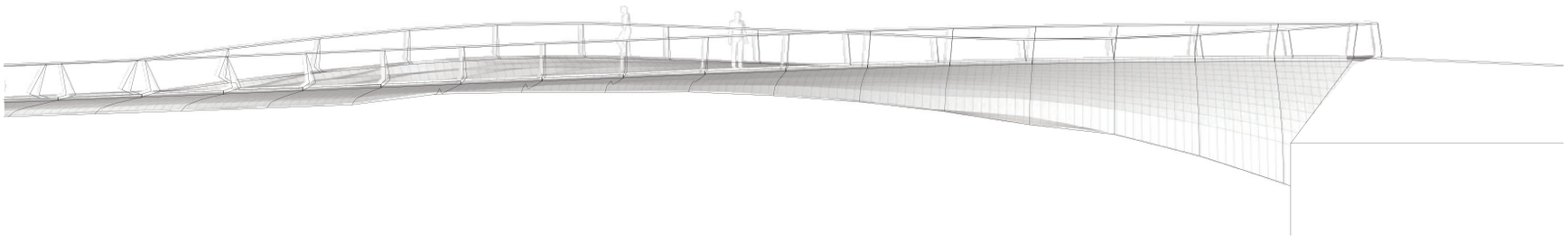
OST_ANSICHT
M 1:200



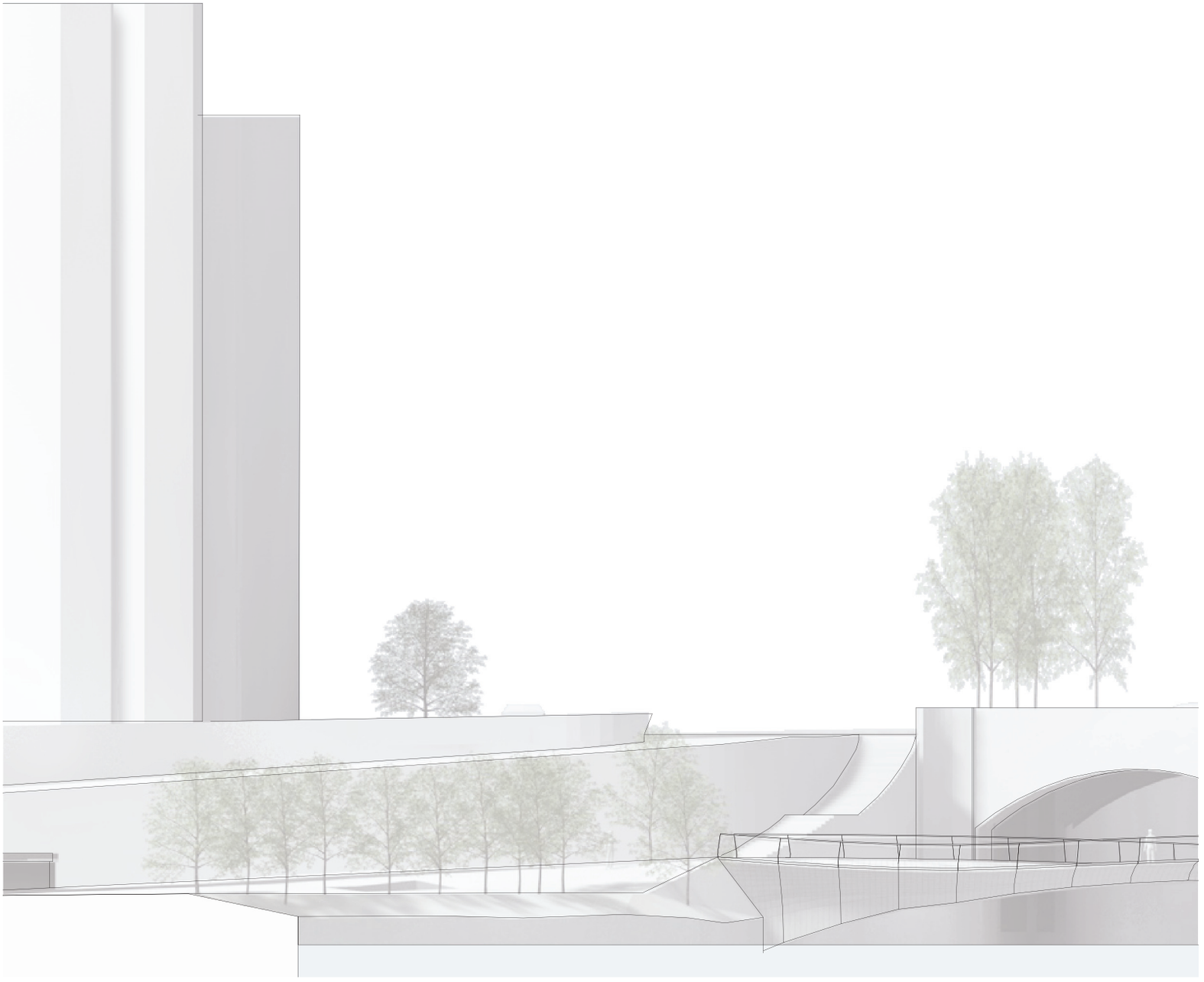
WEST_ANSICHT
M 1:200

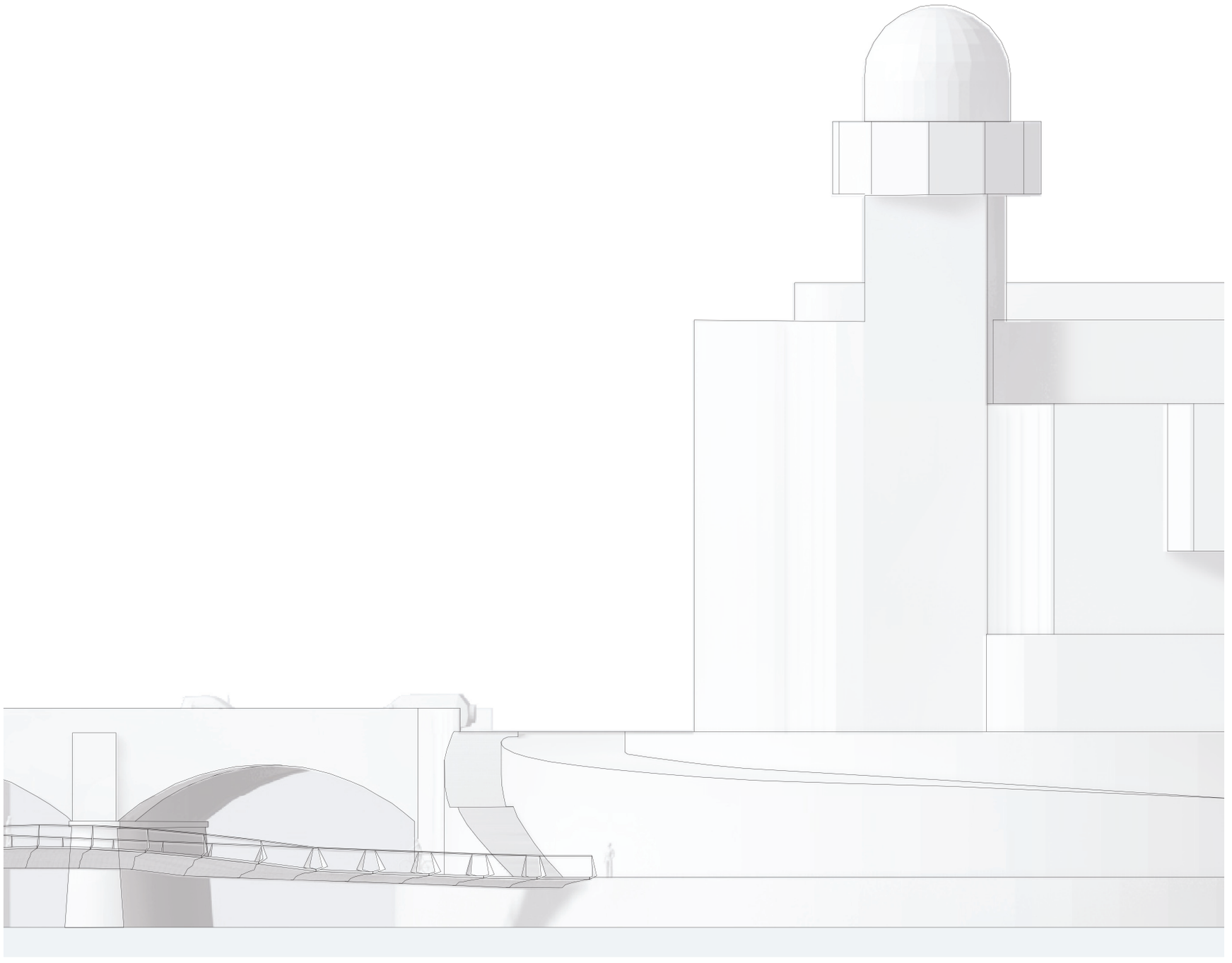


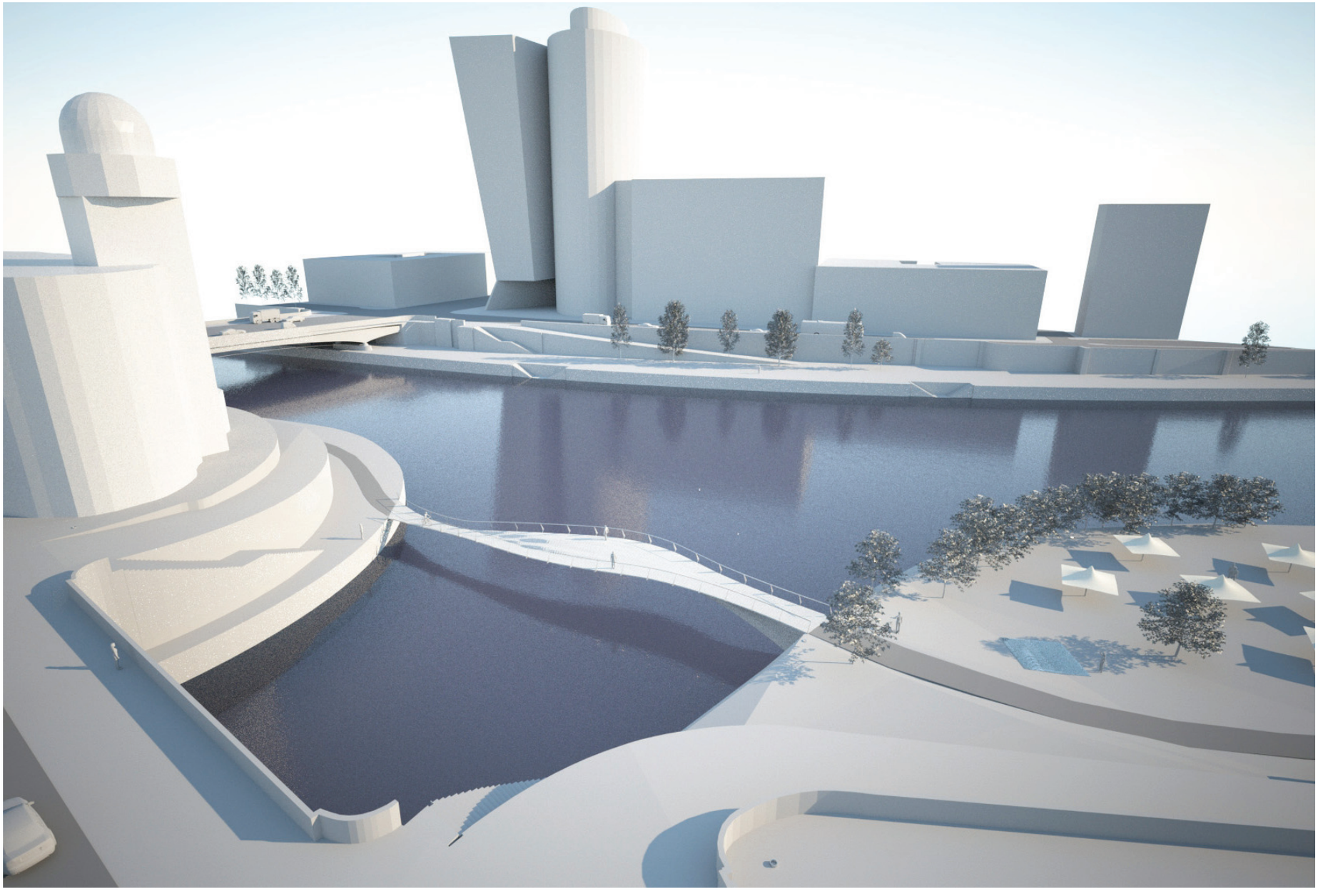
SÜD_ANSICHT
M 1:200

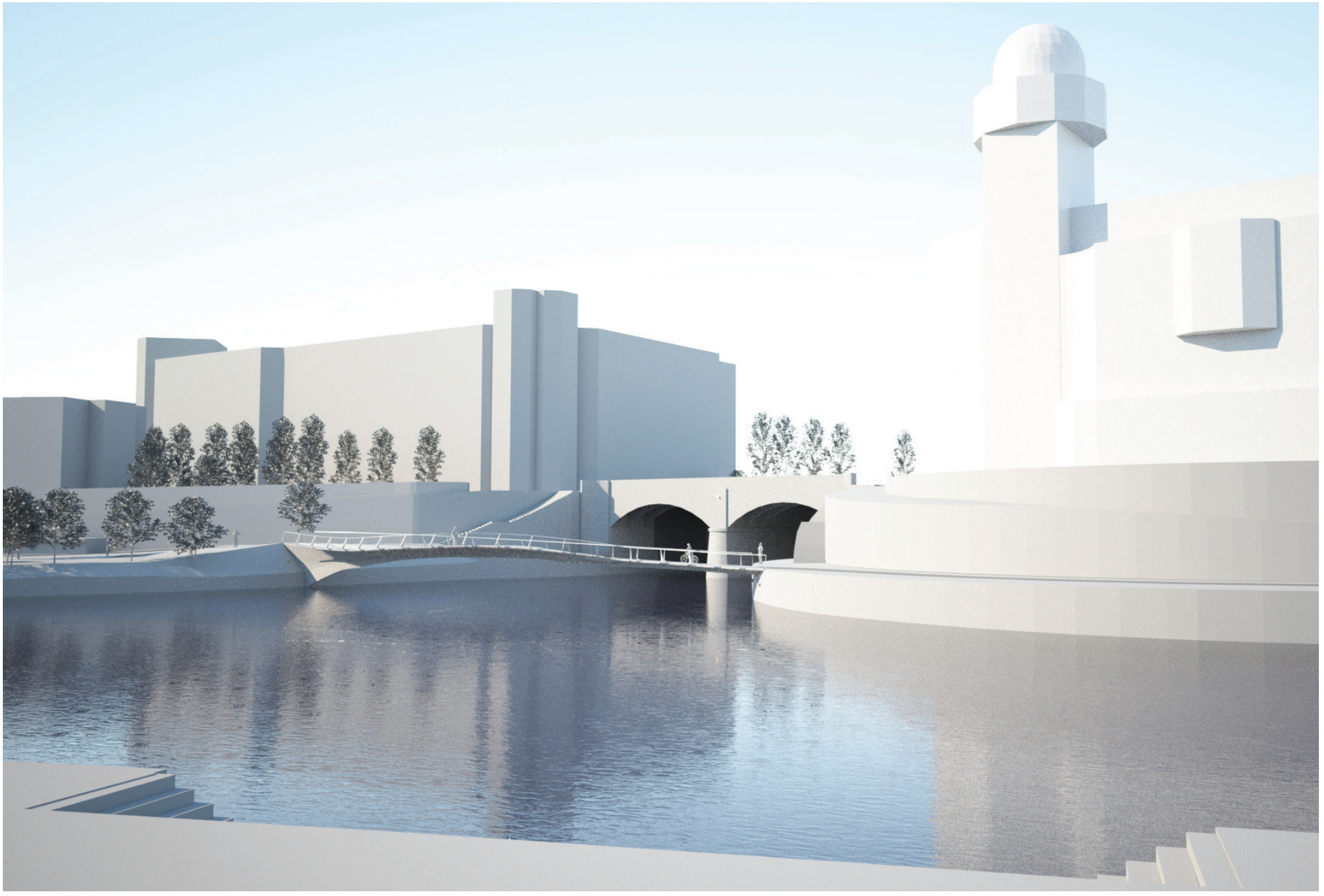


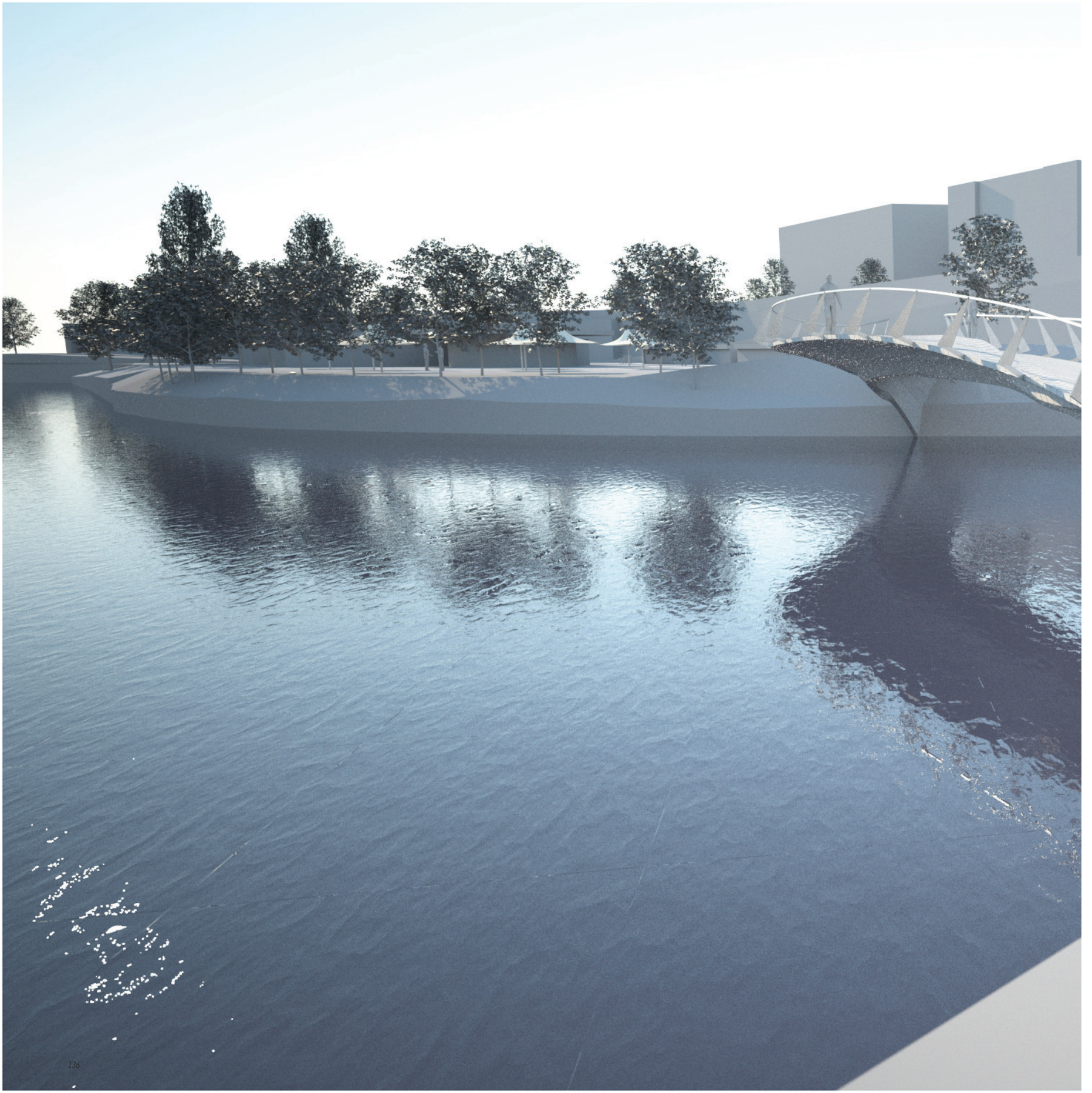
NORD_ANSICHT
1:200

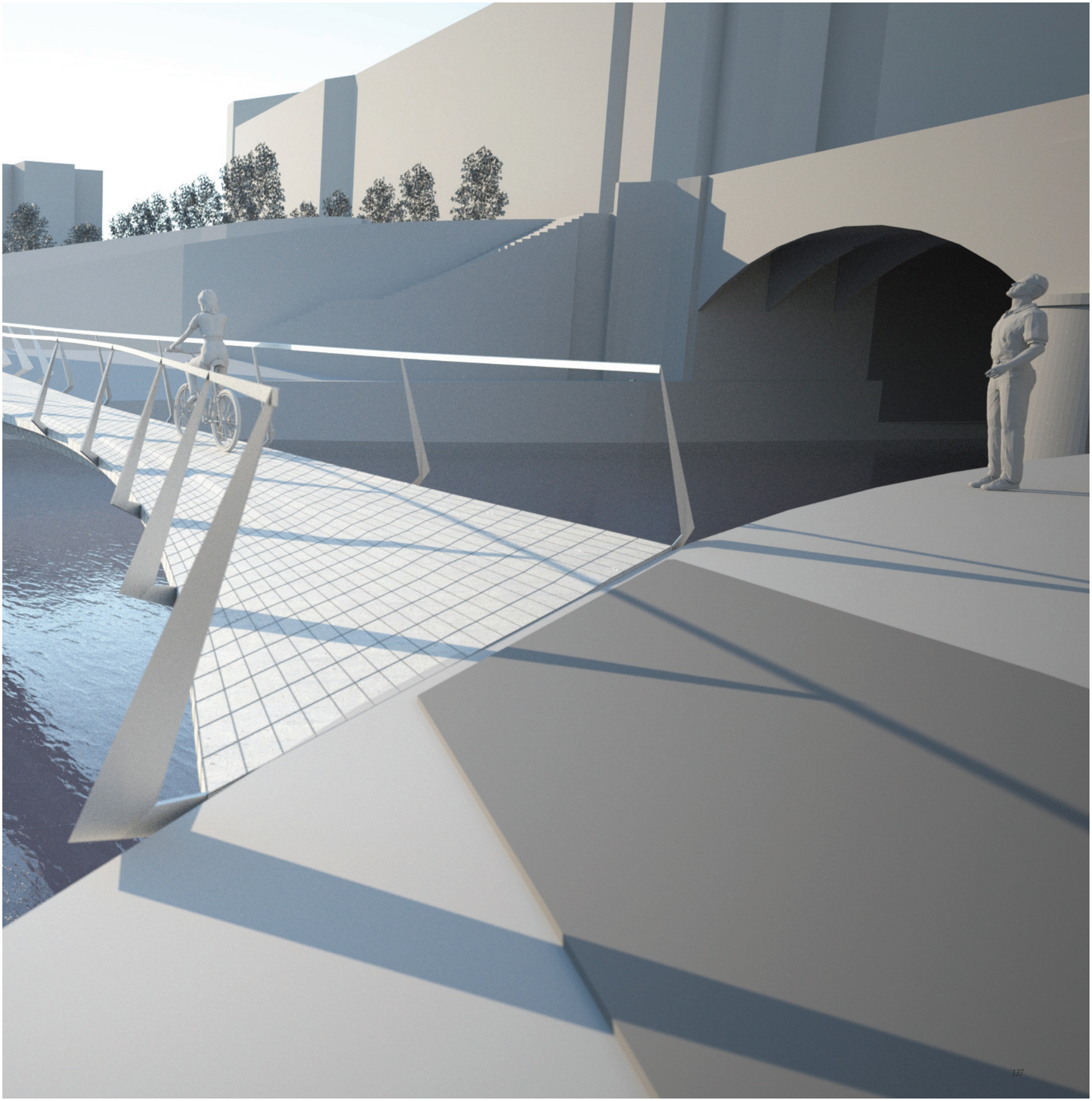


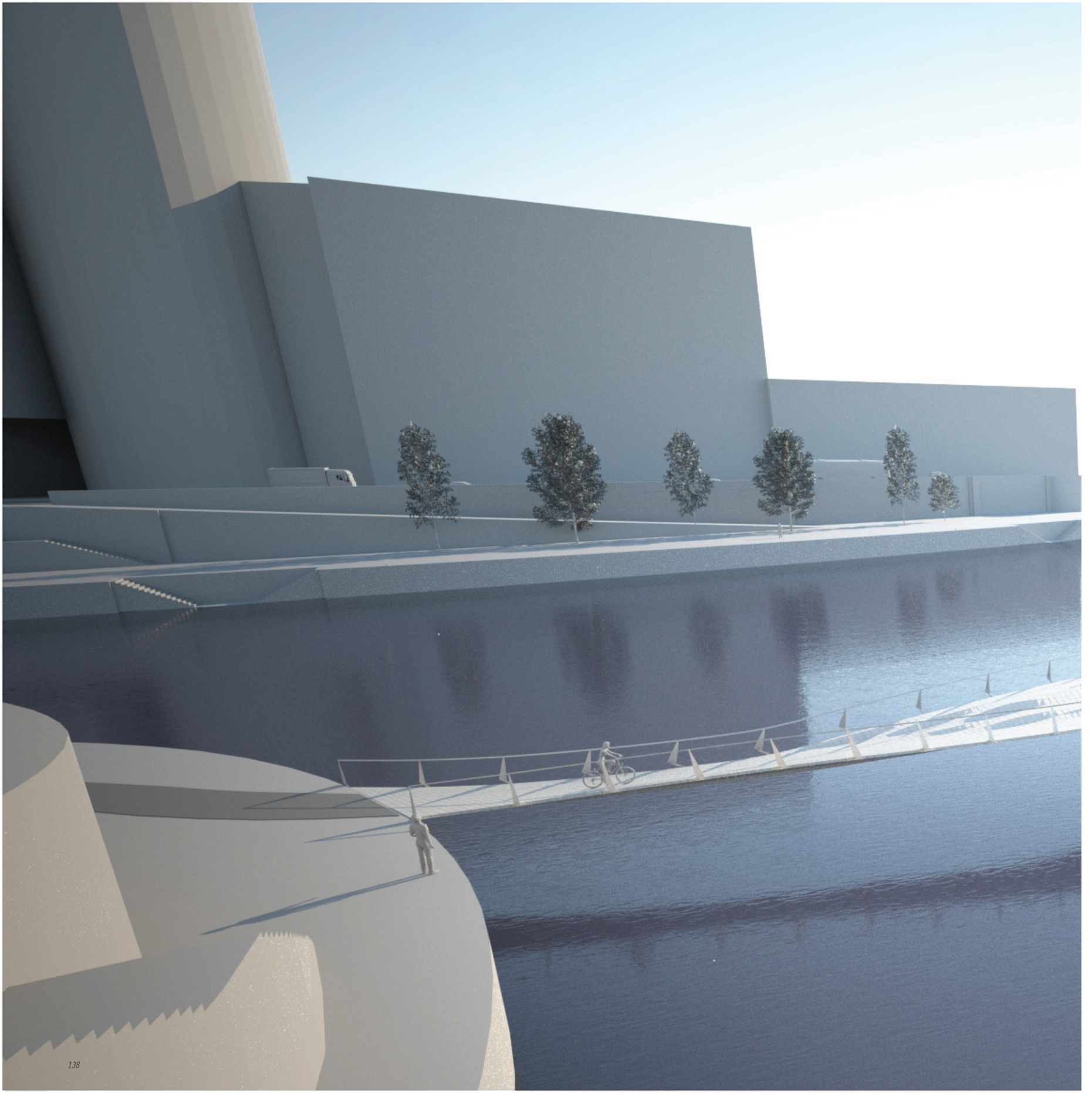


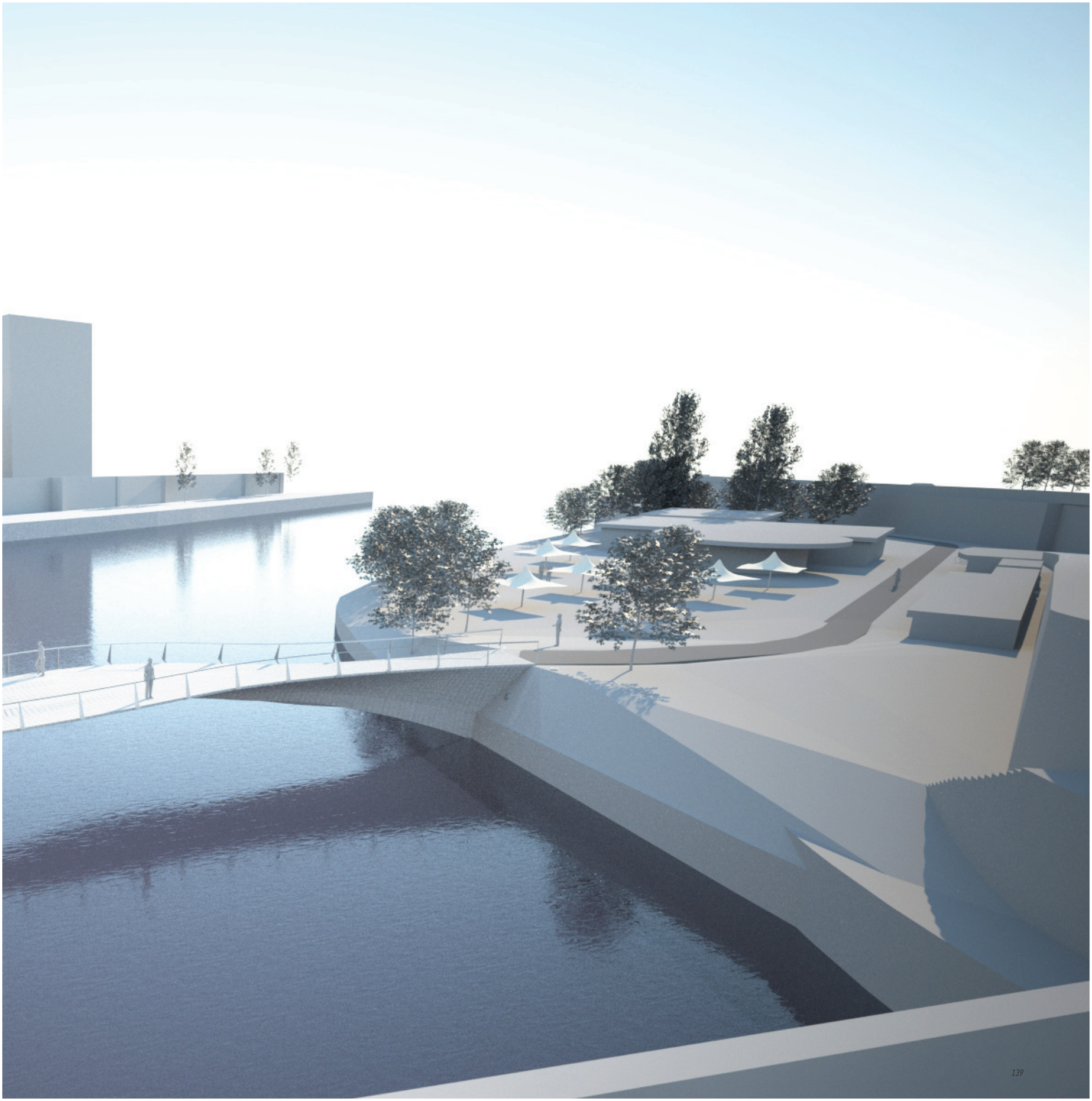


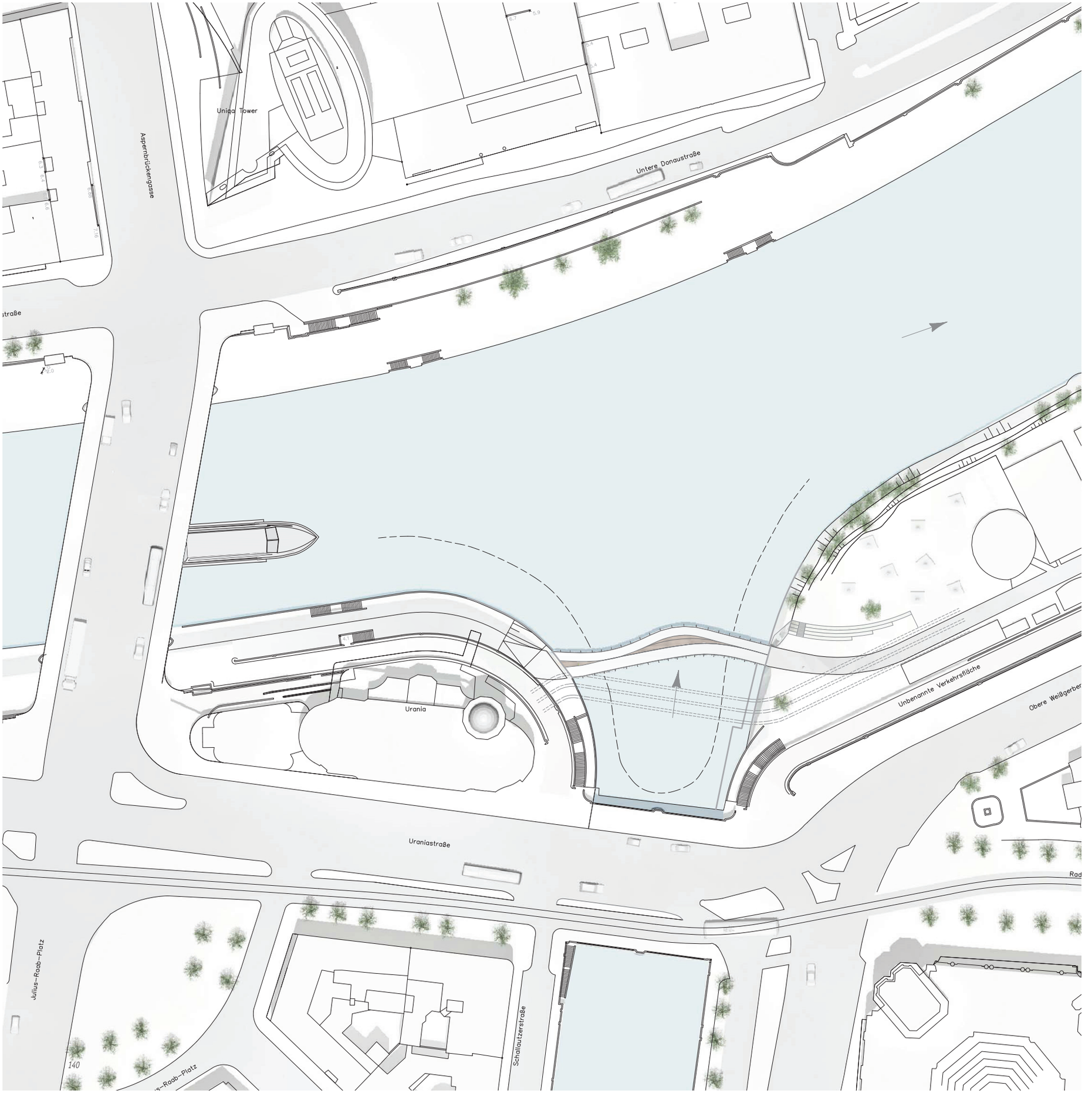


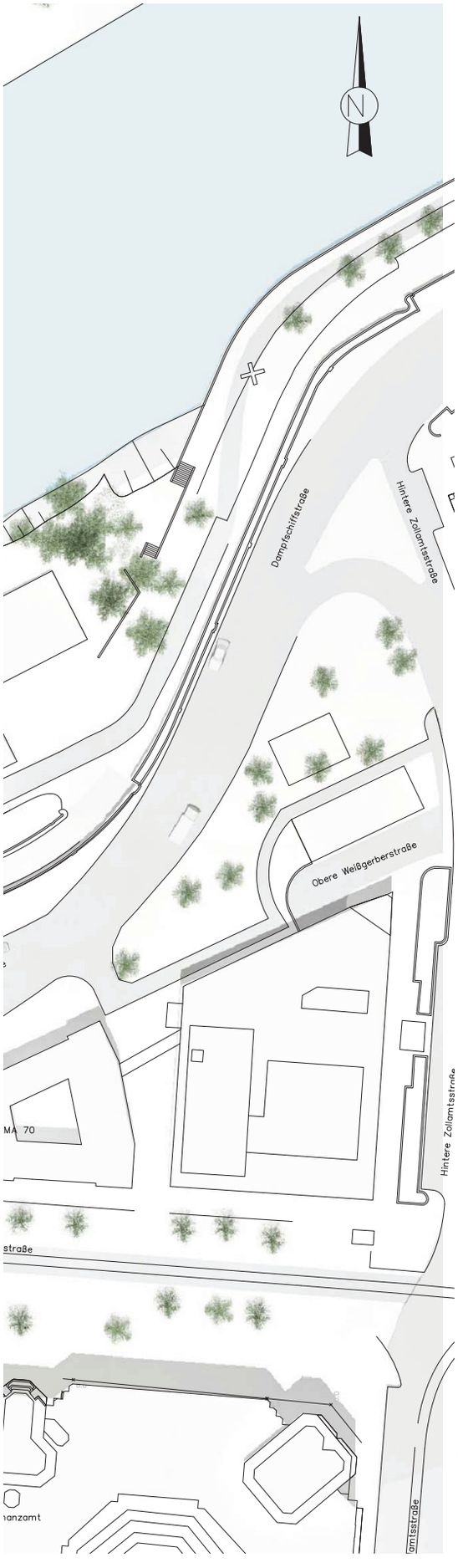









Uniqa Tower
Aspernbrückengasse
Untere Donaustraße
Urania
Unbenannte Verkehrsfläche
Uraniastraße
Julius-Raab-Platz
Schallautzerstraße



LAGEPLAN
M 1:1000

Connecting Link

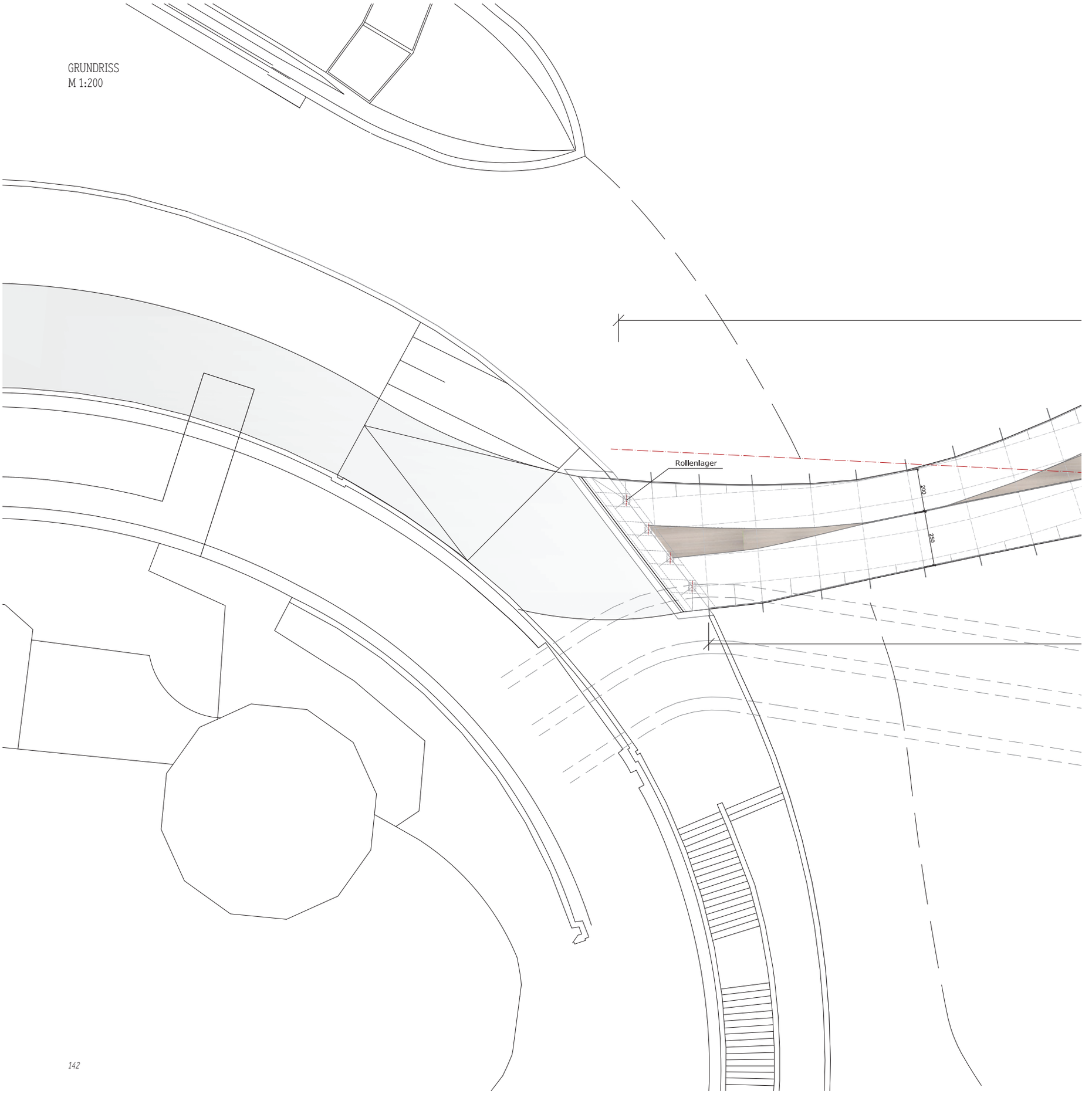
Konzepterweiterung

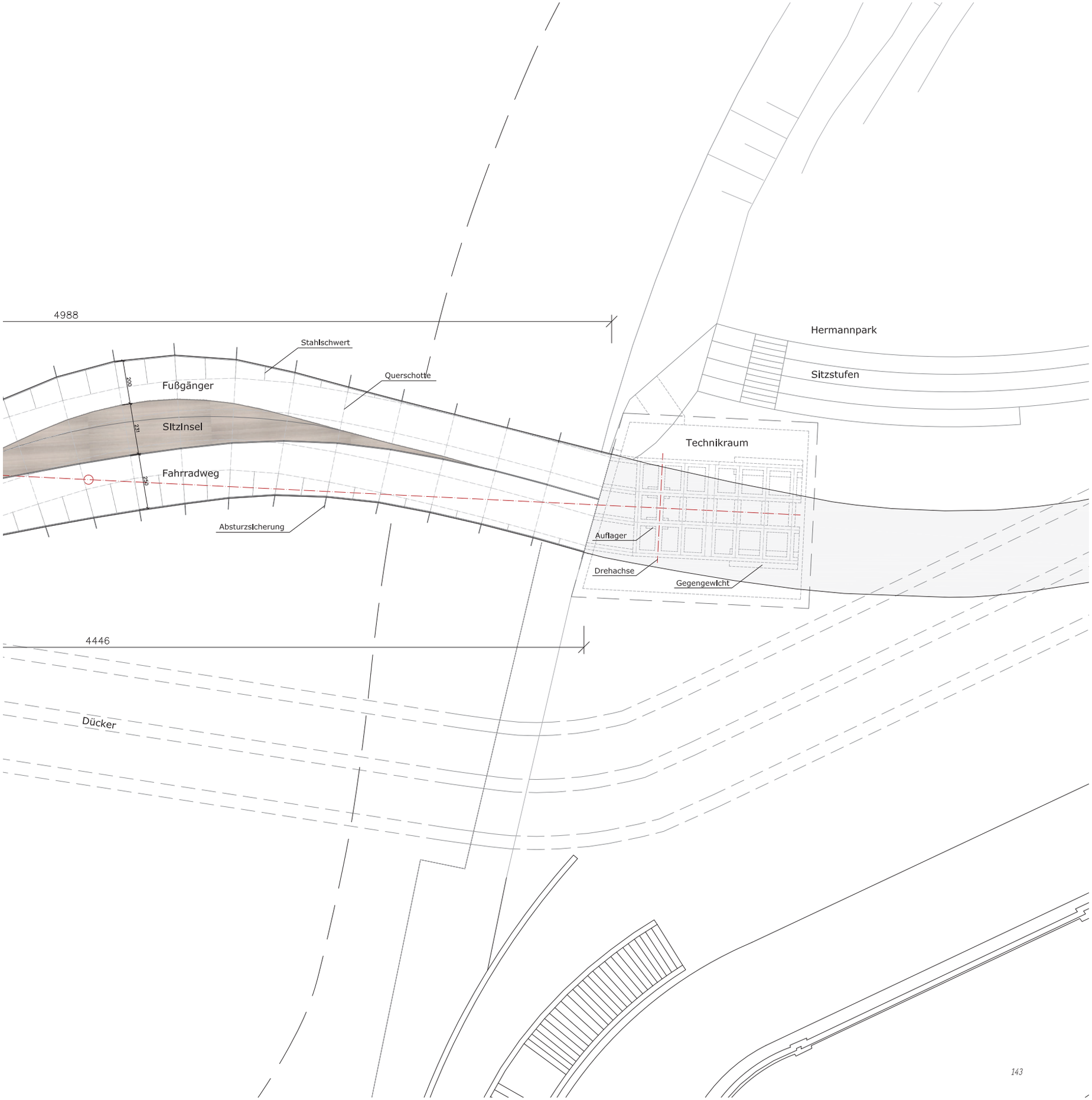
Durch Schaffung eines Kommunikationsraumes im Hermannpark in Form einer Freilufttribüne und einer attraktiven Parkgestaltung findet eine Aufwertung des bestehenden Areals statt. Der vom Radweg bestimmte Durchzugsraum wird durch Ruhezonen vor und auf der Brücke entschleunigt. Die Sitzinsel auf der Brücke ermöglicht ein Verweilen in einer speziellen städtischen Situation über einer bewegten Wasserfläche und lässt durch ihre Großzügigkeit unterschiedliche Nutzungsformen zu.

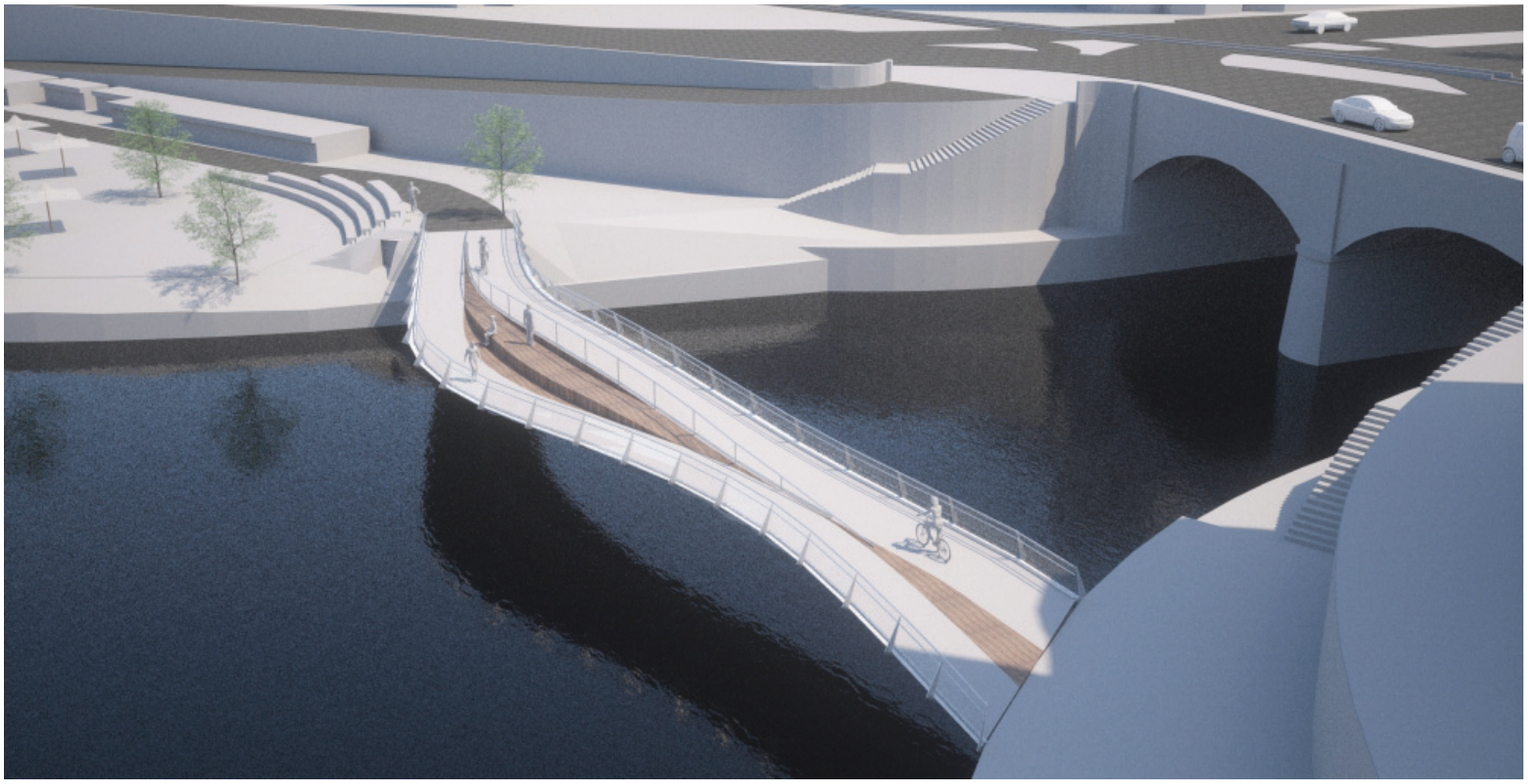
Formgebung

Hermannparkseitig nimmt die Brücke die Fließrichtung des Wienflusses auf und dreht sich im Grundriss in Richtung Donaukanal. Im restlichen Brückenverlauf werden die Uferkanten der Flussmündung durch einen Gegenbogen fortgeführt. Die Ansicht wird vor allem durch das Tragwerk und das Höhenspiel des Radfahrweges bestimmt. Dieser steigt zur Brückenmitte hin bis auf eine Höhendifferenz von 70cm gegenüber dem Fußgängerbereich an, welcher barrierefrei ausgeführt ist. Das Tragwerk (Raumfachwerk) folgt dem Höhenverlauf des Brückendecks, ist jedoch bei Tag nicht sichtbar, weil es durch eine weiße, transluzente Membran umhüllt wird. Diese ist einerseits durch das Erzeugen der Kontur, andererseits zur horizontalen Gliederung (Segmente) für das äußere Erscheinungsbild der Brücke unverzichtbar. Durch eine interne Beleuchtung wird die Tragstruktur an die Innenseite der Membran projiziert und so bei Nacht erlebbar.

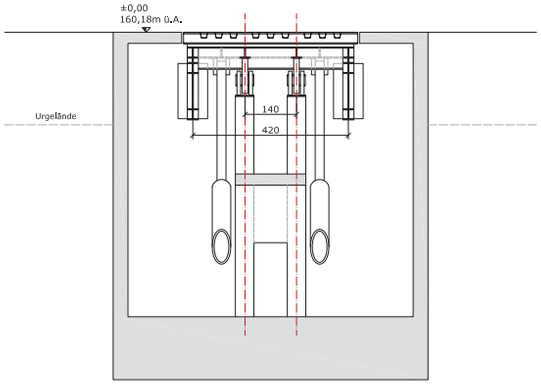
GRUNDRISS
M 1:200



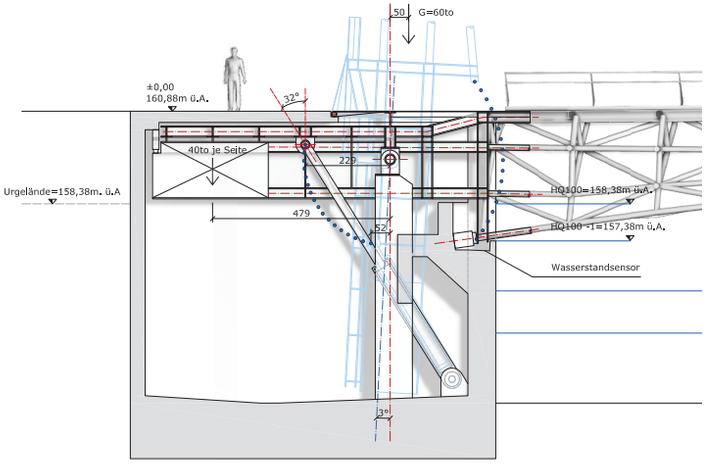

4988
Stahlschwert
Querschotte
Fußgänger
SItzInsel
Fahrradweg
Absturzsicherung
Hermannpark
Sitzstufen
Technikraum
Auflager
Drehachse
Gegengewicht
4446
Dücker

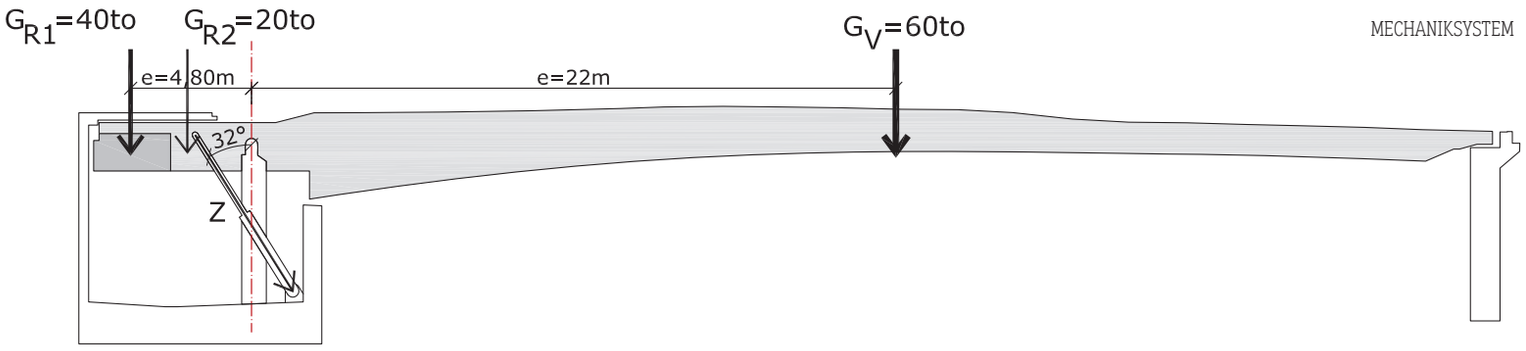


QUERSCHNITT DURCH AUFLAGER
M 1:200



LÄNGSSCHNITT
M 1:200



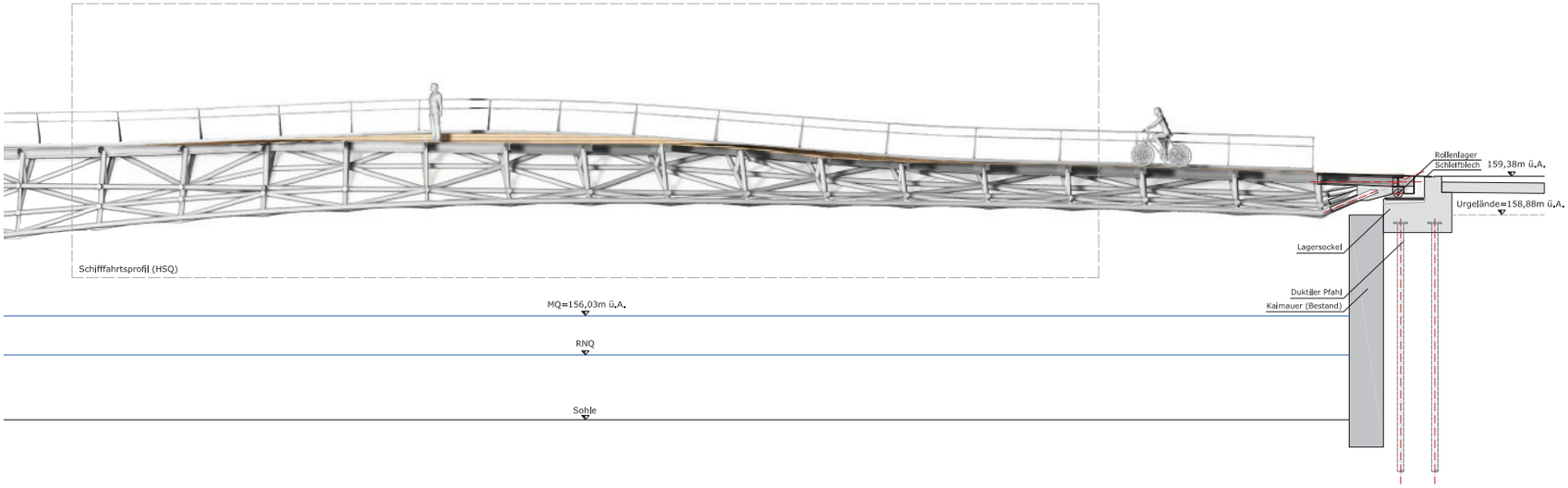


KONSTRUKTIONSDETAIL

Mechanikkonzept

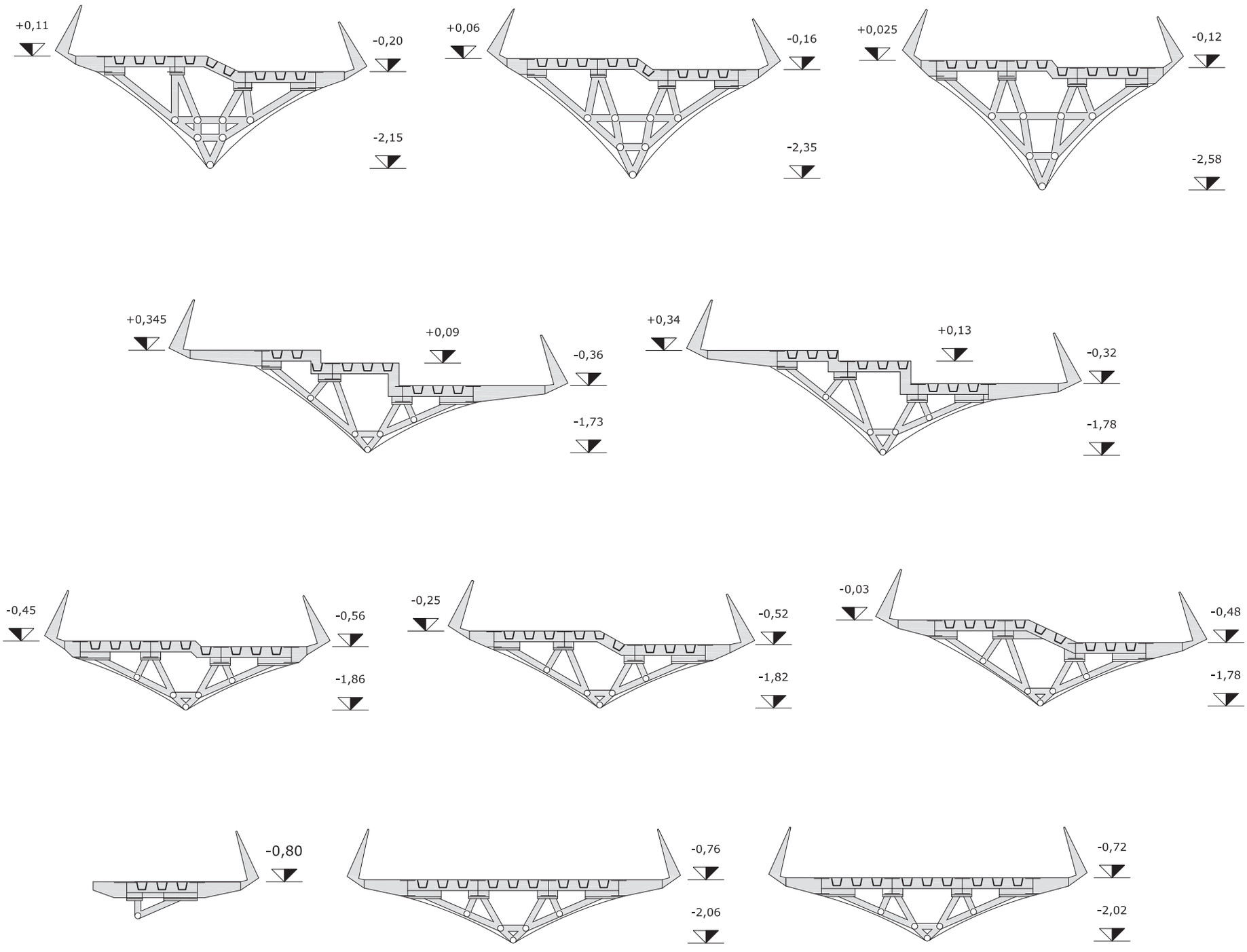
Als Konzept wird ein Klappmechanismus mit Hilfe eines Druckmittelgetriebes gewählt. Der Mechanismus besteht aus einer Drehachse auf der Seite des Hermannparks, die es ermöglicht, die Brücke über zwei symmetrisch zur vertikalen Drehachse angeordneten Hydraulikantriebe (=Kipphydraulik) zu kippen. Bei einem gewählten Kolbendurchmesser von ca. 400mm und einem Verhältnis der Ausladungs- zur Gegengewichtsstrecke von 8:1 sind Drücke von max. 20 MPa erforderlich, um auf der Seite des Gegengewichtes das notwendige Drehmoment zu erzeugen. Zur Verringerung des Kippmoments sind Gegengewichte von 40to je Hauptträger vorgesehen. Nach erfolgtem Klappvorgang wird die Brücke durch zwei weitere Getriebe in ihrer vertikalen Lage fixiert (=Fixierhydraulik).

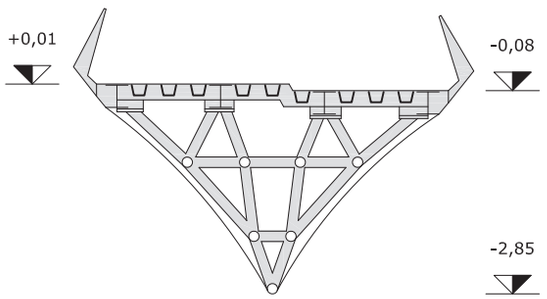
Zwei Hydraulikzylinder mit einem Kolbendurchmesser von ca. 200mm mit Drücken bis ca. 8 MPa im Bereich der Gegengewichte ermöglichen ebenso wie ein zentrisch zur Brückenachse angeordneter Zylinder mit Kolbendurchmesser von ca. 400mm und Drücken bis ca. 20 MPa im Bereich der Kaimauer das Fixieren des Tragwerks für den Arretierungszustand. Die Steuerung sowohl der Kipp- als auch der Fixierhydraulik erfolgt für die unterschiedlichen Kreisläufe zentral von einem Aggregat.

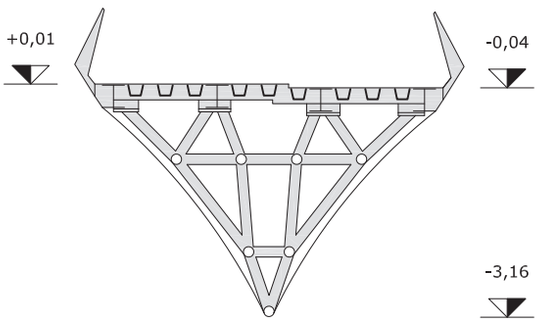


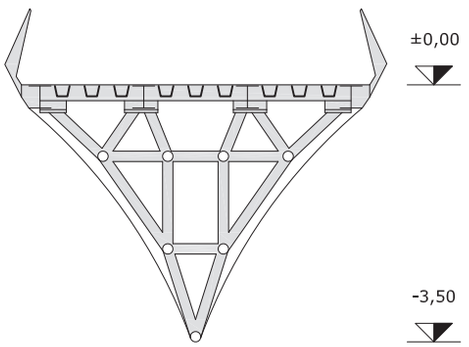
Brückenquerschnitte 1:200

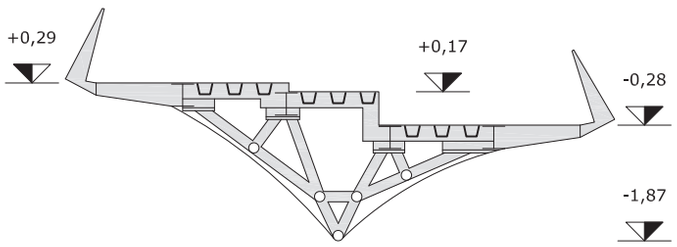
Die automatisch generierten Systemquerschnitte mussten aufgrund des Konstruktionswechsels von einer Stahlplatten- zu einer Fachwerkskonstruktion umgezeichnet werden. Dieser Systemwechsel wurde im Ingenieurbüro beschlossen, da die Fachwerkskonstruktion eine wesentliche Gewichtsersparnis gegenüber der Stahlplattenkonstruktion bedeutet. Außerdem ist eine Realisierung einfacher, da ein Fachwerk über einen dreidimensionalen Stabplan und einzelne Profilzuweisungen mittels Roboter zu einem hohen Grad automatisiert hergestellt werden kann.

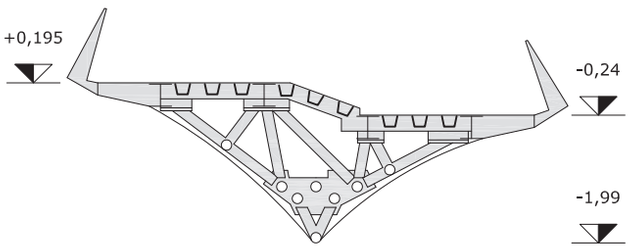


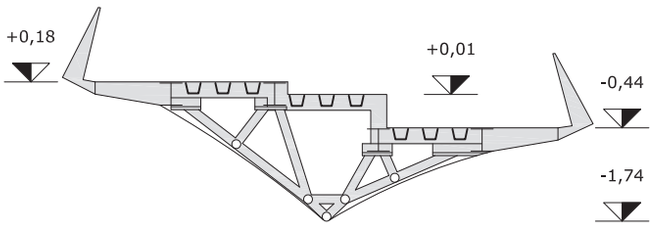

+0,01
-0,08
-2,85

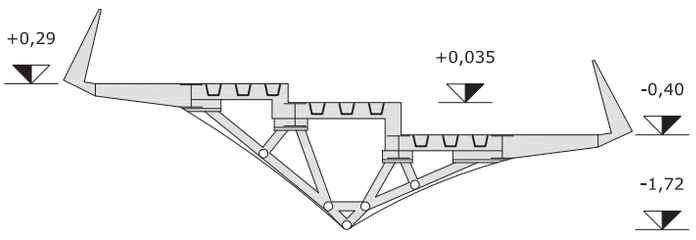

+0,01
-0,04
-3,16

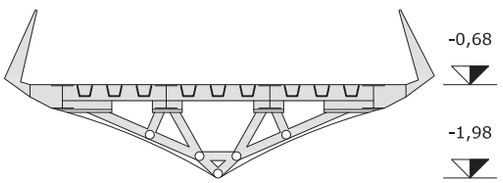

±0,00
-3,50

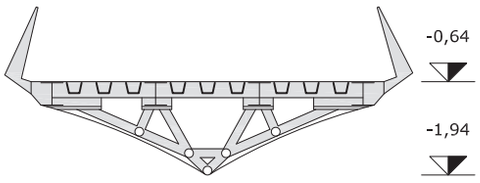

+0,29
+0,17
-0,28
-1,87

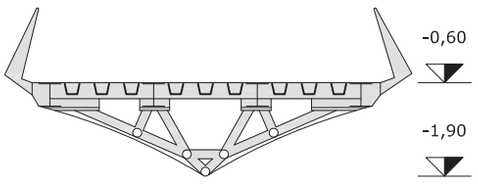

+0,195
-0,24
-1,99

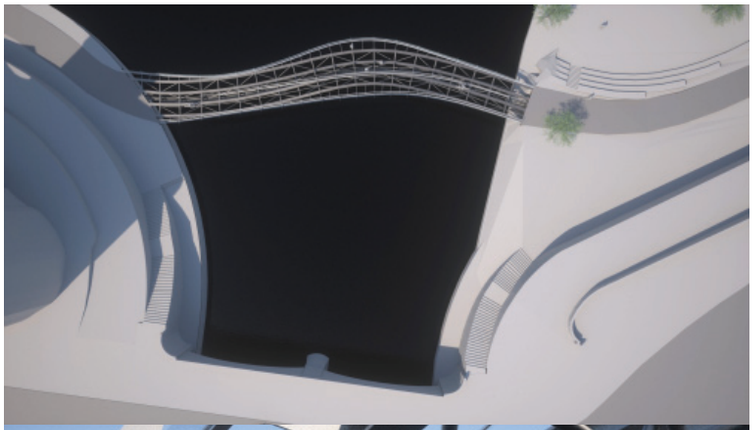

+0,18
+0,01
-0,44
-1,74

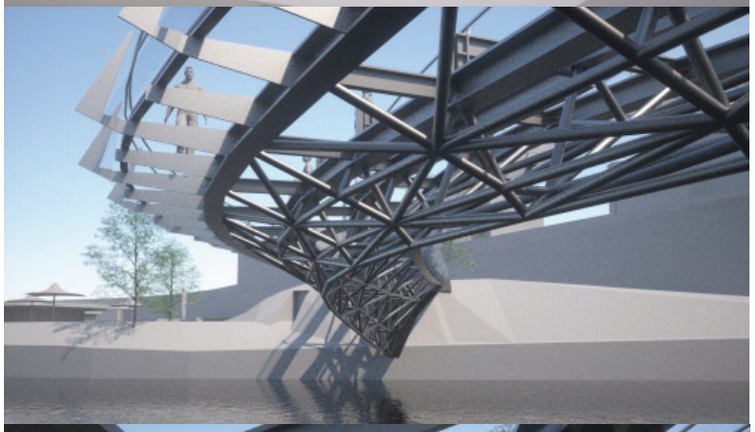

+0,29
+0,035
-0,40
-1,72

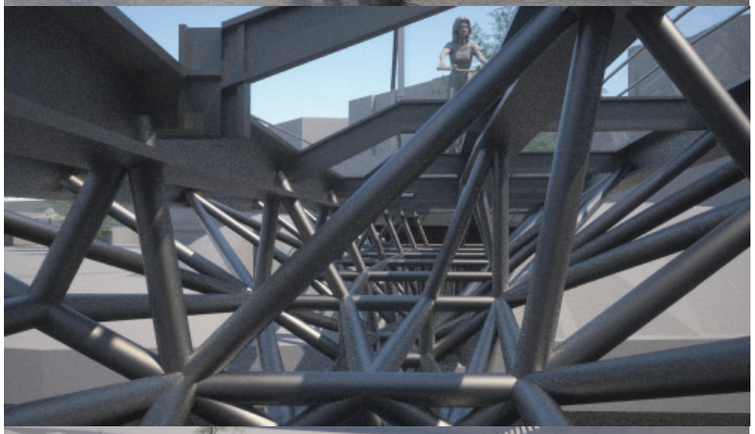

-0,68
-1,98

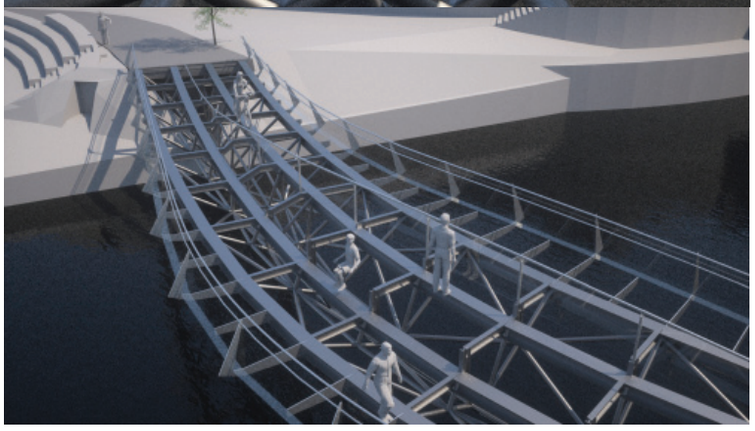

-0,64
-1,94


-0,60
-1,90









Montagekonzept

Das Tragwerk wird in zwei Teilen im vertikalen Zustand montiert. In einem ersten Schritt wird der Blechquerschnitt bis zur Drehachse im Hermannpark vormontiert, vertikal eingehoben und anschließend werden die hydraulischen Antriebe montiert. Im zweiten Schritt wird die Fachwerkskonstruktion ebenfalls auf der Baustelle vormontiert und anschließend als Gesamtkonstruktion mit einer Länge von 48m vertikal eingehoben und im Übergangsbereich mit dem Blechquerschnitt verschraubt.

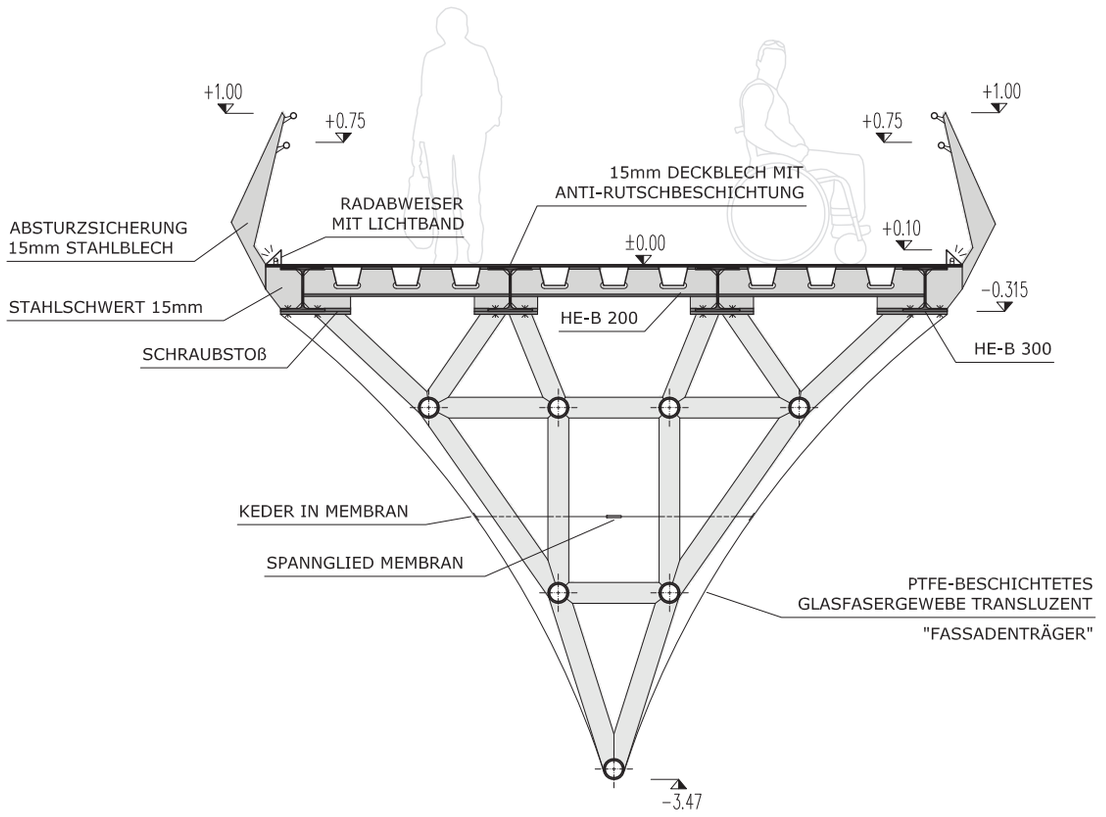
Sicherheitskonzept

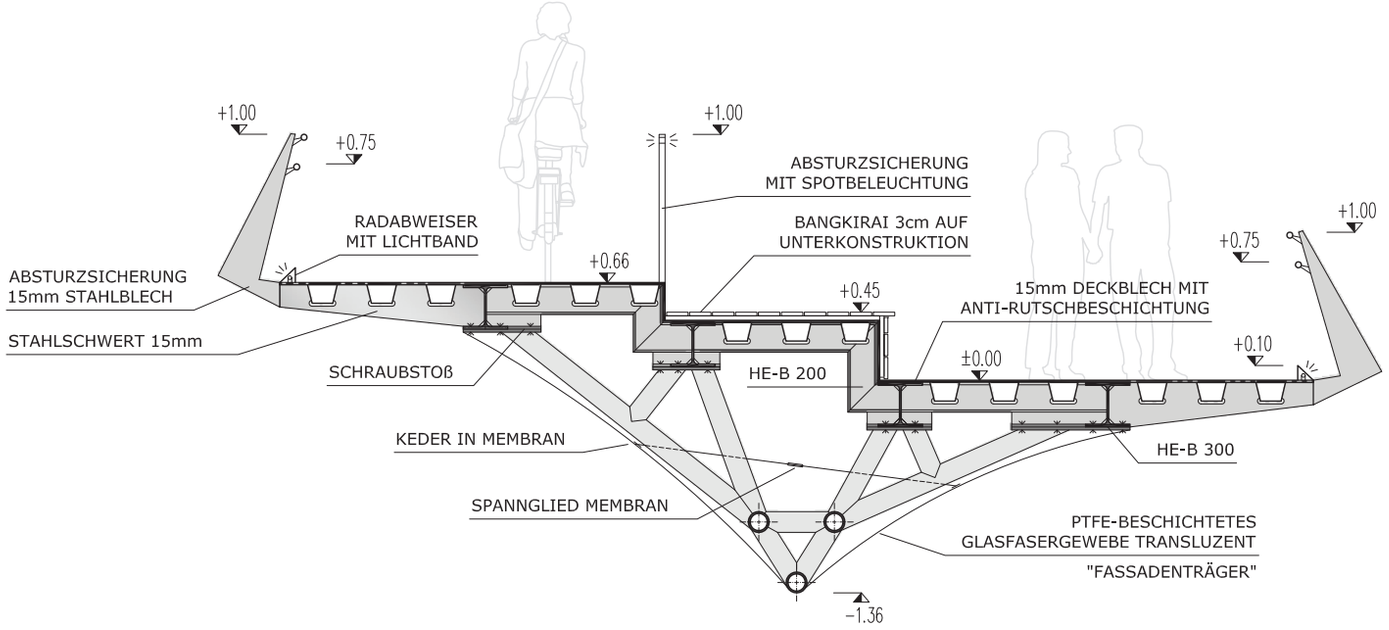
Der Klappvorgang wird durch eine Ampelregelung (optisch) an beiden Tragwerksseiten sowie einer Sirene (akustisch) angekündigt. Ein installiertes Scansystem erkennt auf der Brücke verbleibende Passanten und kann den Kippvorgang stoppen. Zusätzlich zum Scansystem ist eine Videoüberwachung installiert, die es ermöglicht, den Kippvorgang auf Bildschirmen zu verfolgen bzw. bei Notwendigkeit Räumungseinsätze anzuordnen. Uraniaseitig sind automatisierte und klappbare Barrieren sowie Absturzsicherungen vorgesehen. Sicherheitspersonal vor Ort ist nicht notwendig.

Beleuchtungskonzept

Zwei Lichtbänder im Bereich des Brückendecks sowie einzelne Lichtspots im fahrbahntrennenden Geländerbereich sorgen für eine adäquate Orientierung bei schlechten Sichtbedingungen. Die in den äußeren Radabweisern integrierten Lichtbänder betonen die dynamische Form der Brücke. In aufgeklapptem Zustand abstrahieren die Lichtstreifen die Brücke zu einem unverkennbaren Wahrzeichen. Durch das Beleuchten der Tragkonstruktion im Inneren der Brücke wird die Membranoberfläche ornamentiert, wodurch in moderner Form auf den historischen „Fassadenträger" hingewiesen wird.

DETAILLSCHNITTE
M 1:50





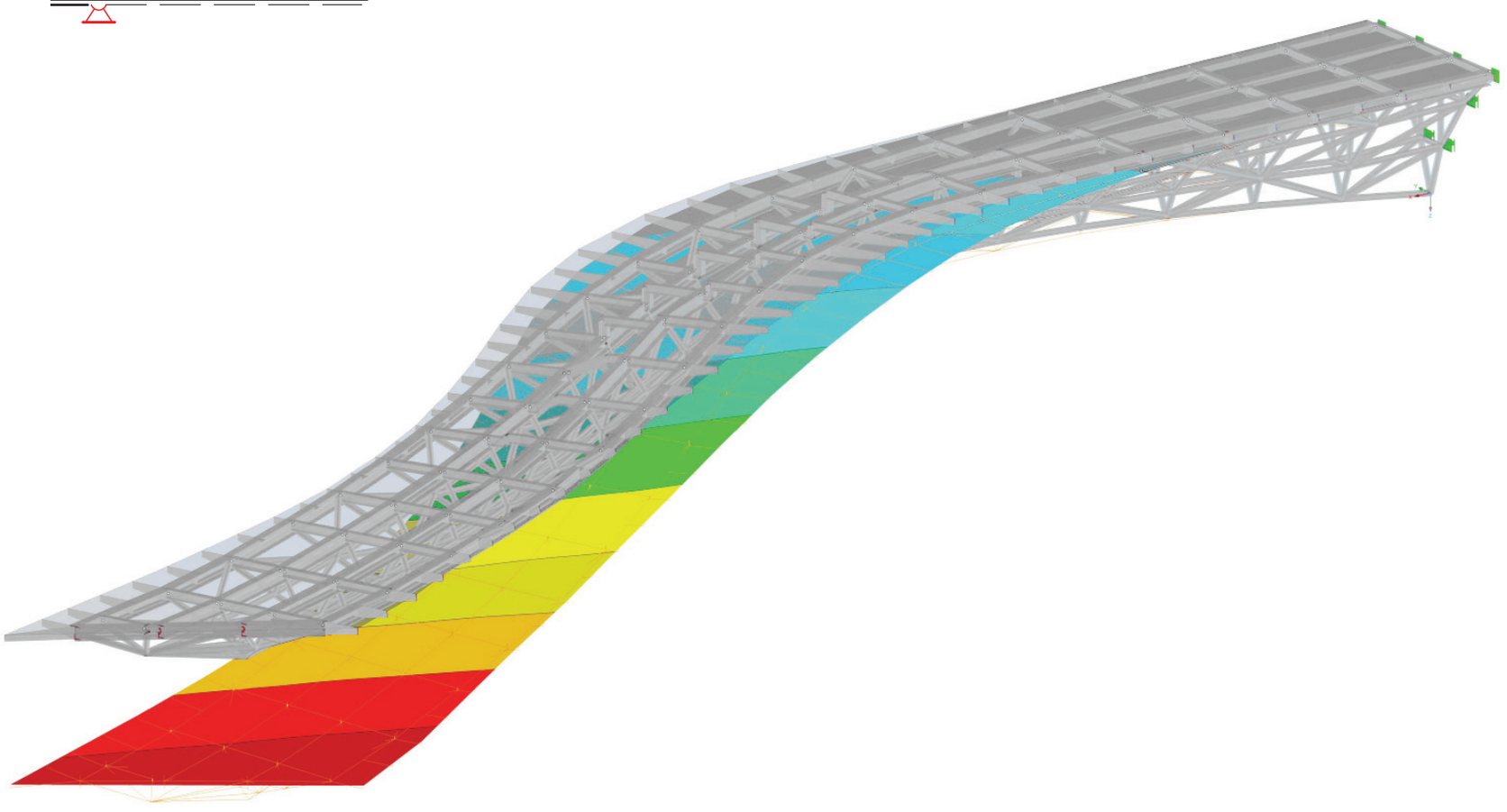
Verformungsverhalten

Die Brücke ist im Kippzustand als Waagebalken mit Gegengewicht wirksam. Das Kragverhältnis beträgt 8:1, wobei das Differenzmoment mit Hilfe eines Druckmittelgetriebes aufgebracht wird. Im völlig gekipptem Zustand (87°) wirkt die Brücke als Skulptur, wobei hierfür eine zusätzliche mechanische Fixierung vorgesehen ist. Die seitlichen Windkräfte werden durch die Spreizung (e=1,40m) der beiden Drehachsen aufgenommen.

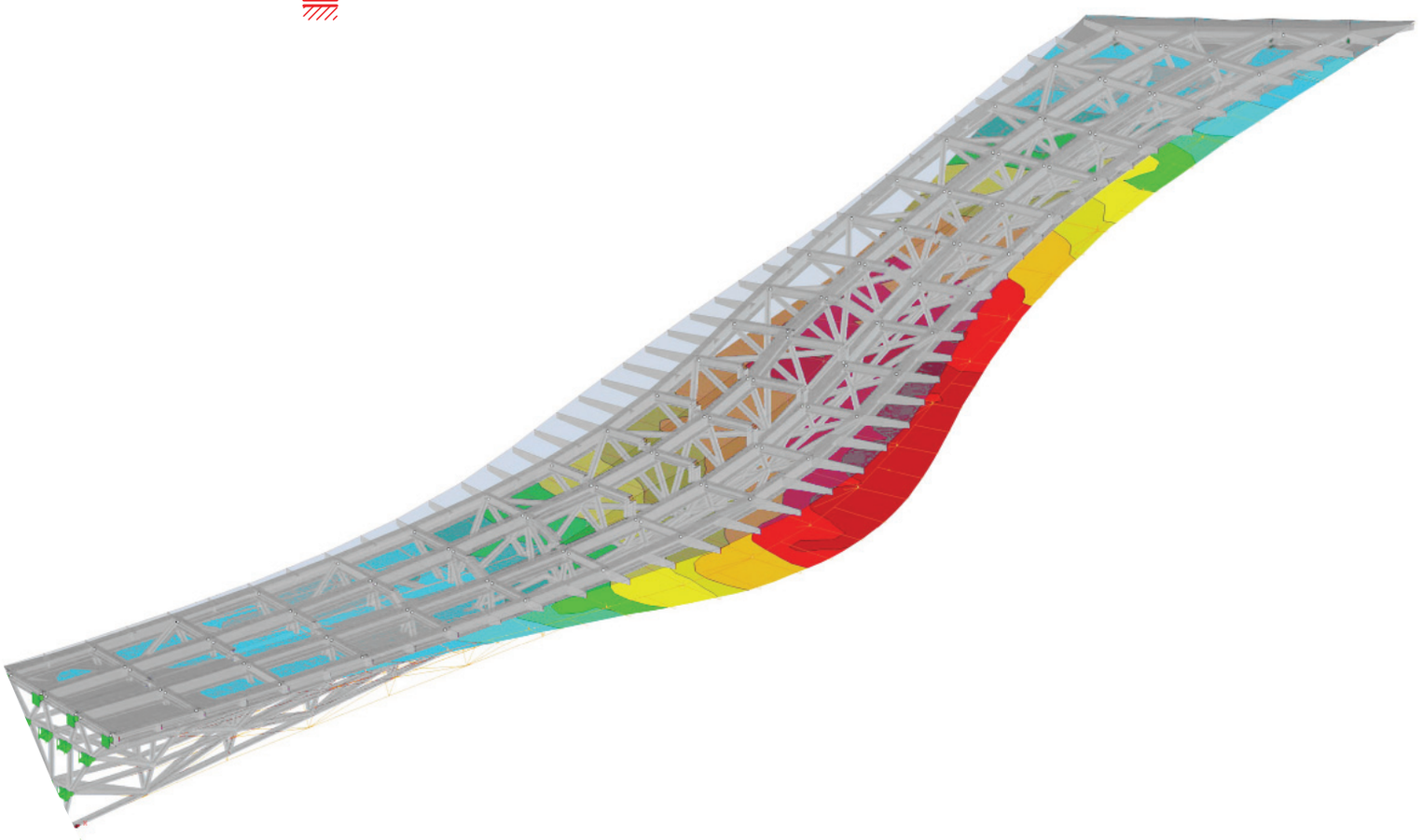
Im Arretierungszustand ist die Brücke als einseitig eingespannter Balken mit frei aufliegendem Ende konzipiert. Die Fixierhydraulik auf der Hermannparkseite ermöglicht die Aufnahme des Einspannmomentes. Die Gegengewichtslänge hinter der Drehachse beträgt ca. 6m und wird zur Gewichtssteigerung als geschweißter, vollwandiger Blechträger (t=40mm) ausgeführt. Die Brücke mit einer Länge von 48m wird als Fachwerkskonstruktion mit orthotroper Platte (Fahrbahnblech t=15mm, Längs- und Querrippen) ausgeführt. Die variable Fahrbahnbreite wird durch auskragende Blechschwerter erzeugt, sodass der Fachwerksobergurt mit einheitlicher Breite ausgeführt werden kann. Die Fachwerksstäbe werden aufgrund ihrer komplexen Knotengeometrie aus Rundrohren in Feldabschnitten von e=2400 mm ausgeführt. Die Lagerung auf Seite der Urania wird als bewegliches Auflager ausgebildet. Damit können die Verformungen des Tragwerks im Öffnungs- und Schließvorgang zwängungsfrei aufgenommen werden.

Der Korpus der Fachwerkskonstruktion wird in Anlehnung an den historisch begründeten „Fassadenträger" (Ohmann und Hackhofer) - "losgelöst von einer statisch determinierten Dimension" - von einer Membran umspannt. Die Membran wird an den Rändern geklemmt und mittels eines innenliegenden Spannseils in ihre konkave Form gebracht.

Kippzustand (A)

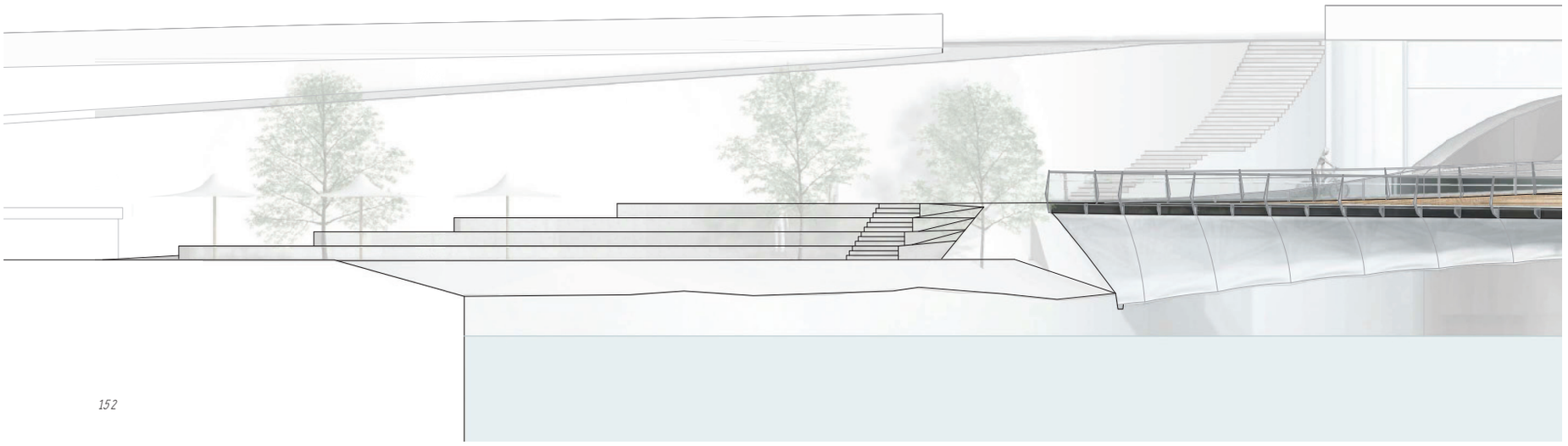


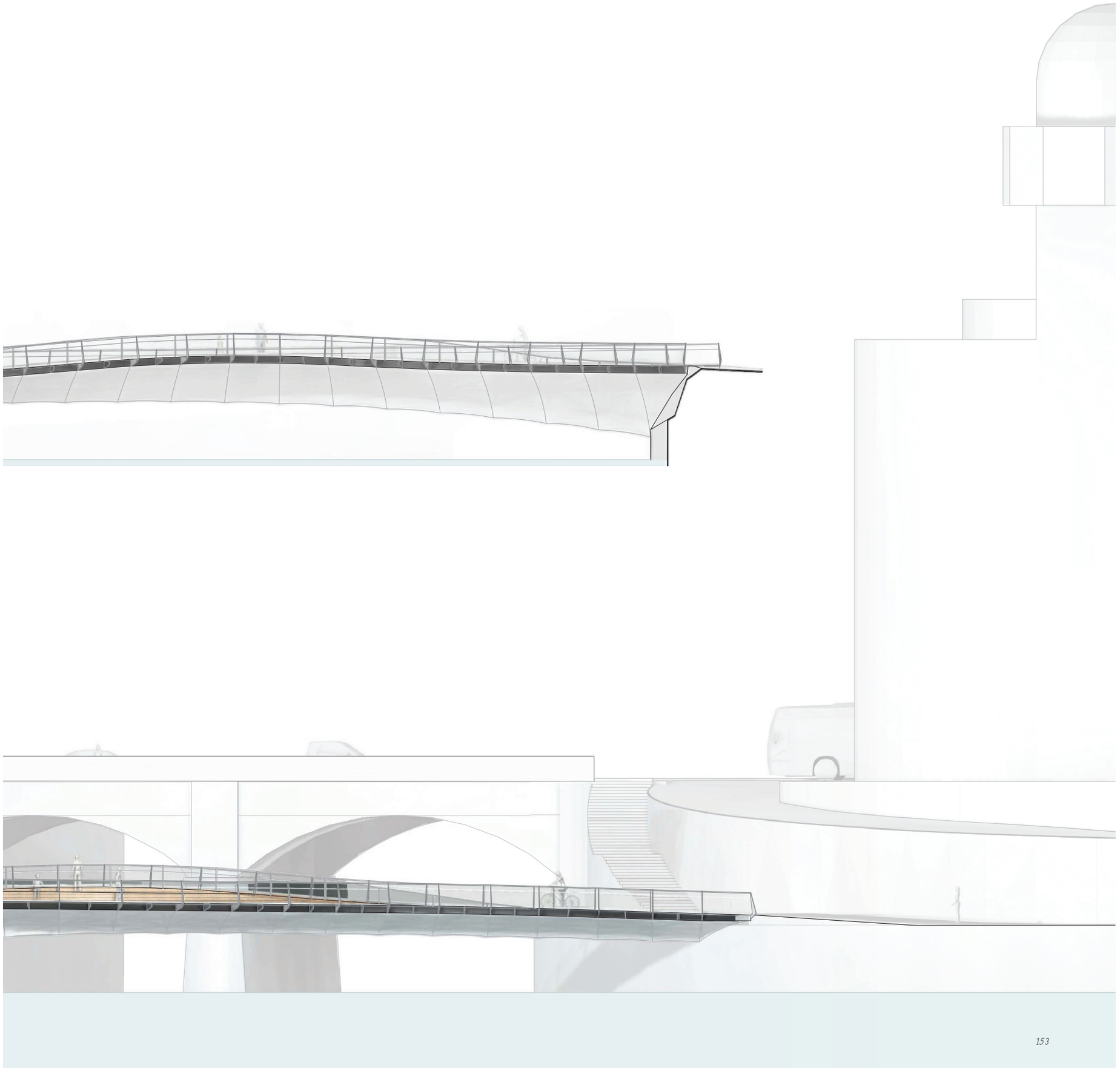
Arretierungszustand (B)

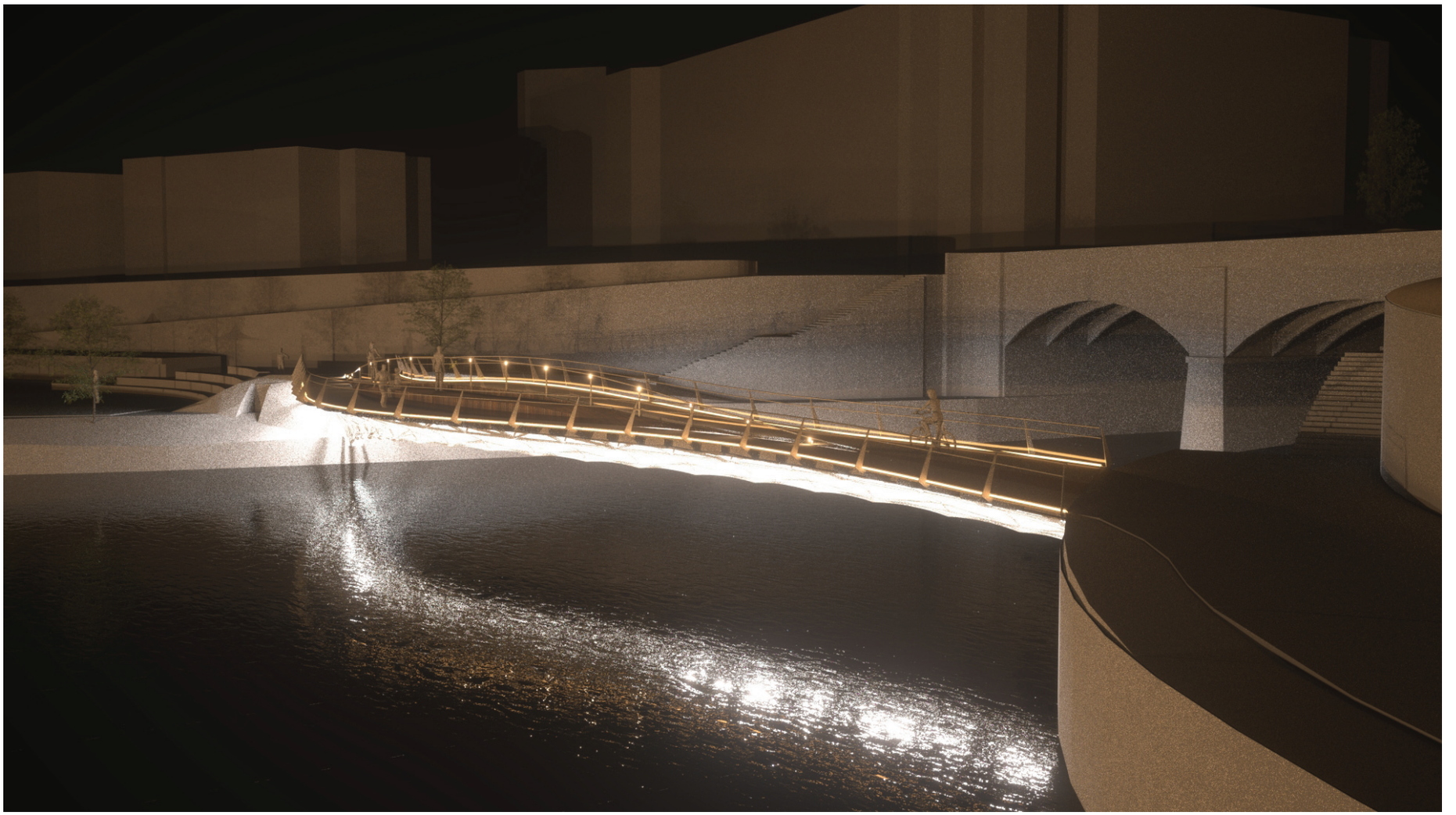


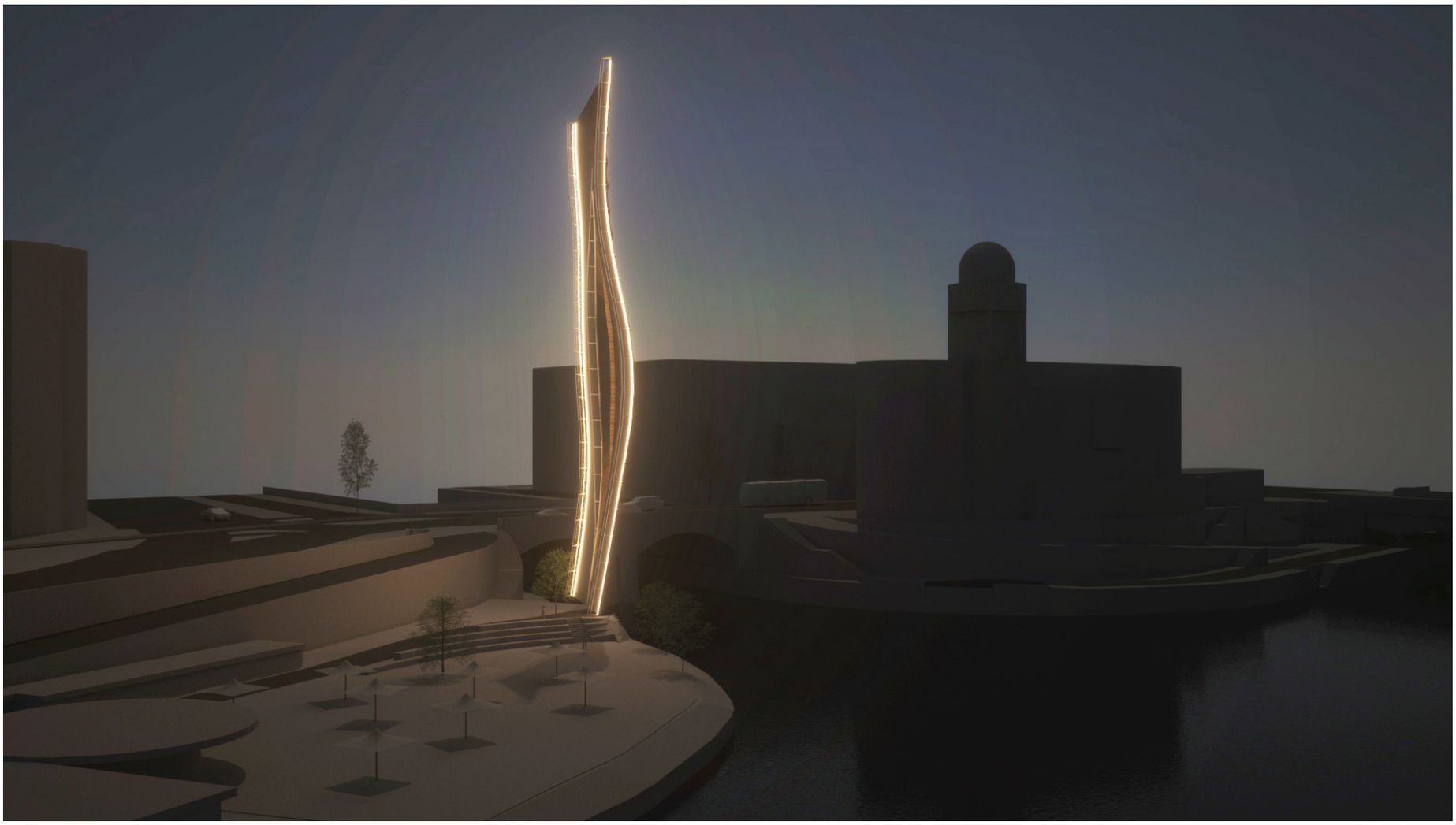
SÜD_ANSICHT
M 1:200

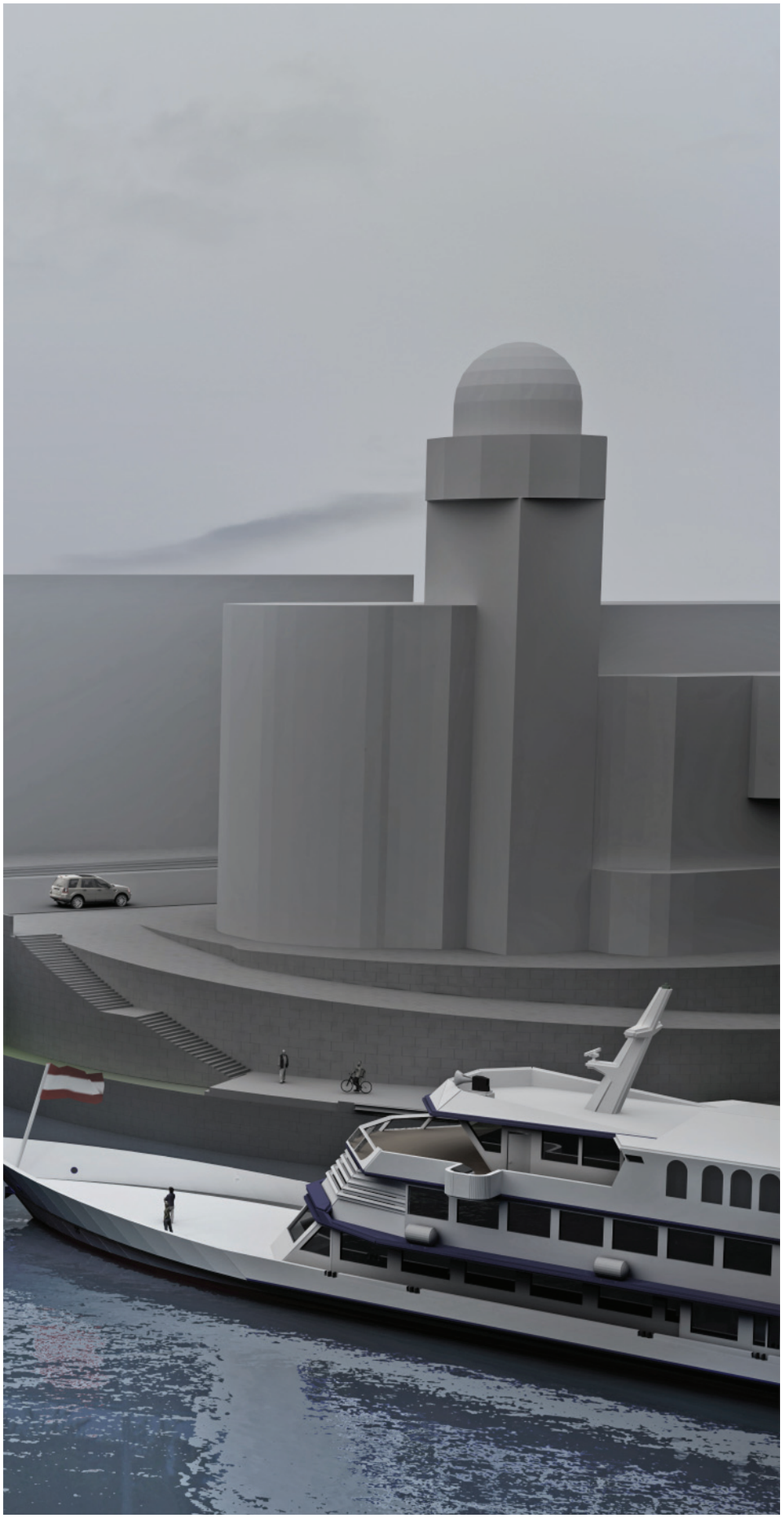
NORD_ANSICHT
M 1:200

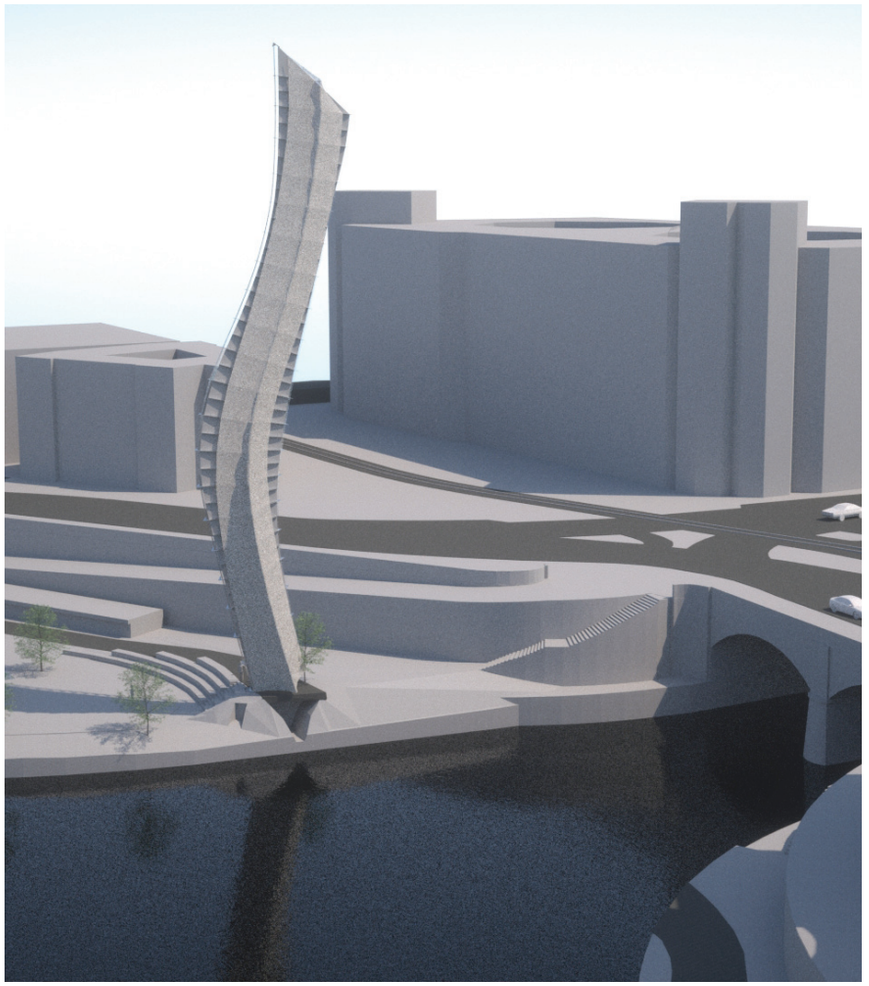


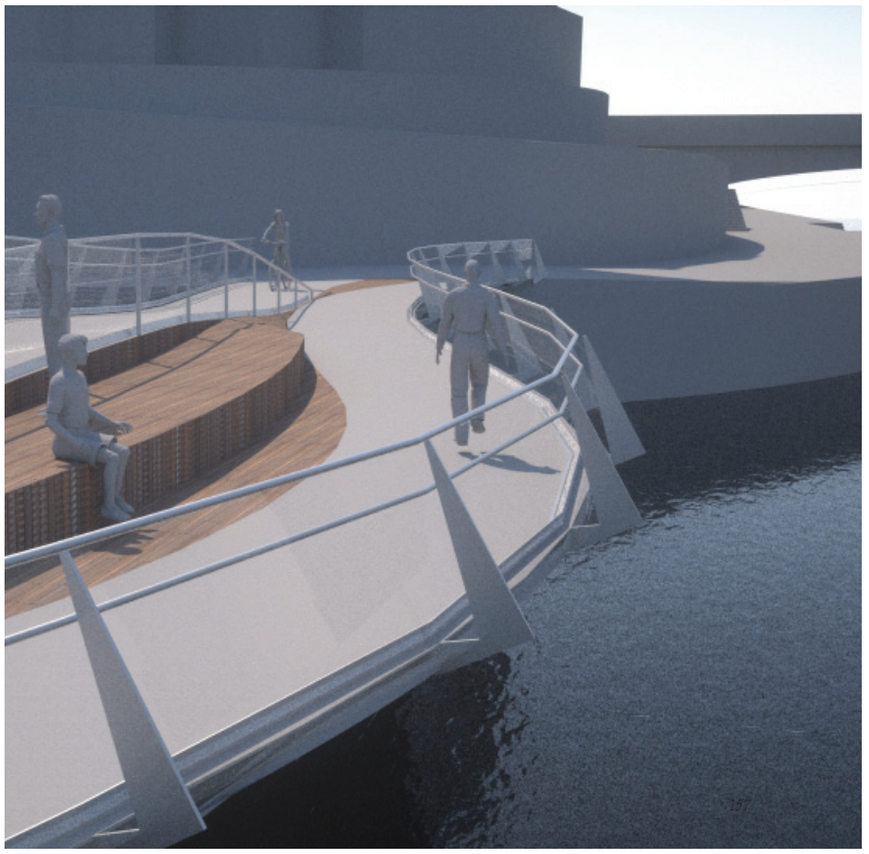












Resümee

Grundsätzlich kann gesagt werden, dass die Auseinandersetzung mit dem gewählten Thema insofern eine besondere Herausforderung darstellte, als am Beginn der Diplomarbeitsphase noch nicht abschätzbar war, inwieweit ein Brückenentwurfsprozess, mit welchem intuitiv Variantenstudien erstellt werden können, parametrisierbar gemacht werden kann und letztlich auch sinnvolle Tragwerke für einzelne Entwurfsintentionen herauskommen können.

Wettbewerbe

Zunächst ist hervorzuheben, dass es sich im Rahmen der Beschäftigung mit dem Entwurfsthema als vorteilhafte Konstellation erwiesen hat, dass zwei Wettbewerbe mit nahezu identischen Aufgabenstellungen ausgeschrieben wurden. So dienten zum einen die Rahmenbedingungen des von der Vereinigung der Österreichischen Zementindustrie im März 2009 ausgeschriebenen Studentenwettbewerbs "Concrete Student Trophy 2009" im Wesentlichen zur Parameterfindung sowie zur Entwicklung des Grundkonzepts des Entwurfsprozesses. Wenige Monate später konnte zudem im Zuge der Teilnahme an dem Realisierungswettbewerb "Connecting Link" der Stadt Wien im September 2009 eine praxisbezogene Verifizierung der Ergebnisse und Korrektur bzw. Weiterentwicklung des Prozesses stattfinden.

Zu den Erfahrungen aus den Wettbewerben muss im Hinblick auf den Brückenentwurf als "Königsdisziplin der BauingenieurInnen" bemerkt werden, dass aus der Sicht der BauingenieurInnen der Begriff der Innovation, wie er tendenziell von ArchitektInnen verstanden wird, vor allem aus Gründen der wirtschaftlichen Machbarkeit sehr nahe an jenem der Utopie zu liegen scheint und Brückenkonstruktionen sehr schnell als „Nonstandard" deklariert werden. Dies fiel auch schon - trotz der positiven Reaktion der JurorInnen in Form eines Anerkennungspreises - beim Studentenwettbewerb auf. Generell wird mE nur allzu gerne auf bereits bewährte Systemlösungen zurückgegriffen.

Für die Teilnahme am Realisierungswettbewerb ergab sich die Möglichkeit zur Zusammenarbeit mit dem Ingenieurbüro "Structural Design Olipitz". Dabei stellte sich etwa beim Überprüfen der Ergebnisse heraus, dass einer der zwei eigens für die Vorbemessung entwickelten Korrekturfaktoren in manchen Fällen die Form zu extrem veränderte, sodass die Konstruktion ab dem zweiten Drittel an der Brückenspitze eine zu geringe Höhe aufwies. Aufgrund der spannenden Ästhetik, die sich aus einem eigentlichen Konzeptfehler ergab und das Tragwerk an die Grenzen der Machbarkeit brachte, wurde dieser Formvorschlag von den Ingenieuren jedoch nicht als unmachbar abgetan, sondern vielmehr wurde nach einer möglichen Lösung gesucht.

Eine der gravierendsten Änderungen, die im zweiten Wettbewerb aus Sicht des leitenden Ingenieurs Michael Olipitz relativ kurzfristig gemacht werden musste, war die Konstruktionsänderung von einer gedachten flächenhaften Stahlplatten- in eine Fachwerkkonstruktion, um die Gesamtkonstruktion leichter zu machen. Da die Zerlegung der gekrümmten Flächen in ein Stabwerk wesentlich einfacher handhab- und herstellbar ist, ist mE das "Leichtermachen" nicht nur aufgrund des physischen Gewichts, sondern auch aus monetären Gründen geschehen.

Generell musste meinerseits in der Zusammenarbeit sowie in Gesprächen mit IngenieurInnen festgestellt werden, dass stets versucht wurde, das Originalsystem für die Berechnung extrem zu vereinfachen. Aus meiner Sicht geschah dies nicht zuletzt deshalb, weil eine komplexe Volumsgeometrie als Tragwerk in einer Statiksoftware praktisch schwer handhabbar und deren Eingabe unüblich oder zumindest mit einem extremen Aufwand verbunden ist. Aus diesem Grund konnte - fernab der Diskussion rund um die Machbarkeit - mein Ziel, die originale Geometrie als tragfähiges Volumen exakt zu überprüfen, nicht erreicht werden.

Prinzipiell hat sich im Rahmen der beim Realisierungswettbewerb durchgeführten Überprüfung einer komplexeren Geometrie herausgestellt, dass mit dem entwickelten Entwurfstool außerordentlich brauchbare Brückentragwerke generiert werden können, solange diese keine "Extremstverformungen" annehmen. Dies bedeutet, dass die Berechnungsergebnisse umso ungenauer werden, je verzerrter die Geometrien sind. Da die Brückengeometrie laut Konzept durch wenige signifikante Schnittprofile und eine überschaubare, möglichst regelmäßige geordnete Anzahl von Punkten definiert wird, konnten die Ergebnisse außerordentlich einfach und schnell in anderen Softwareprogrammen weiterverarbeitet werden. In der Praxis hat sich die Parametrisierung des Entwurfes als extremer Vorteil herausgestellt, da etwa laut Ansicht der Ingenieure beispielweise die vom "Entwurfstool" vorgeschlagene Brückenhöhe sogar verringert werden konnte, wobei diese Änderung letztlich nur mit der Änderung einer Zahl verbunden war und der Entwurf somit durch einen geringen Aufwand innerhalb von einer Minute neu generiert werden konnte.

Brückengenerierungsprozess

Zum Brückengenerierungsprozess als Hauptteil dieser Arbeit ist anzumerken, dass in der Entwicklungsphase immer wieder Entscheidungen getroffen werden mussten, die die Möglichkeit einer Vielzahl an unterschiedlichen Ergebnissen zu einem variierbaren Lösungstypen eingeschränkt haben. So mussten insbesondere aus konstruktions- bzw. materialspezifischen Gründen oder auch einem unverhältnismäßig hohen Arbeitsaufwand sowie mangelnden Scriptingkenntnissen die Entscheidungsfreiheiten im Entwurf zugunsten eines sinnvollen weiterverarbeitbaren Ergebnisses stark begrenzt werden. Innerhalb dieser eingeschränkten Möglichkeiten lassen sich unterschiedlichste Variationen des parametrisierten Grundtypus erzeugen, wobei diese als gleichwertig zu betrachten sind. Dies bedeutet, die Variierbarkeit wurde nicht geschaffen, um einen optimalen Entwurf generieren zu können, sondern um viele einzelne Varianten, die sich aufgrund ihrer gemeinsamen Entstehungsprozessgeschichte in ihrem Erscheinungsbild und ihren Qualitäten ähneln, erzeugen zu können.

In diesem Kontext kann auf die Ansicht von Greg Lynn verwiesen werden, der sich zu den Entwurfsstudien seiner "Embriological Houses" dergestalt äußerte, dass er diese Häuser ähnlich wie seine Kinder betrachten und keines von ihnen bevorzugen würde.[180]

Als einer der größten Nachteile bei der Entwicklung von Prozessen mit "Mashup-Tools" hat sich herausgestellt, dass in der Vernetzung der Module keine Schleifenbildungen möglich sind. Dementsprechend ist an dieser Stelle insbesondere darauf aufmerksam zu machen, dass es zum Beispiel nicht möglich ist, dass ein Modul einen Wert an ein zweites Modul weitergibt und das zweite Modul den bearbeiteten Wert über seinen Ausgang wieder zurück an den Eingang des ersten Moduls gibt. Schleifen sind nur innerhalb der einzelnen Scripte möglich. Dazu tritt häufig noch das Problem, dass die Ein- und Ausgänge eines Moduls immer erst nach Ablauf des internen Scripts aktualisiert werden. Diese Eigenschaften erschweren insgesamt die Zerlegung des Prozesses in einzelne Module.
Trotz der genannten Schwierigkeiten ergibt sich in Hinblick auf den Entwurfsprozess bei der Verwendung von Mashup-Systemen mE der größte Vorteil in der Möglichkeit der Organisation eines flexiblen, offenen Prozesses über die Vernetzung einzelner Funktionsmodule zu einem gesamten Entwurfssystem. Die einzelnen Module können dabei nicht nur verändert bzw. erweitert, sondern auch in anderen Kombinationen für die Generierung neuer Entwürfe verknüpft werden. In diesem Zusammenhang kann etwa das Steigungskorrektur-Modul angeführt werden, welches mit Sicherheit nicht nur für die Bewertung und Korrektur von Brückenwegen einsetzbar ist. Weiters können die genannten Schaltungen besonders in Animationssoftwareprogrammen mit zahlreichen Simulationstools und Plugins zusammengeschalten werden.
Meines Erachtens könnte der/die "ArchitektIn der Zukunft" im Laufe seiner/ihrer Auseinandersetzung mit dieser Art von Modulen ein Repertoir an individuellen Plugins entwickeln, die im Entwurfsprozess immer wieder verwendet werden können, wodurch aus den vernetzten Funktionsfragmenten eine eigene "Handschrift" resultieren könnte. Diese lässt sich jedoch einfach kopieren. So hat Helmut Schober die letzte Brückenvariante für den Wettbewerb "Connecting Link" mittels des in dieser Arbeit entwickelten Entwurfstools generiert, sodass diese mit meiner "Handschrift" entworfen wurde.

Ästhetik der Ergebnisse

Einer der interessantesten Aspekte bezüglich der ästhetischen Erscheinung der generierten Brückengeometrien war das in Zusammenhang bringen von Ansicht und Grundriss der Konstruktion, das vor allem über die eigens entwickelten Korrekturfaktoren hergestellt und eben durch eine Fehleingabe besonders stark sichtbar wurde. So nimmt die Brückenkubatur in der Ansicht nicht etwa den klassischen Momentenverlauf in Form einer Parabellinie an, sondern verbeult beziehungsweise verformt sich aufgrund der Wegbreitenänderungen im Grundriss (siehe Grafik S.104)- da sich ja die Widerstandsmomente, das Eigengewicht etc. an dieser Stelle ändern und dadurch ein konstanter Brückenquerschnittsverlauf tendenziell nicht sinnvoll ist.
Generell kann man mE bezugnehmend auf Jörg Gleiter (S.41) die gesamte Brückenform als modernes Ornament betrachten, da diese ihrer Gestalt nach auf den digitalen, virtuellen Entstehungsprozess hinweist. Je nachdem, wieviele der einzelnen Funktionen in der Entstehung angewendet wurden, umso besser kann am Endergebnis der Entstehungsprozess bzw. der Hinweis auf eine digitale Generierung abgelesen werden.

Zusammenfassend

Wie bereits im allgemeinen Teil festgestellt wurde, kann gesagt werden, dass die aktuelle Diskussion bezüglich des Einsatzes digitaler Entwurfsmethoden in der Architektur in die Richtung einer Entwicklung von leistungsoptimierten, effizienten Bauteilen bzw. Architekturen und der Bildung einer lückenlosen digitalen Planungskette, die von der Entwurfs- bis zur Ausführungsphase reicht, geht. Vor diesem Hintergrund lässt sich auch die vorliegende Arbeit in ihrer Grundmotivation und Haltung einordnen. Jedoch wurde versucht, den Begriff der Effizienz und die alleinige Fokussierung auf das Brückentragwerk über die Gestaltung eines offenen Prozesses, in dem verschiedene Aspekte einfließen können, etwas zu relativieren. Dies bedeutet, dass das Erreichen eines optimierten Tragwerkes nicht primär im Vordergrund stand, sondern ein Vorschlag für eine wahrscheinlich sinnvolle Brückenkonstruktion unter Berücksichtigung unterschiedlichster Parameter geliefert werden sollte. Dies geschah nicht zuletzt aufgrund der Überzeugung, dass bei einer starren Zielsetzung und den knappen Beurteilungskriterien des Computers viele möglicherweise aus anderen Aspekten interessantere Lösungen ausgeschlossen werden könnten. Aus diesem Grund werden dem/der BenutzerIn auch alle Ergebnisschritte zugänglich gemacht, damit dieser/diese nicht gezwungen wird, die vom Rechner als am geeignetsten beurteilte Brückengeometrie weiterverarbeiteten zu müssen, sondern aus unterschiedlichsten Gründen eine andere Variante bzw. Entwicklungsstufe zur Weiterverarbeitung wählen kann. Demzufolge wurde insgesamt ein Vorschlag bzw. eine Idee formuliert, wie ArchitektInnen in Zukunft in ihren Entwürfen mittels digitaler Methoden das Tragwerk als Teil eines komplexen Formfindungsprozesses in den Entwurfsprozess integrieren könnten.

Abschließend kann festgestellt werden, dass das entwickelte Entwurfstool sehr weit und brauchbar von der Formfindungsphase bis zur Übergabe der erzeugten Geometriedaten in eine andere CAD-Software in den Entwurfsprozess eingebunden werden konnte. Da mit diesen neuen digitalen Entwurfstools nach wie vor Ideen aus den 1970er Jahren mit relativ großem Zeitaufwand realisiert und für sinnvolle Ergebnisse in der Regel umfangreiche Basiskenntnisse benötigt werden, möchte ich hinsichtlich einer zukünftigen Standardisierung der generierten Architekturen und der praktischen Alltagstauglichkeit dieser Entwurfstools im Planungsprozess mit Vannevar Bushs Worten schließen: „*that day is not yet here*," und um die Spannung weiter aufrecht zu erhalten "*but has come far closer...*"[181].

180 Greg Lynn im Interview mit Ingeborg Rocker, Zeitschrift, Architectural Design, Programming Cultures, July/August 2006, Wiley Academy, S.92.

181 Michael Friedewald, Der Computer als Werkzeug und Medium, Verlag für Geschichte und Naturwissenschaften und der Technik, Berlin, Diepholz 1999, S.69

Abbildungsverzeichnis

Im Folgenden werden ausschließlich Abbildungen aus externen Quellen angeführt. Die restlichen Abbildungen wurden entweder mit der eigenen Fotokamera aufgenommen oder mit unterschiedlichen Studentenversionen von Softwareprogrammen wie Autocad, Cinema4d, Photoshop erzeugt. Die Grafiken auf Seiten 32 (Abb. 24-24) und 34 (Abb. 26) sind im Zuge des Seminars Simulationstechnik (2007/2008) am Institut für Architektur und Medien an der TU-Graz enstanden. Die Verformungsbilder auf den Seiten 150 und 151 wurden mit der Software RFEM im Büro "Structural Design Olipitz" angefertigt. Die Ruckzuckgrafik auf Seite 64 oben wurde an der Technischen Universität Graz am Institut für Tragwerksentwurf erzeugt.

Abbildung 2: "Enwicklungsgeschichte des Computers", Edgar Vordran, 1982, VDE-Verlag GmbH, Berlin und Offenbach, S.66.

Abbildung 3: Ebd.,S.91 .

Abbildung 4: Ebd.,S.79

Abbildung 5: "Der Computer als Werkzeug und Medium", Michael Friedewald, Verlag für Geschichte und Naturwissenschaften und der Technik, Berlin, Diepholz 1999, S.5.

Abbildung 6: Ebd., S.85.

Abbildung 7: Ebd., S.104.

Abbildung 8: "The Force is in the mind: The making of Architecture", Elke Krasny, Architekturzentrum Wien, Birkhäuser, 2008, S.165.

Abbildung 9: "The Archtecture Machine", Nicholas Negroponte, The MIT Press Cambridge, Massachusetts, 1970,S.18.

Abbildung 10: "The Logic of Architecture: Design, Computation, and Cognition", William J Mitchell, The MIT Press Cambridge, Massachusetts, 1990, S.153.

Abbildung 11: Ebd. S.170

Abbildung 12: Georg Nees - Generative Computergrafik, 1. Auflage 1969, Verlag: Siemens Aktiengesellschaft Berlin München, by Siemens Aktiengesellschaft, Bild 38, S.14.

Abbildung 13: Ebd., S.242.

Abbilung 14: http://www.imachination.net (07.03.2010)

Abbildung 15: http://mysite.pratt.edu/~llaurola/cg550/nake.jpg (07.03.2010)

Abbildung 16: "Formel, Farbem Form", Georg Nees, Springer Verlag, 1. Auflage 1994, S.252.

Abbildung 17: "Ästhetische Probleme der Architektur unter dem Aspekt der Informationsästhetik", Manfred Kiemle, Verlag Schnelle Quickborn, Deutschland, 1967, S.65

Abbildung 18: Ebd., S.69.

Abbildung 19: "The Engeneering Design Revolution", David E. Weisberg, e-Book 2008, http://www.cadhistory.net (28.02.2010).

Abbildung 22: http://www.dma.ufg.ac.at/app/link/Grundlagen:3D-Grafik/module/13444?step=all.

Abbildung 25: "Animate Form", Greg Lynn, Princeton Architectural Press New York, 1999. S.91.

Abbildung 31: http://wiki.mcneel.com/labs/explicithistory/home, (Stand 17.03.2010).

Abbildung 32: http://www.alexhogrefe.com/blog/2009/6/8/multi-object-orientation-grasshopper-script-2.html (Stand 17.03.2010).

Abbildung 33: http://www.grasshopper3d.com/photo (Stand 17.03.2010).

Abbildung 37: http://www.patrikschumacher.com/ (Stand 18.03.2010).

Abbildung 38: Ebd.

Abbildung 39: "Digital Materiality in Architecture", Gramazio, Kohler, Lars Müller Verlag, 2008, linkes Bild S.98 und rechtes Bild S.100-101.

Abbildung 40: "CAM of Freeforms in Architecture", Oliver Fritz, Hg. Institut für Raumplanung der Hochschule Liechtenstein, Michael Imhof Verlag GmbH & Co. KG, S.35

Abbildung 41: "Digital Materiality in Architecture", Gramazio, Kohler, Lars Müller Verlag, 2008, linkes Bild S.106.

Abbildung 42: "Latente Utopien", Zaha Hadid/Patrik Schumacher, steirischer Herbst 2002, S.102.

Abbildung 43: "The Archtecture Machine", Nicholas Negroponte, The MIT Press Cambridge, Massachusetts, 1970,S.60.

Abbildung 44: "Latente Utopien", Zaha Hadid/Patrik Schumacher, steirischer Herbst 2002, S.103.

Abbildung 45: Arch +, Zeitschrift für Architektur und Städtebau, Ausgabe 189, Oktober, 2008. S.66.

Abbildung 46: Ebd.

Abbildung 47: "From Control to Design", Editiert von Sakamoto u. Ferré, Verlag Actar-D 2008, S.108-109.

Abbildung 48: Ebd., S.104 u. 110.

Abbildung 49: Ebd., S.113.

Abbildung 50: "Articukated Grounds: Mediating Enironment and Culture", Architectural Association London, AA Agendas No. 7. Edited by Anne Save de Beaurecueil and Frank Lee, 2009, S.17.,105,69,104,109.

Abbildung 57: http://www.sofistik.de/infoportal/ (Stand 20.04.2010).

Abbildung 58: http://bridges.transportation.org/Pages/Georgia.aspx, (Stand 20.04.2010).

Abbildung 59: http://www.sofistik.de/infoportal/ (Stand 20.04.2010).

Abbildung 60: http://www.sofistik.de/infoportal/ (Stand 20.04.2010).

Abbildung 62: http://www.bestech.co.uk/ (Stand 20.04.2010).

Abbildung 63: http://www.sofistik.de/infoportal/ (Stand 20.04.2010).

Abbildung 64: http://www.larsausa.com/ (Stand 20.04.2010).

Abbildung 65: Google Maps, http://maps.google.at/ (Stand 25.04.2010).

Abbildung 66: Bing Maps, http://www.bing.com/maps/, 3d Ansicht (Stand 25.04.2010).
Bild wurde nachträglich verändert.

Quellenangaben

Bibliografie

Bücher

"A Pattern Language", Christopher Alexander, Oxford University Press, New York, 1977.

"A Pattern Language", Klaus W. Gartler, Technische Universität Graz, Seminar für Planungsmethoden entstandenen Übersetzung, 1979.

"Algorithmic Architecture", Kostas Terzidis, Architectural Press is an imprint of Elsevier, First Edition 2006

"Architekt – Ingenieur", Arbeiten am Institut für Entwerfen und Konstruieren, Kurt Ackermann, Stuttgart, Karl Krämer Verlag, 1997.

Architectural Design, Programming Cultures, Ausgabe July/August 2006, Wiley-Academy.

"architektur_theorie.doc", Hrsg. Gerd de Bruyn, Stephan Trüby, Birkhäuser Verlag, 2003.

"Ästhetische Probleme der Architektur unter dem Aspekt der Informationsästhetik", Manfred Kiemle, Verlag Schnelle Quickborn, Deutschland, 1967.

"Austreibung des Geistes aus den Geisteswissenschaften. Programme des Poststrukturalismus." Friedrich Kittler, Paderborn: Schöningh 1980.

"Being digital", Nicholas Negroponte, Vintage Books a division of Random House, 1995 USA.

"Bridge Engineering: A Global Perspective", Leonardo Fernández Troyano, Thomas Telford Publishing, Colegio de Ingenieros de Caminos, Canales y Puertos, 2003.

"Brücken: Ästhetik und Gestaltung", Fritz Leonhardt, Deutsche Verlags-Anstalt, 3. Auflage 1990.

"Brücken: Kühne Konstruktionen über Flüsse Täler Meere", David J. Brown, , deutsche Übersetzung, Verlag Georg D. W. Callwey GmbH & Co. KG, 2005.

"Brückenbau im 20. Jahrhundert", Dirk Bühler, Verlag Georg D. W. Callwey GmbH & Co. KG, 2005

"CAM of Freeforms in Architecture", Oliver Fritz, Hg. Institut für Raumplanung der Hochschule Liechtenstein, Michael Imhof Verlag GmbH & Co. KG.

"Coding for Fun", Gottfried Wolmeringer, Galileo Press, 1. Auflage, 2009.

"Der Computer als Werkzeug und Medium", Michael Friedewald, , Verlag für Geschichte und Naturwissenschaften und der Technik, Berlin, Diepholz 1999.

"Die Verdrängung des Ornaments – Zum Verhältnis von Architektur und Lebenspraxis", Michael Müller, Suhrkamp Verlag, Erste Auflage 1977.

"Digitales Entwerfen", Marco Hemmerling und Anke Tiggemann, Wilhelm Fink GmbH & Co. Verlags-KG, 2010.

"Digital Materiality in Architecture", Gramazio, Kohler, Lars Müller Verlag, 2008.

"End-user development: tools that empower users to create their own software solutions", A. Sutcliffe and N. Mehandjiev, Communications of the ACM, vol. 47(9), 2004.

"Entwicklungsgeschichte des Computers", Edgar P. Vorndran, 1982, VDE-Verlag GmbH Berlin und Offenbach.

"From Control to Design", Editiert von Sakamoto u. Ferré, Verlag Actar-D 2008.

"Fußgängerbrücken: Konstruktion Gestalt Geschichte", Ursula Baus, Mike Schlaich, Birkhäuser Verlag 2008,

"Grundlage der Tragwerkslehre", Franz Krauss, Wilfried Führer, Hans J. Neukäter, Verlagsgesellschaft Müller, 2002.

"Guidance for good bridge design", International Ferderation for Structural Concret (fib), 2000.

"Kulturtechnik Entwerfen", Hg. Gethmann u. Hauser, 2009 tanscript Verlag Bielefeld Deutschland.

"Latente Utopien", Zaha Hadid/Patrik Schumacher, steirischer Herbst 2002.

"leicht weit: Light Structures", Jörg Schalich, Hg. Annette Bögle, Peter Cachola Schmal, Ingeborg Flagge, Prestel Verlag Deutschland, 2004.

"Massivbau in ganzer Breite", Förderverein Massivbau der TU München e.V. u. Roland Niedermeier, Springer Verlag Berlin Heidelberg, 2005.

"Re-coded: Studio Rocker", Ingeborg Rocker, , Aedes Berlin, Juni 2005.

"Robert Maillart, Brückenschläge", Höhere Schule für Gestaltung Zürich, Schriftenreihe Nr. 13, Redaktion Claude Lichtenstein, 1990.

"Service Mashups: The New Generation of Web Applications", D. Benslimane, S. Dustdar, and A. Sheth , IEEE Internet Compting, vol. 12(5), 2008.

"Soziale Systeme", Niklas Luhmann, Grundriss einer allgemeinen Theorie, Suhrkamp Verlag, Erste Auflage 1987.

"The Logic of Architecture: Design Computation and Cognition", William J. Mitchell,

MIT Press Cambridge, Massachusetts, 1990.

"Tragwerkslehre in Beispielen und Zeichnungen", Leicher Gottfried, Werner Neuwied Verlag, 2006,

"User-Driven Requirements Engineering for Mobile Social Software" N. Seyff and F. Graf, Proceedings of 3rd. International Workshop on Social Software Engineering, 24. Feb. 2010 Paderborn, Germany.

Zeitschriften:

Architectural Design, Digital Cities, Parametricism - A New Global Style for Architecture and Urban Design,Ausgabe 79, Julie/August 2009, Editoren Neil Leach und Hellen Castle.

Arch +, Zeitschrift für Architektur und Städtebau, Ausgabe 148, 1999.

Arch +, Zeitschrift für Architektur und Städtebau, Ausgabe 188, Juli, 2008.

Arch +, Zeitschrift für Architektur und Städtebau, Ausgabe 189, Oktober, 2008.

Arch +, Zeitschrift für Architektur und Städtebau, Ausgabe 195, November, 2009.

Grazer Architektur Magazin, GAM06, Herausgeber Architektur Fakultät Technische Universität Graz, 2010.

Internet

e-Book:
The Engeneering Design Revolution, David E. Weisberg, e-Book 2008,
http://www.cadhistory.net (28.02.2010).

Second Life Hinweis:
http://secondlife.com/?v=1.1 (Stand 14.02.2010).

Gordon Moor:
http://www.intel.com/cd/corporate/pressroom/emea/deu /archive/2005/212674.htm
(Stand 12.02.2010).

Leo Dictionary, Online Wörterbuch, http://dict.leo.org,

Eine Liste von einigen CAD-Softwareprogrammen:
http://de.wikipedia.org/wiki/Liste_von_CAD-Programmen,
http://www.experiencefestival.com/a/List_of_CAD_companies_-_Commercial_CAD_
Software/,
http://crunkish.com/top-ten-cad-software

Geschichte der Animation:
http://www.dma.ufg.ac.at/app/link/app (Stand 15.03.2010).

Geschichte 3d Studio Max:
http://www.maxunderground.com/the_history_of_3d_studio, (Stand 13.03.2010).

CFD-Simulationssoftware:
Next Limit Technologies, http://www.nextlimit.com/techno_fluid.php (Stand 15.03.2010).

Skriptsprachen und Mashup-Tool Informationen:
http://usa.autodesk.com/adsk/servlet/item?siteID=123112&id=2309147, (16.03.2010).
http://www.py4d.com, (16.03.2010);
http://blog.rhino3d.com/2010/03/python-scripting-in-rhino-os-x.html, (16.03.2010),
http://wiki.blender.org/index.php/Doc:Manual/Extensions/Python, (16.03.2010),
http://blog.rhino3d.com/2010/03/python-scripting-in-rhino-os-x.html; (16.03.2010.)
http://www.grasshopper3d.com/ (Stand 17.03.2010).
http://wiki.mcneel.com/labs/explicithistory/home, (Stand 17.03.2010).
http://www.tutorials.de/forum/attachments/cinema-4d/44406d1229278877-xpresso.jpg,
(Stand 17.03.2010).

Mashup-Tool Information:
http://usa.autodesk.com/adsk/servlet/index?siteID=123112&id=9469002, (16.03.2010)

Processing:
http://processing.org

NetLogo:
http://ccl.northwestern.edu/netlogo, (Stand 17.03.2010).

Beispiele für Masscostumization im Internet:
http://www.tailorstore.de/, http://www.zazzle.de/,
http://nikeid.nike.com/,
http://www.lbs-city.de/ (Stand 24.03.2010).

Peter Eisenmann:
Interview von Urban Center Books, About Books and book collecting,
YouTube Video, http://www.youtube.com/watch?v=JvdDQzT56ks (03.04.2010);
Interview mit Peter Eisenmann von Robert Locke am 27.07.2004,
http://archinect.com/features/article.php?id=4618_0_23_0_M (02.04.2010);
Peter Eisenman, Vortag an der Vanderbilt University
http://www.youtube.com/watch?v=AJMMnb0qrXA&NR=1, (Stand 05.04.2010).

Rem Koolhaas:
Interview am 27.03.2006, Spiegel Online International,
http://www.spiegel.de/international/spiegel/0,1518,408748,00.html (02.04.2010).

Interview am 12.06.2008, Zeit Online,
http://www.zeit.de/2008/24, Koolhaas-Interview (02.04.2009).

Peter Zumthor:
Radiointerview, http://irge-uni-stuttgart.de/lehre/themenreihe/identitaet.html,(03.04.2010).

Santiago Calatrava:
Interview mit Charlie Rose, http://kempton.wordpress.com/2009/07/30/calgary-peace-
bridge/ (09.04.2010).

Milau Bridge Information:
http://de.wikipedia.org/wiki/Viaduc_de_Millau, (12.04.2010)

Definiton Bridge Architect bzw. Bridge Designer:
http://en.wikipedia.org/wiki/Bridge_architect (11.04.2010).

Winkelgewichte Berechnung:
Baustatik 1 Skript des Instituts für Baustatik, TU-Graz
http://www.ifb.tugraz.at/backup_old_homepage/educ/teaching_material/Baustatik/
Baustatik1_Skriptum_IBK.pdf, (Stand 04.04.2010).

Finite Elemente Methode Information:
http://en.wikipedia.org/wiki/Finite_element_method (15.04.2010).

AASHTO Information:
"American Association of State Highway and Transportation Officials".
http://www.transportation.org (20.10.2010).

Topologie Optimierungsprogramm im Internet:
TopOpt 3d, Topopt Research Group, Dänemark,
http://www.topopt.dtu.dk/?q=node/11, (Stand 20.04.2010).

Ausschreibungen der zwei Wettbewerbe:
"Concrete Student Trophy 2009", Vereinigung der Österreichischen Zementindustrie,
http://www.zement.at.
"Connectiong Link", Magistrat der Stadt Wien Magistratsabteilung MA 29,
http://www.wien.gv.at/verkehr/brueckenbau/index.html.

Bemerkung: Die Wettbewerbsunterlagen sind inzwischen nicht mehr öffentlich erhältich.

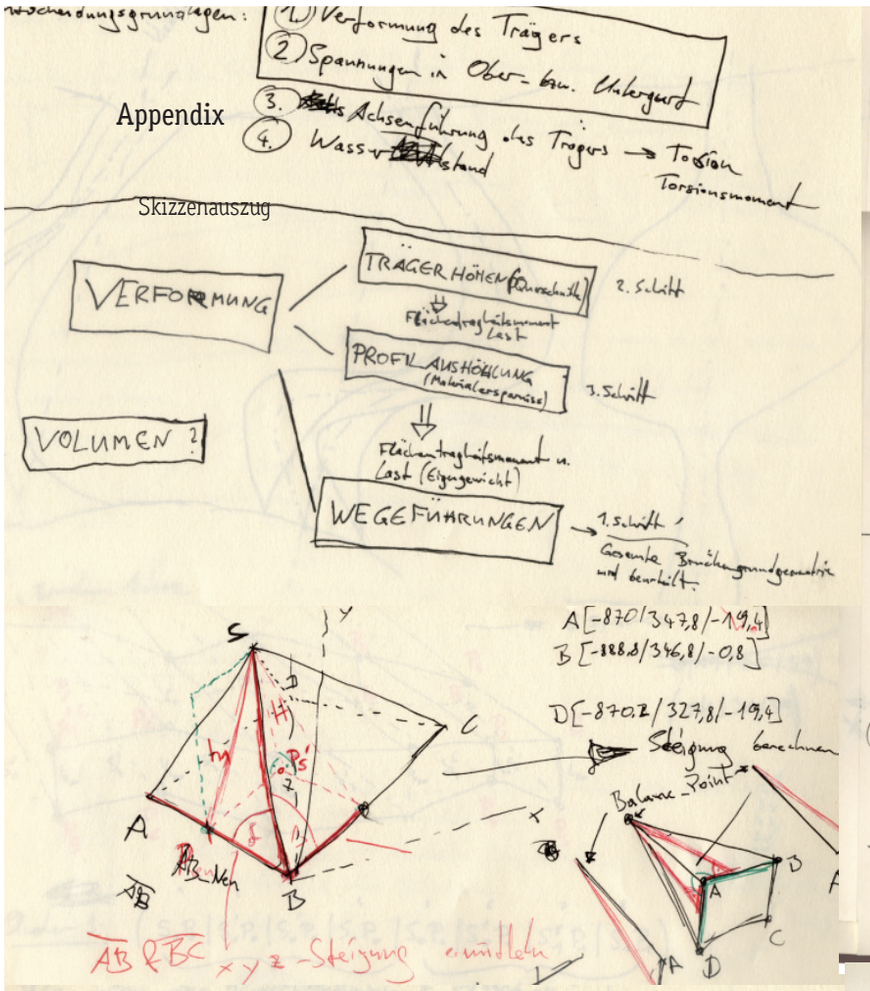
Sonstiges

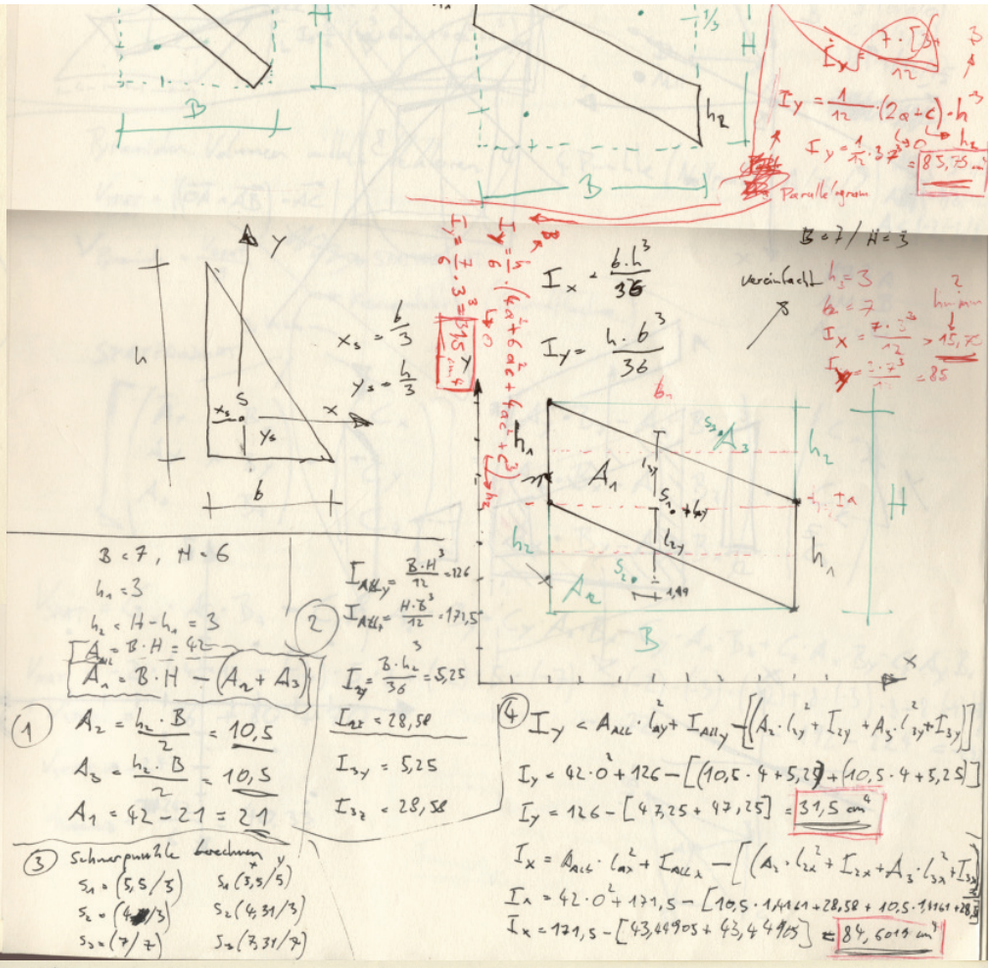
Gespräch mit Softwareentwickler Georg Pircher von ABES Pircher & Partner GmbH am 20.10.2010 in Graz. Georg Picher ist Geschäftsführer und Gesellschafter von der Software Entwickler Firma ABES Pircher & Partner GmbH.

Gespräch mit Dipl. Ing. Florian Graf und Dipl. Ing. Dr. Norbert Seyff, City University of London am 23.02.2010. Bemerkung: Graf und Seyff forschen an Endbenutzerzentrierten Anforderungserhebungsmethoden und entwickelten im Rahmen ihrer Forschung den werkzeugunterstützen iRequire Ansatz.

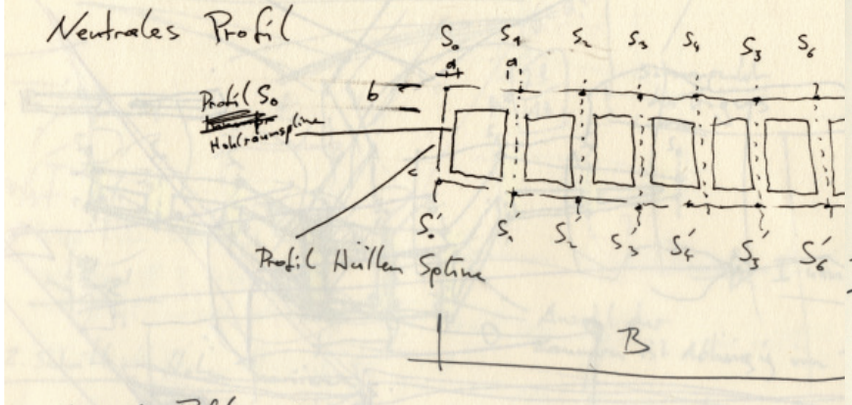
Appendix

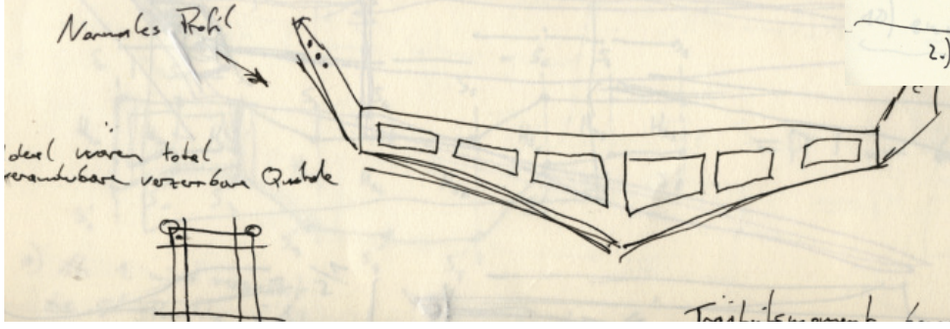
Skizzenauszug



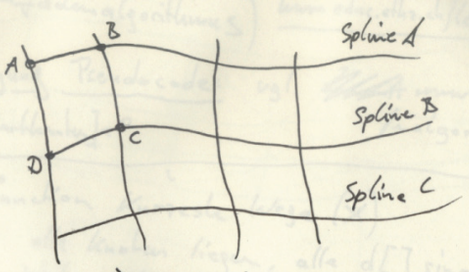


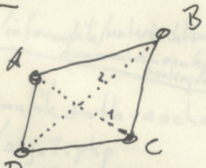
Nach welche Logik wird der Querschnitt variiert und die Wege verschoben?





Fläche der Gesamten Brücken Oberfläche berechnen:





PROBLEM: Welche Diagonale wird für die Dreiecksbildung gewählt.

Die Rechteckfläche wird in 2 Dreiecke zerlegt.

Dreiecksfläche $A = \sqrt{s \cdot (s-a) \cdot (s-b) \cdot (s-c)}$ s = halber Umfang

Berechnung: Δ1: Distanz zwischen $\overline{AB}$, $\overline{BC}$, $\overline{CA}$ berechnen.

1.) $s = \frac{\overline{AB} + \overline{BC} + \overline{CA}}{2}$

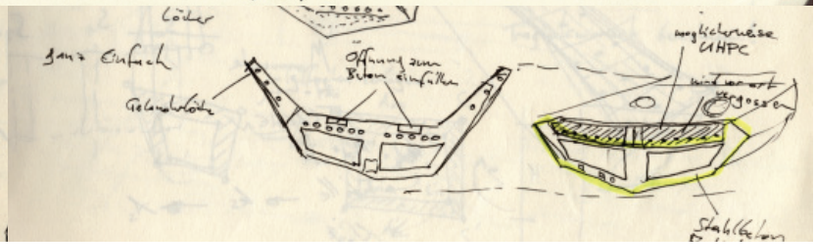
$A = \sqrt{s \cdot (s - AB) \cdot (s - BC) \cdot (s - CA)}$

2.) Δ2: WIEDERHOLEN mit $\overline{AC}$, $\overline{CD}$, $\overline{DA}$

Nach einigen Versuchen Oberflächen mit Cinema 4D zu TRIANGULIEREN, wurde die Diagonale $\overline{AC}$ gewählt

3) Dreiecke addieren

FIN



Konzept/Ideen

Umstellung? Max

Bemerkung:

Die Positionen der Schwerpunkte könnten Auswirkungen auf die Überschlagsformel haben. z.B.: Desto mehr diese in Grundriss von der Idealfläche abweichen, desto schlechter wird $h = L/x$ → $L/20$ → $L/10$ → da Torsion. Desto mehr in die Ansicht (Bogen) desto besser

Berechnung der Durchschnittlichen x & y Entfernungen vom Schwerpunkt.

negative/positive extra Höhe

neues Rechteck.

1.) Strecken $\overline{12}, \overline{23}, \overline{34}, \overline{45}, \overline{56}$ als Platten/Scheiben auf deren Tragfähigkeit beurteilen.

2.) Die Summe der Tragfähigkeit der (5) Einzelplatten/Scheiben muss größer sein als die Mindesttragfähigkeit.

3.) Beurteilung der Einzelnen Elemente

sehr große Tragfähigkeit — keine Tragfähigkeit — große Tragfähigkeit

Problem welche Dimension wird angenommen

Ich hätte gerne → wär dann die Trägerbetrachtung besser geeignet

Trägheitsmomente berechnen? I?

4.) Elemente als Auflager betrachten. Umso Tragfähiger desto ... ein größeres Moment kann auch aufgenommen werden!?

5.) Als Steifen Rahmen betrachten und Momente Linie berechnen. → Dimensionierung schlussfolgern.

Tragfähigkeit = Höhe/Breite → desto größer der Wert desto besser ?

[Stützung<Länge!?]

als steifen Rahmen betrachten

Tragwerkskonzepts-Überlegungen:

Gegeben: Gekrümmte Fläche erzeugt durch Splines.

1.) Man könnte die einzelnen Splines mit ihren Nachbar-Splines vergleichen und die Höhe als Beitrag zur Tragwirkung bewerten. (Scheiben)

2.) Die Fläche wird in Splines zerlegt und jeder Spline wird als Träger betrachtet.

3.) Die gesamte Brücke wird als Träger betrachtet. Widerstandsmoment des Schnittes wird berechnet!

Momentenberechnung

Zu Beginn müssen alle Schnittmomente auf Null gesetzt werden.

HS

Modus 1

ARM 1

Auflagerposition

ARM 2

US

BALANCE-PUNKT

SCHNITTE

6 Splines

Modus 2

Auflagerposition

Schleifen:

ARM 1

Schnitte[c=0; c < Auflagerposition - 1; c++]

{ Teilsummen[a = Auflagerposition - 1; a >= c; a--]

Quader 1: $(S_0P_0 | S_0'P_0 | S_0'P_0 | S_1P_0 | S_0P_1 | S_0'P_1 | S_1'P_1 | S_1P_1)$

A_0 A_1

1.) Schwerpunkt berechnen Σ aller Punktkoordinaten / Punktanzahl (8)

2.) Pyramidenvolumen berechnen → h - mit Volkberechnung.

Was ist wenn ich gerne einen Breiten Rechtecksquerschnitt hätte als nur zwischen # 2 Splines?

Wie funktioniert das, wenn das Modell noch Lofthar sein sollte? = Punkte anzahl gleich sein sollte

Für den Hohlen-Querschnitt gibt es pro Segment Splineanzahl - 1 Lofthorden

Es müsste eigentlich 2 innen Splines geben.

1.) Spline Oben

2.) Spline Unten

Typ 1 Typ 2 Typ 3

Problem

Diese 7 gekrümmten Splines ergeben kein Boolvolumen mehr. → Negativ-Raum

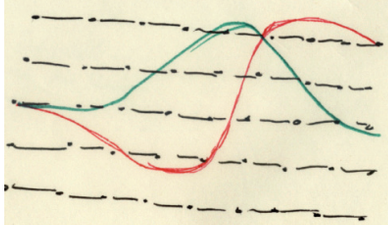
2. Ansatz: Typ 1: Typ 2 Typ 3 Typ 4

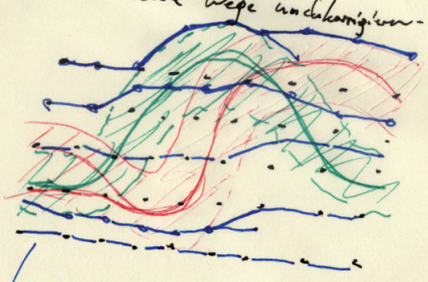
Jetzt muss vor jeder Berechnung festgestellt werden, welcher Typ zu Berechnen ist.

Stepsize

Trägheitsmomente zerlegen

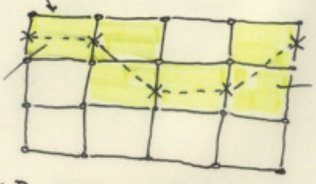
→ Hilfsprobleme → Prioritäten → andere Wege nachkorrigieren





Es wird ein neues möglichst gleichmäßiges Netz (Matrix) erzeugt. Wobei sich keine Matrixsplines mit einer anderen überlappen darf.

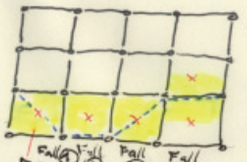
Matrix



Pfad

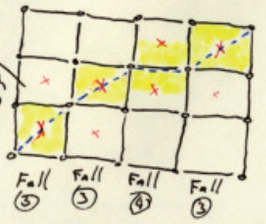
Fläche mit Netzlast

Problem: Der Pfad verläuft in Wirklichkeit nicht in der Mitte der Kanten sondern genau entlang der Matrixkanten.

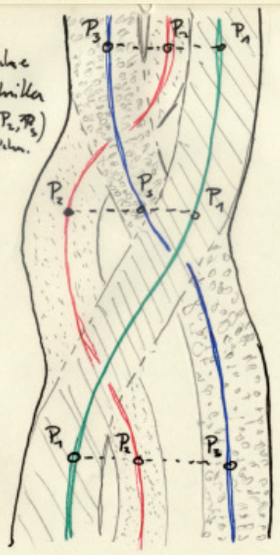


betroffene Flächenschwerpunkte

Problem: Die originale Matrix ist nie so schön gleichmäßig wie hier angenommen. Um die Netzlast zu messen zu können sollte zuerst im Wegbereich eine brauchbare Matrix erzeugt werden.



die Berechnung der Grundfläche muss die Brücke mehreren Stellen durchschneiden und die Punkte (P_a, P_b, P_c) können dann miteinander verglichen und festgestellt welche davon den Rand bilden.



— barrierefreier Weg
Priorität = hoch

— Radfahrweg
Priorität = mittel

— Fußgängerweg
Priorität = niedrig

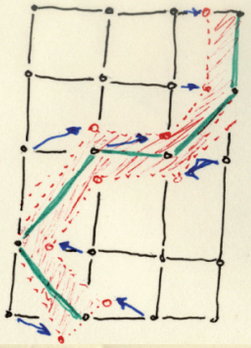
→ Umrisslinie der Grundfläche
Basis-Brückengeometrie.
Rand.

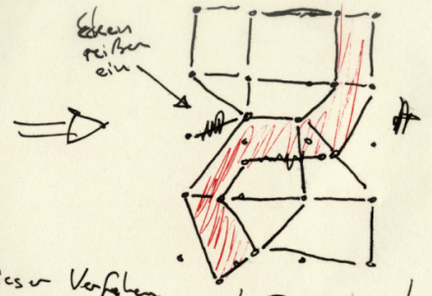
→ zwischen diesen Linien wird ein homogenes Netz (Matrix) erzeugt.

Path_Clearance: Wegkorridor erzeugen.

Im Pfadbereich sollte eine ordentliche Grundmatrix erzeugt werden um die Netzlast einfach zuweisen zu können und um im Wegbereich eine ebene Oberfläche zu erhalten.

1. Strategie: Alle Nachbarpunkte der Wegsplines werden auf die halbe Wegmanddistanz verschoben.



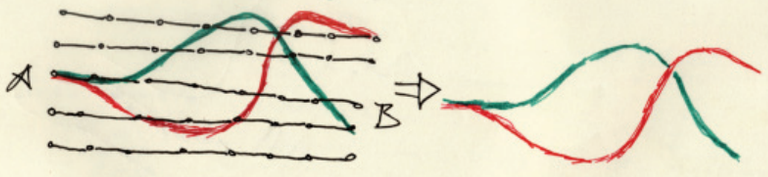


Dieser Verfahren wurde Programmiert es entstanden keppnah Oberflächen. → Verschneidungen

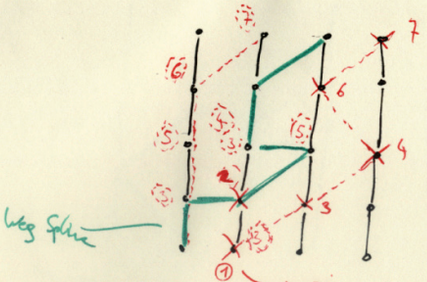
Path Clearance Programm: Definiert Wege- und Geländerzonen

Aufgaben: nimmt Basic Surface und verändert dieses aufgrund der vorhandenen Pfade.

Was muss beim verzerren beachtet werden?



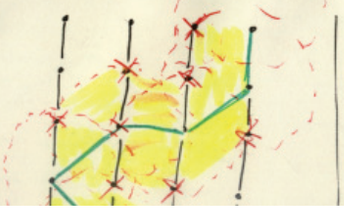
2.) Strategie: Nur Y-Werte der nächsten Nachbarsplive und deren nächsten Punkt zu dem aktuellen Pfadpunkt mit Pfadpunkt Y-Wert gleichsetzen.



Problem dieses Verfahrens ist, dass manchmal sehr breite Wege entstehen können.
→ weil das Netz sehr unregelmäßig sein kann und der nächste Punkt sehr weit entfernt sein kann.

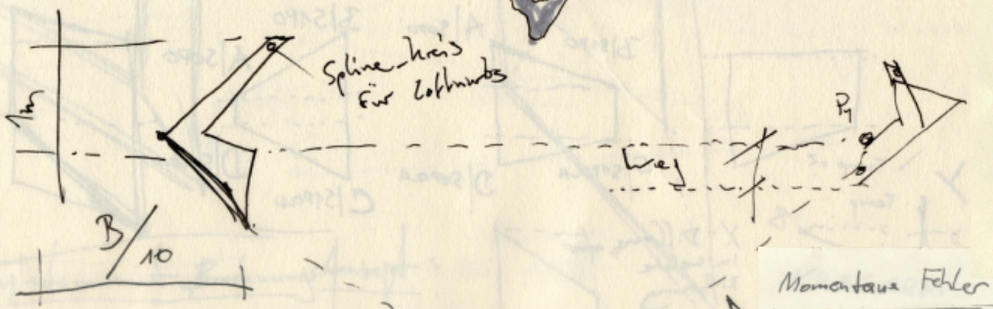
Dieser Punkt müsste gesperrt werden weil sonst im nächsten Schritt wieder auf diesen zugegriffen wird.

3. Strategie: Radius-Wegkorridor checken.
Alle Punkte im Radiusumkreis auf gleichen Y-Wert setzen.

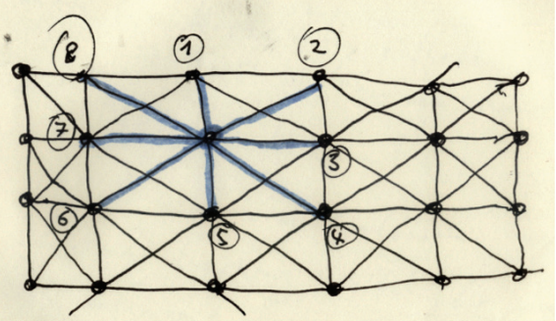


4. Strategie: ermittelte 3. Strategie.
Alle nicht als X oder Pfad markierten Punkte auf nächsten Punkt ausrichten und Y-Wert gleichsetzen.

Wie wird das Geländer erzeugt?

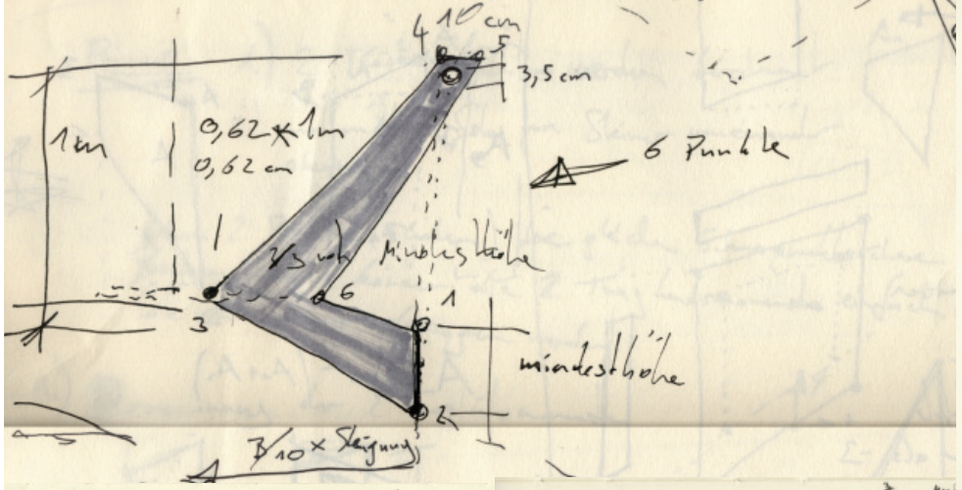


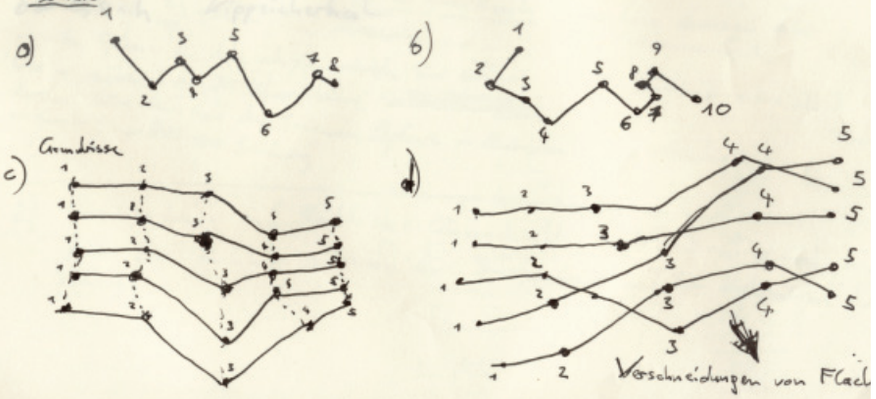
Definition d. Nachbar-Punkte

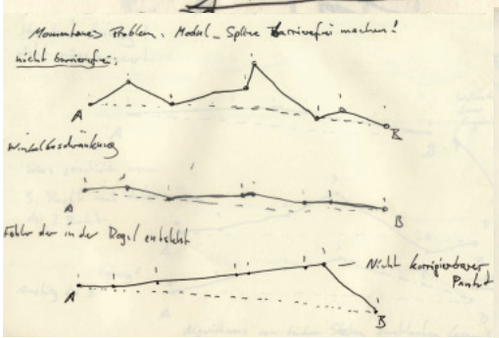


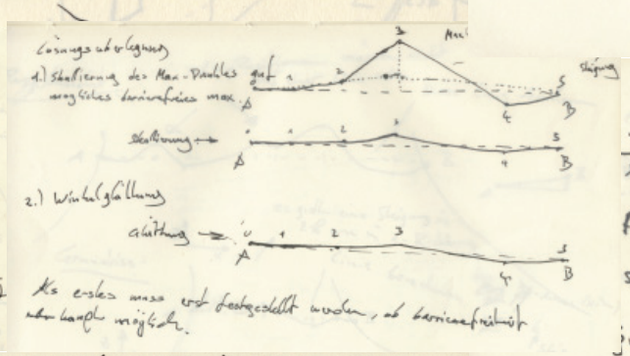
Momentane Fehler oder Ausnahmen ~~Fehler~~ - Querschnitte / Sonderfälle

Je nach Deformer kann es z.B.: zu Überlappungen oder Splines bzw. Verschneidungen der Flächen kommen.

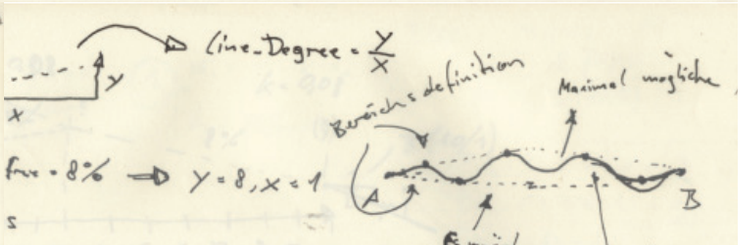




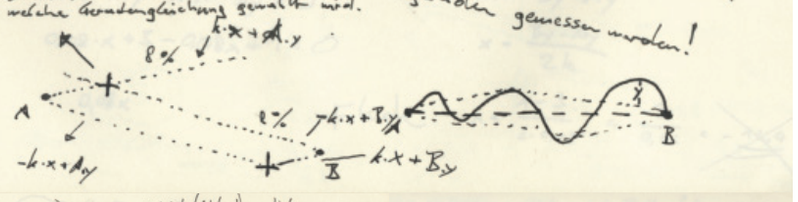




Als erstes muss erst festgestellt werden, ob Kurvenfreiheit überhaupt möglich.



Eigentlich müsste die größte Abweichung von der maximal möglichen Barrierefreiheitsgeraden gemessen werden!



Kürzester Weg über die Brücke → Dijkstra Algorithmus

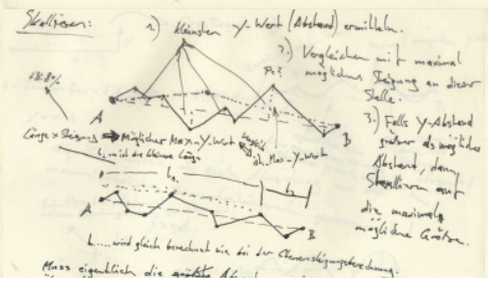
(Bindefadenalgorithmus) www.educ.ethz.ch/lehrpersonen/informatik/unterrichtsmaterialien/kommunikationskryptographie/routing/las3.pdf

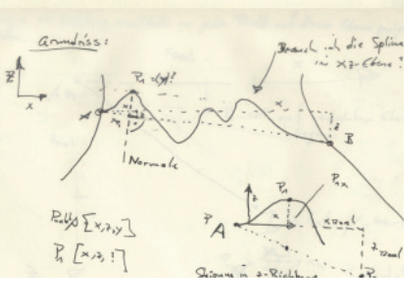
Überlegung Pseudocode: vgl. ~~www.mathe-aachen.de~~ www-i1.informatik.rwth-aachen.de/algorithmus/algo7.php

d[Startknoten] = 0

```
1) function Kürzeste Wege (x)
2)     alle Knoten liegen, alle d[] sind unendlich, nur d[Startknoten] = 0
3)     while es liegende Knoten gibt do
4)         v := der liegende Knoten mit dem kleinsten d[v]
5)         mache v hängend
6)         for all Fäden von v zu einem Nachbarn u der Länge L do
7)             if d[v] + L < d[u] then d[u] = d[v] + L
8)         End for
9)     endwhile
10) end
```

kürzester Weg hin zu v





Entwurfsprozessüberlegungen

Where to begin an neverending cycle?

Theory about brigdes and the basic competion parameters.

The aim of this diploma thesis is to develop a bridge design process with ...

The process is strictly related to an architectural concept the was developt regarding to the competition Conscret Student Trophy 2009.

This is a short graphical overview that will also change during the development of the brigde design process.

The whole process don't has to be done within the use of one software programm. If it is needed any single Module can be outsourced in to another software programm or also into an analog editing. But every change happens under the strict reglementation rules of the BRIDGE_DESIGN_CONCEPT.

GENERATE_BASIC_SURFACE

Description: Generate a basic planar-surface beween two 3d -Splines. For the Surface generation not the whole spline lenght is used. On each spline theres a start- or endingarea defined. This area can be set manualy or via a random function. The two areas are connected by splines that will finaly define the basic surface.

Parameter:	
- SUPPORT SPLINE A	LINK.OBJECT
- SUPPORT SPLINE B	LINK.OBJECT
- RANDOM POSITION	ENABLE_YES/NO
or POSTION A , POSITION B	PERCENT %
- BRIDGE WIDTH	PERCENT %
- NUMBER OF SPLINES	FLOAT-SLIDER
- SPLINE POINTS	Int.NUMBER
- CREATE	Int.NUMBER
	ENABLE_YES/NO

Because it's a concept for a brigde some basic conditions are needet. A bridge always connects two sides. This is why a Start- and Endingarea is set and can't be changed anymore.

INITIAL

PROCESS_THERORY

CHANGING IN TIME

Theory about digital inspitation.

ART_CONCEPT_DEFORMATION

CLOTH-DEFORMING

Description: ...

Parameter: -.... | ...

SURFACE-CREATOR

Description: ...

Parameter: -.... | ...

PHYSICS_EXPERIMENTS__DEFORMING

Description: ...

Parameter: -.... | ...

Theory about creating surfaces and studies about how to make modells and constructions of them.

FORMULA_DEFORMER

Description: ...

Parameter: -.... | ...

SKETCH_INTERPRETER_DEFORMER

Description: ...

Parameter: -.... | ...

Deforming the surface after a concept. This can happen manualy or through specific deformer functions.

PROBLEMS:
INFLUCENES AND PRIORITIES
Where to begin?

MANUALY_DEFORM

Description: ...

Parameter: -.... | ...

Direct analoge influences

LIMITATIONS_EVALUATION_DEFORMATION

Theory about form and construction under influence of forces.

FORCE_MATERIAL_DEFORMING

Description: ...

Parameter: -.... | ...

BARRIER_FREE_DEFORMING

Description: ...

Parameter: -.... | ...

OTHER_LIMITATIONS

Description: ...

Parameter: -.... | ...

Therory about evaluating froms and deforming them.

MOVING_BRIDGE_DEFORMING

Description: ...

Parameter: -.... | ...

Concept created: 23.04.2009
Version 1.0
23.04.2009

The **scripted bridge** is an example project to show how the computer can be intergrated in the architectural drafting process.

The basic conditions and the architectural concept are regarding to the competition `**Concret Student Trophy 2009**`.

Usually bridge_shapes are mainly determined by the forces that apear because of the loads and the choosen construction_type. Design a complex shape durring thedeveloping process by thinking forces...from ...lose complex form, nonstandart structure, design in forces , complexity show the process in the architecture...time, transformation...

Strictly refering to the **bridge_design_concept** a dynamic **bridge_design_process** is developted. The process is defind by different modules that interact with eachother and determine the final output design. The moduls can be understood as different stages in the **bridge_design_process** which are considering different priorities and are influenced by diffent input parameters. The changing of this priorities and parameters finaly lead to varieties of more or less similar bridges.

The whole process don't has to be done within the use of one software programm. If it is needed any single module can be outsourced in to another digital or analoge procedure.

The focus this theses is the design_process. To document this process this a short graphical overview that will also change during the development of the brigde design process.

Theory about brigdes and the basic competion parameters.

Where to begin an neverending cycle?

GENERATE_BASIC_SURFACE

Description: Generate a basic planar-surface beween two 3d -Splines. For the Surface generation not the whole spline lenght is used. On each spline theres a start- or endingarea defined. This area can be set manualy or via a random function. The two areas are connected by splines that will finaly define the basic surface.

INPUT	OUTPUT
- SUPPORT SPLINE A - SUPPORT SPLINE B - RANDOM POSITION or POSTION A , POSITION B - BRIDGE WIDTH - NUMBER OF SPLINES - SPLINE POINTS - CREATE	LOFTNURBSOBJECT - SPLINES -

Note: The oringin-points of the input splines have to be equal to the origin point of the world coordinates point zero.

Because it's a concept for a brigde some basic conditions are needet. A bridge always connects two sides. This is why a **Start-** and **Endingarea** is set and can't be changed anymore.

INITIAL

PROCESS_THERORY

CHANGING IN TIME

ART_CONCEPT_DEFORMATION

Theory about digital inspitation.

CLOTH-DEFORMING

Description: ...

Parameter: -.... | ...

PHYSICS_EXPERIMENTS__DEFORMING

Description: ...

Parameter: -.... | ...

SURFACE-CREATOR

Description: ...

Parameter: -.... | ...

Theory about creating surfaces and studies about how to make modells and constructions of them.

FORMULA_DEFORMER

Description: ...

Parameter: -.... | ...

SKETCH_INTERPRETER_DEFORMER

Description: ...

Parameter: -.... | ...

Direct analoge influences

MANUALY_DEFORM

Description: ...

Parameter: -.... | ...

Deforming the surface after a concept. This can happen manualy or through specific deformer functions.

PROBLEMS:
INFLUCENES AND PRIORITIES
Where to begin?

To start a complex process fist the modules are developted and checked one their own. The MAKE_SPLINE_BARROER_FREE.Modul is in no direct connection to any other Modul.

To check the single modules on complex distort surfaces the spline points are moved manually. To save time an automatic spline deformer should be created next.

LIMITATIONS_EVALUATION_DEFORMATION

Theory about form and construction under influence of forces.

FORCE_MATERIAL_DEFORMING

Description: ...

Parameter: -.... | ...

MAKE_SPLINE_BARRIER_FREE

Description: Compares the degree between the points of a single Spline. If the degree is bigger than input value [percent] the curve is changed until it's barrier-free.

INPUT	OUTPUT
- STEIGUNG - SPLINENAME - CHECK ON/OFF	SPLINE -

Note: Temporary the SPLINENAME is ignored and all splines are checked and evantually changed.

OTHER_LIMITATIONS

Description: ...

Parameter: -.... | ...

Therory about evaluating froms and deforming them.

MOVING_BRIDGE_DEFORMING

Description: ...

Parameter: -.... | ...

Concept created: 23.04.2009
Version 2.0
14.05.2009

The **scripted bridge** is an example project to show how the computer can be intergrated in the architectural drafting process.

The basic conditions and the architectural concept are regarding to the competition '**Concret Student Trophy 2009**'. Usually bridge_shapes are mainly determined by the forces that apear because of the loads, the choosen construction_type and the material that it is made of. The basic concept is to design a freeform bridge within the permanent thinking of forces which are apearing.

The main shape of the bridge is defined by 3 differnt paths that are generatet from one bank to the other. These 3 paths are are related to eachother via their load bearing capacity.

Strictly refering to the **bridge_design_concept** a dynamic **bridge_design_process** is developted. The process is defind by different modules that interact with eachother and determine the final output design. The moduls can be understood as different stages in the **bridge_design_process** which are considering different priorities and are influenced by diffent input parameters. The changing of this priorities and parameters finaly lead to varieties of more or less similar bridges.

The focus this theses is the design_process. To document this process this a short graphical overview that will also change during the development of the brigde design process.

BASIC_SETTINGS

Description: Setting basic conditions that influcene the bridge shape.

OUTPUT

- AUFLAGER A
- AUFLAGER B
- SPLINES
- SPLINE_PUNKTE
- BRÜCKENBREITE
- DEFORMFAKTOR
- BARRIEREFREIHEIT
- PATHDEGREE_FACTOR -
- FUßGÄNGER BREITE -
- RADFAHRER_BREITE -
- ROLLSTUHLFAHRER_BREITE -
- EIGENGEWICHT-
- NUTZLAST-

Theory about bridges and the basoc competition parameters

Because it's a concept for a brigde some basic conditions are needet. A bridge always connects two sides. This is why a **Start-** and **Endingarea** is set and can't be changed anymore.

INITIAL

PROCESS_THERORY

CHANGING IN TIME

GENERATE_BASIC_SURFACE

Description: Generate a basic planar-surface beween two 3d -Splines. For the Surface generation not the whole spline lenght is used. On each spline theres a start- or endingarea defined. This area can be set manualy or via a random function. The two areas are connected by splines that will finaly define the basic surface.

INPUT	OUTPUT
- SUPPORT SPLINE A - SUPPORT SPLINE B - RANDOM POSITION or POSTION A , POSITION B - BRIDGE WIDTH - NUMBER OF SPLINES - SPLINE POINTS - CREATE	LOFTNURBSOBJECT - SPLINES -

Note: The oringin-points of the input splines have to be equal to the origin point of the world coordinates point zero.

Theory about form and construction under influence of forces.

LIMITATIONS_EVALUATION_DEFORMATION

STUCTURE_EVALUATING

Description: Defines a design_space for the current geometry and evaluates the divergence between ideal_design_space and the current_geometry.

INPUT	OUTPUT
- SPLINES - STRUCTURE_LOAD - SERVICE_LOAD - PROPORTION_FACTOR - CHECK ON/OFF	SPLINES - STRUCTURECALC_OBJECT- SINGLE_BALANCE_SPHERES- GEOMETRY_BALANCE_POINT- BRIDGE_SUFACE_AREA- BRIDGE_WEIGHT- SECTION_TORQUE-

Note: To calculate the maximum torque and der temporary torque in different bridge sections a load has to be set.

Theory about evaluating forms and deforming them.

MAKE_SPLINE_BARRIER_FREE

Description: Compares the degree between the points of a single Spline. If the degree is bigger than input value [percent] the curve is changed until it's barrier-free.

INPUT	OUTPUT
- STEIGUNG - SPLINENAME - CHECK ON/OFF	PATH_SPLINE_0 -

Note: Temporary the SPLINENAME is ignored and all splines are checked and eventually changed.

PATH_CLEARANCE

Description:

INPUT	OUTPUT
- SPLINENAMES - PEDESTRIAL_WIDTH - BIKER_WIDTH - WHEELCHAIR_WIDTH - CHECK ON/OFF	PATH_SPLINE_0 -

Note: ...

MOVING_BRIDGE_DEFORMING

Description: ...

Parameter: -.... | ...

Deforming the surface after a concept. This can happen manualy or through specific deformer functions.

PROBLEMS:

Can the GENERATE_BASIC_SURFACE Modul be replaced by the PATH Gernerating Moduls?

NO!

It's easier to keep GENERATE_BASIC_SURFACE Modul to generate a easy handable matrix. The basic surface has to be seen as PLASTICINE or as free formable modelling clay.

The main resolution of the bridge can be set by the basic surface.

The PATH_CLEARANCE_MODULE kneats the basic surface to the main bridge shape with is determined by the 3 different paths.

BRIDGE_OBJECT

PATH_OBJECTS

ART_CONCEPT_DEFORMATION

SIMPLE_SPLINE_DEFORMER

Description: The spline_points are distorred by a cosinus_funktion. The frequency and amplitude is choosen by coincidence. With a global DISTORSION_FACTOR the distorsion strenght can be controlled.

INPUT	OUTPUT
- SPLINES - DISTORSION_FACTOR - CHECK ON/OFF	SPLINES -

Note: The beginning- and the endpoints of the splines are not distorted.

Theory about digital inspitation.

PATH_FINDER

Description: This module generates 3 Spline_Paths over the bridge surface. Through the GRAD_FACTOR it can be defined how much the changing of surface_high matters to the path calculation. The script is based on the Dijkstra_Algorithmus.

INPUT	OUTPUT
- SPLINES - GRADE_FACTOR - CHECK ON/OFF	PATH_SPLINE_0 - PATH_SPLINE_1 - PATH_SPLINE_2 -

Note: If the bridge is extremly distort, sometimes an error can apear (endless loop in the module). It has to do with the comparision value (Smallest_Distance) at the beginnung of the program.

PATH_OBJECT_GENERATOR

Description: Creates 1 Path Spline_Object from one side to another.

INPUT	OUTPUT
- SPLINE POINTS - CREATE	PATH_SPLINE_0 -

MANUALLY_DEFORM

Description: All spline_points can be moved manually at every time by selecting points via MOUSE.CLICK and transforming the analog mouse position into digital point coordinates.

INPUT	OUTPUT
- MOUSE.CLICK - MOUSE.POSITION	SPLINE.POINTS -

SURFACE-CREATOR

Description: ...

Parameter: -.... | ...

Direct/Indirect Analoge Influences

ATTRACTOR_PATH_GENERATOR

Description: Creates 1 Path Spline from one side to another. This Spline is distorted by the ATTRAKCOR_OBJECTS in the scene.

INPUT	OUTPUT
- SPLINE POINTS - ATTRACTOR_NAMES - CREATE	PATH_SPLINE_1 -

Theory about creating surfaces and studies about how to make modells and constructions of them.

Started: 23.04.2009
by Christian Pichlkastner
Version 6.0
07.07.2009

The **scripted bridge** is an example project to show how the computer can be intergrated in the architectural drafting process.

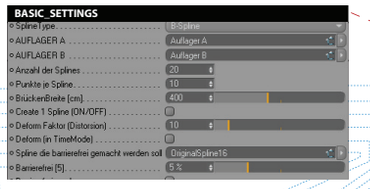
The basic conditions and the architectural concept are regarding to the competition `**Concret Student Trophy 2009**´. Usually bridge_shapes are mainly determined by the forces that apear because of the loads, the choosen construction_type and the material that it is made of. The basic concept is to design a freeform bridge within the permanent thinking of forces which are apearing.

The main shape of the bridge is defined by 3 differnt paths that are generatet from one bank to the other. These 3 paths are are related to eachother via their load bearing capacity.

Strictly refering to the **bridge_design_concept** a dynamic **bridge_design_process** is developted. The process is defind by different modules that interact with eachother and determine the final output design. The moduls can be understood as different stages in the **bridge_design_process** which are considering different priorities and are influenced by diffent input parameters. The changing of this priorities and parameters finaly lead to varieties of more or less similar bridges.

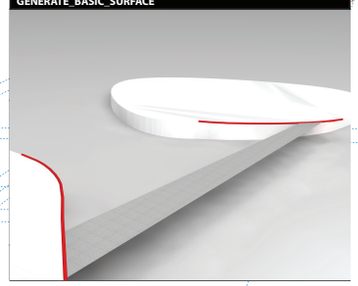
The focus this theses is the design_process. To document this process this a short graphical overview that will also change during the development of the brigde design process.

BASIC_SETTINGS



Theory about bridges and the basoc competition parameters

GENERATE_BASIC_SURFACE



Because it's a concept for a brigde some basic conditions are needet. A bridge always connects two sides. This is why a **Start-** and **Endingarea** is set and can't be changed anymore.

INITIAL

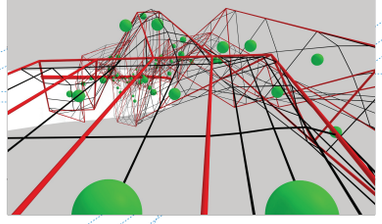
PROCESS_THERORY

CHANGING IN TIME

Theory about form and construction under influence of forces.

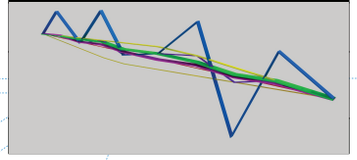
LIMITATIONS_EVALUATION_DEFORMATION

STUCTURE_EVALUATING

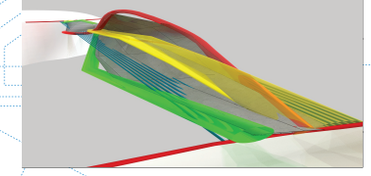


Therory about evaluating forms and deforming them.

MAKE_SPLINE_BARRIER_FREE



PATH_CLEARANCE



MOVING_BRIDGE_DEFORMING

Deforming the surface after a concept. This can happen manualy or through specific deformer functions.

PROBLEMS:

Can the GENERATE_BASIC_SURFACE Modul be replaced by the PATH Gernerating Moduls?

NO!

It's easier to keep GENERATE_BASIC_SURFACE Modul to generate a easy handable matrix. The basic surface has to be seen as PLASTICINE or as free formable modelling clay.

The main resolution of the bridge can be set by the basic surface.

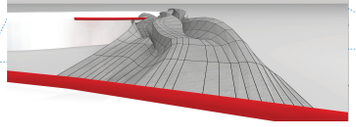
The PATH_CLEARANCE_MODULE kneats the basic surface to the main bridge shape with is determined by the 3 different paths.

ART_CONCEPT_DEFORMATION

BRIDGE_OBJECT

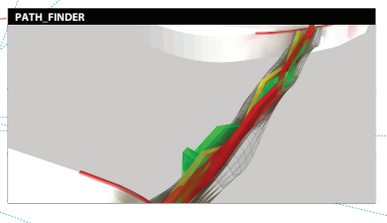
PATH_OBJECTS

SIMPLE_SPLINE_DEFORMER



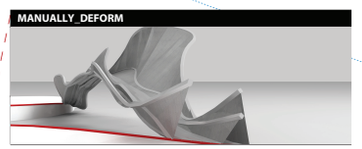
Theory about digital inspitation.

PATH_FINDER

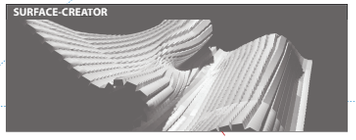


PATH_OBJECT_GENERATOR

MANUALLY_DEFORM



SURFACE-CREATOR



Direct/Indirect Analoge Influences

ATTRACTOR_PATH_GENERATOR

Theory about creating surfaces and studies about how to make modells and constructions of them.

Started: 23.04.2009
by Christian Pichlkastner
Version 6.0
07.07.2009

"Scripted Bridge" is an example project that shows how the computer can be integrated in the architectural drafting process. With an animation software a bridge designing process using scripting- and mashup-tools" was developed. With this process variations of the basic bridge concept can be generated. The basic conditions are regarding to the competitions "Concrete Student Trophy 2009" and "Connecting Link" which request a moveable bridge for pedestrians and bikers. Referring to current tendencies in architecture it was proofed how these digital tools could support the pre-designing process of bridge-constructions which deviate from standard shapes. In this context it is also questioned how material and construction decisions could be integrated. Due to the point of view that bridge designs are not only determined by problems that come out of dealing with forces an open process was created that allows to include other criteria.

The main shape of the bridge is defined by 3 differnt paths that are generatet from one bank to the other. These 3 paths are are related to eachother via their load bearing capacity.

Strictly refering to the bridge_design_concept a dynamic bridge_design_process is developted. The process is defind by different modules that interact with eachother and determine the final output design. The moduls can be understood as different stages in the bridge_design_process which are considering different priorities and are influenced by diffent input parameters. The changing of this priorities and parameters finaly lead to varieties of more or less similar bridges.

The focus this theses is the design_process. This is a short graphical overview which documents the developing process and changed during the development of the bridge design process.

BASIC_SETTINGS

Description: Setting basic conditions that influcene the bridge shape.

OUTPUT

- NUMBER OF ATTRACTOR SPHERES
- CREATE ATTRAKTOR RANDOM MODE ON/OFF
- MAKE ATTRACTOR SPHERES
- KNEAD PATH BETWEEN ATTRACTORS
- ATTRACTED (PATH)-SPLINE
- SPLINE TYPE
- SUPPORT SPLINE A
- SUPPORT SPLINE B
- BRIDGE WIDTH
- NUMBER OF SPLINES
- SPLINE POINTS
- CREATE SURFACE SPLINES (ON/OFF)
- PEDESTRIAN PATH POINTS
- CREATE PEDESTRIAN PATH
- BARRIER FREE PATH POINTS
- CREAT BARRIER FREE PATH
- PATHFINDER HEIGHT FACTOR
- BIKER PATH (PATHFINDER)
- WATERLEVEL
- BRIDGE MATERIAL
- MATERIAL WEIGHT [kg/m³]
- SERVICE LOAD [kN/m²]
- DIAGRAM FACTOR (TORQUEHEIGHT /x)
- SIGMA_ZULÄSSIG [kN/cm²]
- STATIC SYSTEM
- SUPPORT POSITION
- EVALUATE STRUCTURE
- PEDESTRIAN PATH WIDTH
- BIKER PATH WIDTH
- WHEELCHAIR WIDTH
- DEFORM FACTOR (DISTORSION)
- DEFORM MODE ON
- BARRIER FREE PATH DEGREE
- MAKE BARRIER FREE
- SCALE FACTOR (PATH OVERLAPING)
- KNEAD BASIC SURFACE to 2 PATHS
- PATH CLEARANCE
- CREATE HANDRAIL
- NUMBER OF CALCULATIONS

GENERATE_BASIC_SURFACE

Description: Generate a basic planar-surface beween two 3d -Splines. For the Surface generation not the whole spline lenght is used. On each spline theres a start- or endingarea defined. This area can be set manualy or via a random function. The two areas are connected by splines that will finaly define the basic surface.

INPUT	OUTPUT
- SPLINE TYPE - SUPPORT SPLINE A - SUPPORT SPLINE B - BRIDGE WIDTH - NUMBER OF SPLINES - SPLINE POINTS - CREATE SURFACE SPLINES	LOFTNURBS_OBJECT - BRIDGE_SURFACE_SPLINES -

Note: The oringin-points of the input splines have to be equal to the origin point of the world coordinates point zero.

MAKE_SPLINE_BARRIER_FREE

Description: Compares the degree between the points of a single Spline. If the degree is bigger than input value [percent] the curve is changed until it's barrier-free.

INPUT	OUTPUT
- BARRIER FREE PATH DEGREE - SPLINE_PATH_NAME - MAKE BARRIER FREE	SPLINES_CONTROLLPOINTS -

Note: The SPLINE_PATH_NAME is defined in the script.

SIMPLE_SPLINE_DEFORMER

Description: The spline_points are distorred by a cosinus_funktion. The frequency and amplitude is choosen by coincidence. With a global DISTORSION_FACTOR the distorsion strenght can be controlled.

INPUT	
- NUMBER OF SPLINES - SPLINE POINTS - DISTORSION_FACTOR - DEFORM MODE ON	BRIDGE_SURFACE_SPLINES -

Note: The beginning- and the endpoints of the splines are not distorted.

MAKE_ATTRACTOR_SPHERES

Description: Creates a specific number of spheres with two different basic names ("NEGATIV_", „POSITIV_"). If the random mode is switched on, the spheres are placed in variable positions and get different radiuses.

INPUT	OUTPUT
- NUMBER OF ATTRACTOR SPHERES - CREATE ATTRAKTOR RANDOM MODE ON/OFF - MAKE ATTRACTOR SPHERES	SPHERE_OBJECTS -

SCENE_OBJECTS

BRIDGE_OBJECT

PATH_OBJECTS

MAKE_PEDESTRIAN_PATH

Description: Creates 1 Path Spline_Object from one side to another.

INPUT	OUTPUT
- NUMBER OF SPLINES - SPLINE POINTS - PEDESTRIAN PATH POINTS - CREATE PEDESTRIAN PATH	PATH_SPLINE_1 -

KNEAD_PATHS_TO_ATTRACTORS

Description: Kneads the selected spline (Path) depending on the positions, names and radiuses of Attractor-Spheres. It attracts the Spline-Controllpoints like magnets.

INPUT	OUTPUT
- KNEAD PATH BETWEEN ATTRACTORS - ATTRACTED (PATH)-SPLINE - ATTRACTOR_SPHERES_DATA	SPLINE_CONTROLLPOINTS -

MAKE_BARRIER_FREE_PATH

Description: Creates 1 Path Spline_Object from one side to another.

INPUT	OUTPUT
- NUMBER OF SPLINES - SPLINE POINTS - PEDESTRIAN PATH POINTS - CREAT BARRIER FREE PATH	PATH_SPLINE_2 -

PATH_CLEARANCE

Description: This modul ensures that the bridge is useable for different users. The bridgesurface is flattened in the different pathclearance areas.

INPUT	OUTPUT
- NUMBER OF SPLINES - SPLINE POINTS - PEDESTRIAL_WIDTH - BIKER_WIDTH - WHEELCHAIR_WIDTH - PATH CLEARANCE	BAISC_SURFACE_POINTS -

PATH_FINDER

Description: This module generates a Spline_Path over the bridge surface. Through the GRAD_FACTOR it can be defined how much the changing of surface_high matters to the path calculation. The script is based on the Dijkstra_Algorithmus.

INPUT	OUTPUT
- NUMBER OF SPLINES - SPLINE POINTS - PATHFINDER HEIGHT FACTOR - BIKER PATH (PATHFINDER)	PATH_SPLINE_0 - PATH_SPLINE_1 - PATH_SPLINE_2 -

Note: If the bridge is extremly distort, sometimes an error can apear (endless loop in the module). It has to do with the comparision value (Smallest_Distance) at the beginnung of the program.

STUCTURE_EVALUATING

Description: Creates a volume proposal for the current surface. A lot of 3d objects that visualize the calculation data are created. To get less weight for the volume the structure is holed after a certain concept that considers material and construction.

INPUT	OUTPUT
- NUMBER OF SPLINES - SPLINE POINTS - WATERLEVEL - BRIDGE MATERIAL - MATERIAL WEIGHT - SERVICE LOAD - DIAGRAM FACTOR - SIGMA_ZULÄSSIG - STATIC SYSTEM - SUPPORT POSITION - EVALUATE STRUCTURE - BEAM NUMBER - PEDESTRIAN PATH WIDTH - BIKER PATH WIDTH - WHEELCHAIR WIDTH - DEFORM FACTOR (DISTORSION) - BARRIER FREE PATH DEGREE - SCALE FACTOR (PATH OVERLAPING)	SECTION_VOLUME_SPLINES - LOFTNURBS_OBJECT - BRIDGE_SURFACE_SPLINES - SINGLE_BALANCE_SPHERES- GEOMETRY_BALANCE_POINT- BRIDGE_SUFACE_AREA- BRIDGE_WEIGHT- STRUCTUR-TENSIONS- STRUCTUR_BENDING- SECTION_TORQUE_OBJECTS- DEFORM MODE ON - MAKE BARRIER FREE- KNEAD PATH BETWEEN ATTRACTORS - KNEAD BASIC SURFACE to 2 PATHS - PATH CLEARANCE - CREATE HANDRAIL -

ITERATOR

Description: Starts the calculation x-times.

INPUT	OUTPUT
- NUMBER OF CALCULATIONS	BEAM_NUMBER -

MANUALLY_DEFORM

Description: All spline_points can be moved manually at every time by selecting points via MOUSE.CLICK and transforming the analog mouse position into digital point coordinates.

INPUT	OUTPUT
- 3d_OBJECTS_DATA	3D_OBJECTS_DATA -

KNEAD_TO_PATHS

Description: Kneads the basic surface to two paths.

INPUT	OUTPUT
- NUMBER OF SPLINES - SPLINE POINTS - PEDESTRIAL_WIDTH - BIKER_WIDTH - WHEELCHAIR_WIDTH - SCALE FACTOR (PATH OVERLAPING) - KNEAD BASIC SURFACE to 2 PATHS	BAISC_SURFACE_POINTS -

CREATE_HANDRAIL

Description: Creates 1 Path Spline from one side to another. This Spline is distorted by the ATTRAKCOR_OBJECTS in the scene.

INPUT	OUTPUT
- NUMBER OF SPLINES - SPLINE POINTS - PEDESTRIAN PATH WIDTH - BIKER PATH WIDTH - WHEELCHAIR WIDTH - CREATE HANDRAIL	HANRAIL_OBJECTS -

Because it's a concept for a bridge some basic conditions are needet. A bridge always connects two sides. This is why a **Start-** and **Endingarea** is set and can't be changed anymore.

INITIAL

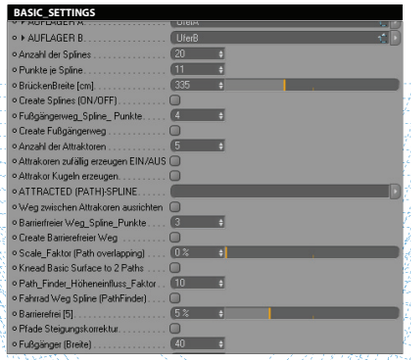
PROCESS_THERORY

CHANGING IN TIME

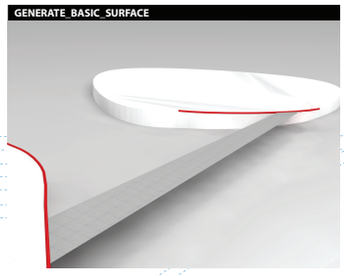
Deforming the surface after a concept. This can happen manualy or through specific deformer functions.

Started: 23.04.2009
by Christian Pichlkastner
Version 9.0
03.02.2010

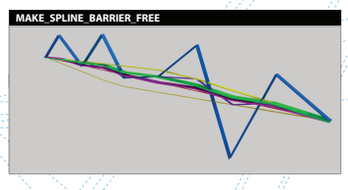
BASIC_SETTINGS



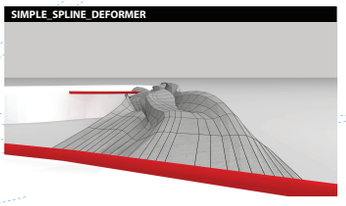
GENERATE_BASIC_SURFACE



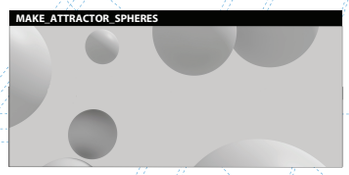
MAKE_SPLINE_BARRIER_FREE

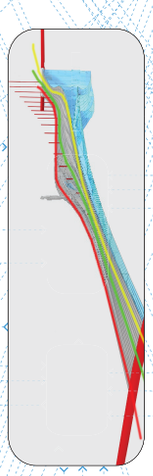


SIMPLE_SPLINE_DEFORMER

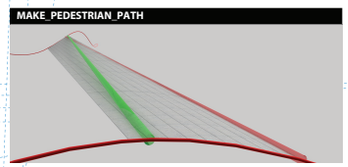


MAKE_ATTRACTOR_SPHERES

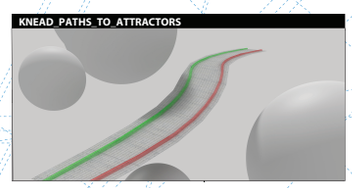




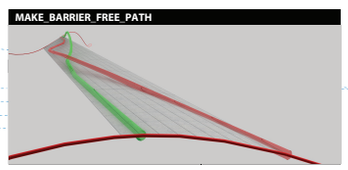
MAKE_PEDESTRIAN_PATH



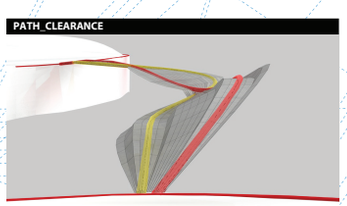
KNEAD_PATHS_TO_ATTRACTORS



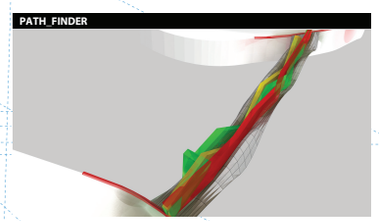
MAKE_BARRIER_FREE_PATH



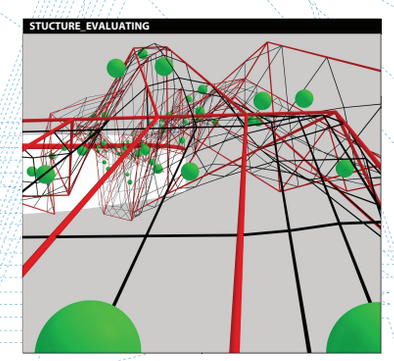
PATH_CLEARANCE



PATH_FINDER



STUCTURE_EVALUATING

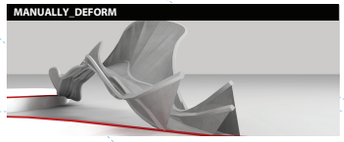


ITERATOR

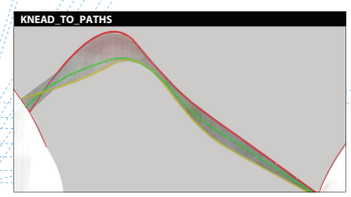
Description: Starts the calculation x-times.

INPUT	OUTPUT
- NUMBER OF CALCULATIONS	BEAM_NUMBER -

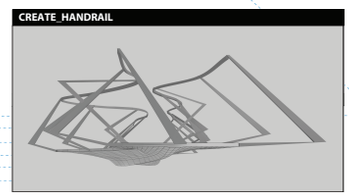
MANUALLY_DEFORM



KNEAD_TO_PATHS



CREATE_HANDRAIL



Because it's a concept for a bridge some basic conditions are needet. A bridge always connects two sides. This is why a **Start-** and **Endingarea** is set and can't be changed anymore.

INITIAL

PROCESS_THERORY

CHANGING IN TIME

Deforming the surface after a concept. This can happen manualy or through specific deformer functions.

Started: 23.04.2009
by Christian Pichlkastner
Version 9.0
03.02.2010

Quellcode

```
//Zu Beginn werden die am häufigsten verwendeten Unterprogramme
//zusammengefasst. Dadurch wird der Quellcode kürzer.
//Diese Unterprogramme müssen in jedes Skript (in den einzelnen Modulen),
//welches diese verwendet kopiert werden.

//Winkel zwischen 2 vektoren berechnen (wobei der Nullpunkt auch angegeben werden muss!!!
Angle_between_Vectors_Calc(v0,v1,v2)
{
  println(" ");
  println("Winkel zwischen 2 Vektoren berechnen.");
  println(" ");
  var L1, L2;
  println("1.Vektor: 0.Punkt nach 1.Punkt");
  println("2.Vektor: 0.Punkt nach 2.Punkt");
  println("0. Punkt",v0," Nullpunkt)");
  println("Die zwei Punkte müssen um den Nullpunkt verschoben werden.");
  println("1. Punkt",v1);
  v1=v1-v0;
  v1.y=0;//Da der Winkel in der XZ-Ebene berechntet wird -> Achse=0;
  println("1. Punkt",v1," Neuer");
  println("2. Punkt",v2);
  v2=v2-v0;
  v2.y=0;//Da der Winkel in der XZ-Ebene berechntet wird -> Achse=0;
  L1=vlen(v1);
  L2=vlen(v2);
  V_Angle=180*acos((v1.x*v2.x+v1.y*v2.y+v1.z*v2.z)/(L1*L2))/pi;
  println("2. Punkt",v2," Neuer");
  println("Winkel: ",V_Angle,"°");
  return V_Angle;
}

//XZ-Abstand zwischen 2 Punkten berechnen
XZ_Dist_2_Points_Calc(p1,p2)
{
  //println(" ");
  //println("XZ-Abstand zwischen 2 Punkten berechnen");
  var DistPoint_X,DistPoint_Z;
  DistPoint_X=p2.x-p1.x;
  DistPoint_Z=p2.z-p1.z;
  Dist_current_Point=sqrt((DistPoint_X*DistPoint_X)+(DistPoint_Z*DistPoint_Z));
  println("X-Abstand= ",DistPoint_X);
  println("Z-Abstand= ",DistPoint_Z);
  println("XZ-Abstand(Diagonale)= ",Dist_current_Point);
  return Dist_current_Point;
}

//XYZ-Abstand zwischen 2 Punkten berechnen
XYZ_Dist_2_Points_Calc(p1,p2)
{
  //println(" ");
  //println("XYZ-Abstand zwischen 2 Punkten berechnen");
  var DistPoint_X,DistPoint_Y,DistPoint_Z;
  var i;
  DistPoint_X=p2.x-p1.x;
  DistPoint_Y=p2.y-p1.y;
  DistPoint_Z=p2.z-p1.z;
  DistPoint_Y=DistPoint_Y/Grade_Factor;  //Die Höhe spielt eine untergeordnete Rolle -> Radfahrer
  Dist_current_Points=sqrt((DistPoint_X*DistPoint_X)+(DistPoint_Y*DistPoint_Y)+(DistPoint_Z*DistPoint_Z));
  return Dist_current_Points;
}

//Punktdaten der Brückenoberfläche (LoftNurb) einlesen
GENERATE_BRIDGE_GEOMETRY_ARRAY()
{
  println(" ");
  println("PUNKTE_ARRAY der GESAMTEN aktuellen BRÜCKENGEOMETRIE erzeugen!");
  println(" ");
  var a,cnt;
  var SplineNumber;
  var SplineName,Spline; //Variblen für den SplineChooser
  var Spline_Points = new(array, SplinePunkte);
  for (a=1;a<=Anzahl_Splines;a++) //Start Spline Chooser
  {
    SplineName=(stradd("OriginalSpline",tostring(a)));
    if(doc->FindObject(SplineName))
    {
      Spline=doc->FindObject(SplineName);
      //println("Punkte von Spline:",SplineName," in BRIDGE_POINTS_ARRAY schreiben!!!");
      SplineNumber=a-1;
      //Spline Punkte auslesen
      for (cnt=0;cnt<SplinePunkte;cnt++)
      {
        Spline_Points[cnt]=Spline->GetPoint(cnt);
        BRIDGE_POINTS_ARRAY[SplineNumber][cnt]=Spline_Points[cnt];
        //println("BRIDGE_POINTS_ARRAY[",SplineNumber,",",cnt,"] schreiben: ",BRIDGE_POINTS_ARRAY[SplineNumber][cnt]);
      }
    }
    else println("Haven't found yet");
  } // Ende ARRAY ERZEUGEN FOR SCHLEIFE
  return BRIDGE_POINTS_ARRAY;
}

//UPDATE_BRIDGE_SPLINES
UPDATE_BRIDGE_SPLINES()
{
  var SplineName,Spline,a,cnt;
  var New_Spline_Points=new(array,SplinePunkte);
  for (a=1;a<=Anzahl_Splines;a++) //Start Spline Chooser
  {
    SplineName=(stradd("OriginalSpline",tostring(a)));
    if(doc->FindObject(SplineName))
    {
      for(cnt=0;cnt<SplinePunkte;cnt++)
      {
        Spline=doc->FindObject(SplineName);
        New_Spline_Points[cnt]=BRIDGE_POINTS_ARRAY[a-1][cnt];
        //println(a-1,"|",cnt,"Point_Koordinaten: ",New_Spline_Points[cnt]);
      }
      Spline->SetPoints(New_Spline_Points);
    }
  }
}

//Berechnent die Punkte der 2 zu generierenden Grundpfade zwischen den Brückenenden
PATH_POINTS_CALC()
{
  //-------------------------Punkte Position erzeugen---------------------------------
  var i=0;
  var X_Cord,Y_Cord,Z_Cord;
  var X_Dist,Y_Dist,Z_Dist;
  var SplineNr;
  var Last_Path_Point_Num=Path_Punkte+1;
  var zufall_1=new(Random);
  var zahl_1,zahl_2;
  zufall_1->Init(time());
  var zahl_2=zufall_1->Get01();
  Path_Points=new(array,Last_Path_Point_Num+1);
  //Anfangspunkt generieren
  zahl_1=zufall_1->Get01();
  if(zahl_1>=0.5) SplineNr=Anzahl_Splines-1;
  if(zahl_1<0.5) SplineNr=0;
  //SplineNr=int(Anzahl_Splines*zahl_1);
  //if(SplineNr==0) SplineNr=1;
  //if(SplineNr==Anzahl_Splines) SplineNr=Anzahl_Splines-2;
  println("1. SplineNummer",SplineNr);
  Path_Points[0]=BRIDGE_POINTS_ARRAY[SplineNr][0];
  //Endpunkt generieren
  zahl_2=zufall_1->Get01();
  if(zahl_2>=0.5) SplineNr=Anzahl_Splines-1;
  if(zahl_2<0.5) SplineNr=0;
  //SplineNr=int(Anzahl_Splines*zahl_2);
  //if(SplineNr==0) SplineNr=1;
  //if(SplineNr==Anzahl_Splines) SplineNr=Anzahl_Splines-2;
  println("2. SplineNummer",SplineNr);
  Path_Points[Last_Path_Point_Num]=BRIDGE_POINTS_ARRAY[SplineNr][SplinePunkte-1];
  //--------Achsdistanzen ausrechnen-------------------------
  println("Achsen Distanzen berechnen!");
  X_Dist=Path_Points[0].x-Path_Points[Last_Path_Point_Num].x;
  Y_Dist=Path_Points[0].y-Path_Points[Last_Path_Point_Num].y;
  Z_Dist=Path_Points[0].z-Path_Points[Last_Path_Point_Num].z;
  //--------Segmentlängen für gleichmäßige Punktsetzung----
  X_Dist=X_Dist/(Last_Path_Point_Num);
  Y_Dist=Y_Dist/(Last_Path_Point_Num);
  Z_Dist=Z_Dist/(Last_Path_Point_Num);
  //---------Punkte zwischen den Auflagern generieren-----
  for(i=1;i<Last_Path_Point_Num;i++)
  {
    X_Cord=Path_Points[0].x-X_Dist*i;
    Y_Cord=Path_Points[0].y-Y_Dist*i;
    Z_Cord=Path_Points[0].z-Z_Dist*i;
    Path_Points[i]=vector(X_Cord,Y_Cord,Z_Cord);
  }
  for(i=0;i<=Last_Path_Point_Num;i++)
  {
    println("Punkt",i," Koordinaten:",Path_Points[i]);
  }
  return Path_Points;
```

```
}

//Spline mit eine bestimmten Punkteanzahl Zeichnen
DRAW_PATH_SPLINE()
{
  println(" ");
  println("PATHSPLINE zeichnen!!!");
  println(" ");
  //------------------------------SplineObjekt erzeugen---------------------------------
  var Path_Spline;
  var Path_Name=("Pedestrian_Path_Spline");
  var Path_Splines_Null;
  var Path_Splines_Null_Name=("Path_Splines");
  var i;

  //NullObject Als Ordner ERSTELLEN
  if(!doc->FindObject(Path_Splines_Null_Name))
  {
    Path_Splines_Null=new(NullObject);//Neues InstanzObjekt //Hier könnte man ander Objekte nehmen
    Path_Splines_Null->SetName(Path_Splines_Null_Name); //InstanzObjektName zuweisen
    doc->InsertObject(Path_Splines_Null,Null,Null);//Fügt ein Instancobjekt ein, wobei das Parent Objekt des die
  }
  if(!doc->FindObject(Path_Name))
  {
    Path_Splines_Null=doc->FindObject(Path_Splines_Null_Name);
    Path_Spline=new(SplineObject);
    var vc=new(VariableChanged);
    var bt=new(BackupTags);
    bt->Init(Path_Spline);    // required because we are changing the number of points
    vc->Init(0,Path_Punkte+2); // it used to have 0 points, now has 5
    if (Path_Spline->Message(MSG_POINTS_CHANGED, vc))
    {
      bt->Restore();
    }
    doc->InsertObject(Path_Spline,Path_Splines_Null,NULL);
    Path_Spline->SetName(Path_Name);
    Path_Spline#SPLINEOBJECT_TYPE=3; // set the type of spline here
}

//MAKE_RECTANGLE
MAKE_RECTANGLE(Left_Pos,Right_Pos,Height,Name, Parent)
{
  var Rectangle_Object;
  var Rectangle_Points=new(array,4);

  if(!doc->FindObject(Name))
  {
    Rectangle_Object=new(SplineObject);
    var vc=new(VariableChanged);
    var bt=new(BackupTags);
    bt->Init(Rectangle_Object);    // required because we are changing the number of points
    vc->Init(0,4); // it used to have 0 points, now has 5
    if (Rectangle_Object->Message(MSG_POINTS_CHANGED, vc))
    {
      bt->Restore();
    }
    doc->InsertObject(Rectangle_Object,Parent, null);
    Rectangle_Object->SetName(Name);
    Rectangle_Object#SPLINEOBJECT_TYPE=0; // set the type of spline here
    Rectangle_Object#SPLINEOBJECT_CLOSED=True;
  }

  Rectangle_Points[0]=Left_Pos;
  Left_Pos.y=Left_Pos.y-Height;
  Rectangle_Points[1]= Left_Pos;
  Right_Pos.y=Right_Pos.y-Height;
  Rectangle_Points[2]= Right_Pos;
  Right_Pos.y=Right_Pos.y+Height;
  Rectangle_Points[3]=Right_Pos;
  //println(Path_Punkte);
  Rectangle_Object=doc->FindObject(Name);
  Rectangle_Object->SetPoints(Rectangle_Points);
  Rectangle_Object->Message(MSG_UPDATE);
}

//MAKE_SPHERE
MAKE_SPHERE(Scale,Name,Position,Parent)
{
  var Sphere;
  if(!doc->FindObject(Name))
  {
    Sphere = new(SphereObject);
    Sphere->SetName(Name);
    Sphere#PRIM_SPHERE_RAD=Scale;
    Sphere->SetPosition(Position);
    doc->InsertObject(Sphere,Parent, null);
  }
  else
  {
    Sphere = doc->FindObject(Name);
    Sphere->SetPosition(Position);
    Sphere#PRIM_SPHERE_RAD=Scale;
  }
  Sphere->Message(MSG_UPDATE);
}



//-----------------------------------------------------
//GENERATE_BASIC_SURFACE from SPLINE A to B  |
//-----------------------------------------------
//by Christian Pichlkastner
//
//In diesem Skript wird die Matrix, Grundknetfläche bzw.Brückenoberfläche erzeugt.
//
//Input: Typ, Aufl_A, Aufl_B, Bridge_Width, ONOFF, SplineAnzahl, SplinePunkte
//Output: Output1

//GLOBALE VARIABLEN
var Spline_A_Position,Spline_B_Position;
var LetzterPunkt;
var Count;
var doc;
var LoftObjekt,LoftObjekt_Name;
var zufall_1,zufall_2,zahl_1,zahl_2;
var percent_width;

//Brückenendpunke erzeugen
ENDPUNKTE_ERZEUGEN(PointArray,doc,LetzterPunkt)
{
  println("-------------------------------------");
  //println("IN SUB_FUNKTION ENDPUNKTE ERZEUGEN!");
  //------Position in Prozent berechnen---------
  var Spline_A_Length,Spline_A_Name,ProzentPosition_A,SplineA,SplineAbstand_A;
  var Spline_B_Length,Spline_B_Name,ProzentPosition_B,SplineB,SplineAbstand_B;
  var Overlap_Check;
  var Width;
  //-------------------------------------------------------------------------------------
  println("Position Auflager A",zahl_1);
  Spline_A_Name=Aufl_A->GetName();
  SplineA=doc->FindObject(Spline_A_Name);
  SplineA->InitLength(0);
  Spline_A_Length =SplineA->GetLength();
  Width=Bridge_Width;

  if(Bridge_Width>=Spline_A_Length) Width=Spline_A_Length;

  println("Spline_A_Length: ",Spline_A_Length);
  println("Bridge_Width: ",Bridge_Width);
  //Wenn Spline länge 100% dann sind Breite/Anzahl wieviel Prozent?
  percent_width=((1/Spline_A_Length)*Width); //In Prozent umrechnen
  println("Width in Prozent der Gesamtlänge: ",percent_width);
  SplineAbstand_A=percent_width/(SplineAnzahl-1);
  println("SplinesZwischenAbstand_A: ",SplineAbstand_A);
  //------Wenn die Brücke außerhalb des Auflagers aufliegen würde---
  Overlap_Check=zahl_1+(SplineAnzahl-1)*SplineAbstand_A;
  println("1.Overlap_Check=",Overlap_Check);

  if (Overlap_Check>=1) zahl_1=zahl_1-(Overlap_Check-1);

  if (Overlap_Check<=0) zahl_1=zahl_1+(Overlap_Check-1);

  println("gändert ZAHL_1: ",zahl_1);
  ProzentPosition_A=zahl_1+SplineAbstand_A*(Count-1); //Prozentposition ausrechnen
  println("aktuelle PROZENTPOSITION_A: ",ProzentPosition_A);
  //-------------------------------------------------------------------------------------
  println("Position Auflager B",zahl_2);
  Spline_B_Name=Aufl_B->GetName();
  SplineB=doc->FindObject(Spline_B_Name);
  SplineB->InitLength(0);
  Spline_B_Length =SplineB->GetLength();
  Width=Bridge_Width;

  if(Bridge_Width>=Spline_B_Length) Width=Spline_B_Length;
  println("Spline_B_Length: ",Spline_B_Length);
  println("Bridge_Width: ",Bridge_Width);
  //Wenn Spline länge 100% dann sind Breite/Anzahl wieviel Prozent?
  percent_width=((1/Spline_B_Length)*Width); //In Prozent umrechnen
```

```
  println("Width in Prozent der Gesamtlänge: ",percent_width);
  SplineAbstand_B=percent_width/(SplineAnzahl-1);
  println("SplinesZwischenAbstand_B: ",SplineAbstand_B);
  //------Wenn die Brücke außerhalb des Auflagers aufliegen würde---
  Overlap_Check=zahl_2+(SplineAnzahl-1)*SplineAbstand_B;
  println("2.Overlap_Check=",Overlap_Check);
  if (Overlap_Check>=1) zahl_2=zahl_2-(Overlap_Check-1);

  if (Overlap_Check<=0) zahl_2=zahl_2+(Overlap_Check-1);

  println("gändert ZAHL_2: ",zahl_2);
  ProzentPosition_B=zahl_2+SplineAbstand_B*(Count-1); //Prozentposition ausrechnen
  println("aktuelle PROZENTPOSITION_B: ",ProzentPosition_B);
  //----------------SplinePunktPoistion ermitteln------------
  println("SplinePunktPosition ermitteln:");
  //Path_1_Pos_Temp1=Path_1_Object->GetSplinePoint(Path_1_Object->UniformToNatural(Percent_Position_I
  Spline_A_Position=Aufl_A->GetSplinePoint(Aufl_A->UniformToNatural(ProzentPosition_A),0);
  //Spline_A_Position=Aufl_A->GetSplinePoint(ProzentPosition_A,0);
  //Spline_B_Position=Aufl_B->GetSplinePoint(ProzentPosition_B,0);
  Spline_B_Position=Aufl_B->GetSplinePoint(Aufl_B->UniformToNatural(ProzentPosition_B),0);
  PointArray[0]=Spline_A_Position;
  PointArray[LetzterPunkt]=Spline_B_Position;
  return PointArray,Spline_A_Position,Spline_B_Position,LetzterPunkt;
}

//Punkte zwischen den Uferzonen erzeugen
PUNKTE_ERZEUGEN(PointArray,Spline_A_Position,Spline_B_Position)
{
  //-------------------------Punkte Position erzeugen---------------------------------
  var i=0;
  var X_Cord,Y_Cord,Z_Cord;
  var X_Dist,Y_Dist,Z_Dist;
  println("Bin in der Restl.-PUNKTE_ERZEUGEN Funktion");
  //println("Der letzte Punkt ist Punkt Nummer ",LetzterPunkt);
  PointArray[i]=Spline_A_Position;
  //println("Punkt",i," Koordinate_Auflager A:",Spline_A_Position); //---------Anfanfspunkt------
  PointArray[LetzterPunkt]=Spline_B_Position;
  //println("Punkt",LetzterPunkt," Koordinaten_Auflager B:",Spline_B_Position); //---------Endpunkt------
  //--------Achsdistanzen ausrechnen-------------------------
  /*
  println("Achsen Distanzen berechnen!");
  println("Punkt_Aufl_A.x:",Spline_A_Position.x);
  println("Punkt_Aufl_A.y:",Spline_A_Position.y);
  println("Punkt_Aufl_A.z:",Spline_A_Position.z);
  println("Punkt_Aufl_B.x:",Spline_B_Position.x);
  println("Punkt_Aufl_B.y:",Spline_B_Position.y);
  println("Punkt_Aufl_B.z:",Spline_B_Position.z);
  */
  X_Dist=Spline_B_Position.x-Spline_A_Position.x;
  Y_Dist=Spline_B_Position.y-Spline_A_Position.y;
  Z_Dist=Spline_B_Position.z-Spline_A_Position.z;
  /*
  println("X-Distanz vor der Schleife =",X_Dist);
  println("Y-Distanz vor der Schleife =",Y_Dist);
  println("Z-Distanz vor der Schleife =",Z_Dist);
  */
  //--------Segmentlängen für gleichmäßige Punktsetzung----
  X_Dist=X_Dist/(LetzterPunkt);
  Y_Dist=Y_Dist/(LetzterPunkt);
  Z_Dist=Z_Dist/(LetzterPunkt);
  /*
  println("Segmentgröße X=",X_Dist);
  println("Segmentgröße Y=",Y_Dist);
  println("Segmentgröße Z=",Z_Dist);
  */
  //---------Punkte zwischen den Auflagern generieren-----
  for(i=1;i<LetzterPunkt;i++)
  {
    /*
    println(i);
    println("X-Distanz in der Schleife für Punkt",i,"=",X_Dist);
    println("Y-Distanz in der Schleife für Punkt",i,"=",Y_Dist);
    println("Z-Distanz in der Schleife für Punkt",i,"=",Z_Dist);
    */
    X_Cord=Spline_A_Position.x+X_Dist*i;
    Y_Cord=Spline_A_Position.y+Y_Dist*i;
    Z_Cord=Spline_A_Position.z+Z_Dist*i;
    PointArray[i]=vector(X_Cord,Y_Cord,Z_Cord);
    /*
    println("Punkt",i," Auflager A X-Wert",Spline_A_Position.x);
    println("Punkt",i," Koordinaten:",PointArray[i]);
    */
  }
  return PointArray;
}

//-----------------------------------------------------------------------------------------------
//-----------------------------------------------------------------------------------------------
main()
{
  var PointArray=new(array,SplinePunkte); // this example creates a spline with 5 points
  doc=GetActiveDocument();
  var SplinePunkteAnzahl;
  Count=1;
  SplinePunkteAnzahl=SplinePunkte;
  LetzterPunkt=SplinePunkteAnzahl-1;
  LoftObjekt_Name=("LoftObjekt");
  if(ONOFF==TRUE)
  {
    //LoftObject als Ordner ERSTELLEN
    if(!doc->FindObject("LoftObjekt"))
    {
      LoftObjekt=new(LoftObject);//Neues InstanzObjekt //Hier könnte man ander Objekte nehmen
      LoftObjekt->SetName(LoftObjekt_Name); //InstanzObjektName zuweisen
      doc->InsertObject(LoftObjekt,Null,Null);//Fügt ein Instancobejt ein, wobei das Parent Objekt des die Refenz ist
    }
    println("Erzeuge Zufallszahlen für Auflagerpositionen:");
    zufall_1=new(Random);
    zufall_2=new(Random);
    zufall_1->Init(time());
    zufall_2->Init(time());
    zahl_1=zufall_1->Get01();
    zahl_2=zufall_1->Get01();
    zahl_1=zufall_1->Get01();
    println("zahl_1 START:",zahl_1);
    println("zahl_2 START:",zahl_2);
    for(Count=1;Count<=SplineAnzahl;Count++)
    {
      //Brückenendpunke erzeugen
      ENDPUNKTE_ERZEUGEN(PointArray,doc,LetzterPunkt);
      //Punkte zwischen den Uferzonen erzeugen
      PUNKTE_ERZEUGEN(PointArray,Spline_A_Position,Spline_B_Position);
      //Spline Zeichen
      DRAW_SPLINE(PointArray);
    }
    //LoftNurbs-Updaten
    LoftObjekt=doc->FindObject(LoftObjekt_Name);
    LoftObjekt#ID_BASEOBJECT_GENERATOR_FLAG=False;
    LoftObjekt#ID_BASEOBJECT_GENERATOR_FLAG=True;
    Output1=FALSE;
  }
}



//----------------------------------
//SIMPLE_SPLINE_DEFORMER  |
//----------------------------------
//by Christian Pichlkastner
//Dieses Cinema4d Coffee Skript verzerrt Splines mittels eines Zufallgenerators.
//
//Eingang: Anzahl_Splines,SplinePunkte, Faktor, ONOFF1
//Ausgang: OUT
Spline_Deform(Spline,Spline_Points,DeformedPoints,cnt,zahl_1,SplinePunkte,cnt,zufall_1)
{
  for(cnt;cnt<(SplinePunkte-1);cnt++)
  {
    zahl_1=zufall_1->Get01(); //Neue Zufallszahl
    println("In Verzerren-Unterprogramm:");
    println("Punkt",cnt," X-Koordinaten:",DeformedPoints[cnt].x);
    println("Punkt",cnt," Y-Koordinaten:",DeformedPoints[cnt].y);
    println("Punkt",cnt," Z-Koordinaten:",DeformedPoints[cnt].z);
    println("Zufallszahl: ",zahl_1);
    DeformedPoints[cnt].x=DeformedPoints[cnt].x+(Faktor*0.1)*cos(zahl_1*cnt*0.3);   //je größer die zahl in der Sinusklammer desto hö
    DeformedPoints[cnt].y=DeformedPoints[cnt].y+(Faktor*5)*cos(zahl_1*cnt*(Faktor*4)); //je größer die zahl in der Sinusklammer des
    DeformedPoints[cnt].z=DeformedPoints[cnt].z+(Faktor*0.1)*cos(zahl_1*cnt*1); //je größer die zahl in der Sinusklammer desto höher
  }
  Spline_Points=DeformedPoints;
  Spline->SetPoints(Spline_Points);
}

//-----------------------------------------------------------------------------------------------
//-----------------------------------------------------------------------------------------------
main()
{
  var doc=GetActiveDocument();
  var Spline,Spline_Name,Spline_Points,DeformedPoints;
  var a,cnt;
```

```
  var zufall_1=new(Random);
  zufall_1->Init(time());
  var zahl_1=zufall_1->Get01();
  if (ONOFF1==TRUE && doc->FindObject("LoftObjekt") && doc->FindObject("OriginalSpline1"))
  {
    for(a=1;a<=Anzahl_Splines;a++) //Start Spline Chooser
    {
      println("Zufallszahl_", zahl_1);
      Spline_Name=(stradd("OriginalSpline",tostring(a)));
      if(doc->FindObject(Spline_Name))
      {
        Spline=doc->FindObject(Spline_Name);
        Spline_Name=Spline->GetName();
        println("Found: ",Spline_Name);
        Spline_Points=Spline->GetPoints();
        DeformedPoints=Spline_Points;
        //Deform Spline
        if (a<=Anzahl_Splines) //blödsinn--
        {
          cnt=1;
          Spline_Deform(Spline,Spline_Points,DeformedPoints,cnt,zahl_1,SplinePunkte,cnt,zufall_1);
        }
      }
      else println("Haven't found Spline to deform");
    }//Ende ForSchleifge
    var LoftnurbsNEW,Loftname;
    Loftname=("LoftObjekt");
    LoftnurbsNEW=doc->FindObject(Loftname);
    LoftnurbsNEW#ID_BASEOBJECT_GENERATOR_FLAG=False;
    LoftnurbsNEW#ID_BASEOBJECT_GENERATOR_FLAG=True;
  }//If Input1==True -> Ende
  else
  {
    if (ONOFF1==TRUE) println("Splines können nicht verzerrt werden, da kein LoftObjekt und keine Splines vorl
  }
  OUT=False;
}



//-------------------------------------
//MAKE_BARRIER_FREE_SCRIPT  |
//-------------------------------------
//by Christian Pichlkastner
//
//Dieses Cinema4d Coffee Skript überprüft eine Spline auf Barrierefreiheit.
//Wenn diese nicht barrierefrei ist dann wird diese barrierefrei gemacht!!!
//
//Eingang: Spline, ONOFF, Steigung
//Ausgang: OFF
//Globale Variable
var CHECKED, V_Angle, X12, X34;
var LastPoint,FirstPoint,Ideal_Line_Degree,Ideal_Line_XZ_Length;
var Dist_MaxEbene_X,Dist_MaxEbene_Y,Dist_MaxEbene_Z,Dist_current_Point;
var PointLenght_on_XZ_Ideal_Line,Y_,Y_Divergence_to_Ideal_Line;
var pt,Curve_Degree;
var Y_DISTANCE,Biggest_Value,Biggest_Point,Scale_Faktor;

//Überprüfen ob Barrierefreiheit überhaupt möglich----------
Check_if_Barrier_Free_is_Possible(SplineObjekt,SplinePunkte)
{
  var Checked,Possible_Y;
  println(" ");
  println("ÜBERPRÜFEN OB BARRIEREFREIHEIT ÜBERHAUPT MÖGLICH IST!");
  println("-----------------------------------------------------------------------------------------------------");
  println("Anfangs- und Endpunkt finden.");
  FirstPoint=SplineObjekt->GetPoint(0);
  LastPoint=SplineObjekt->GetPoint(SplinePunkte-1);
  println("Die Spline besteht aus ",SplinePunkte," Punkten.");
  println("Firstpoint: X=",FirstPoint.x," Y=",FirstPoint.y," Z=",FirstPoint.z);
  println("Lastpoint: X=",LastPoint.x," Y=",LastPoint.y," Z=",LastPoint.z);
  println("Abstand der 2 Punkte im Raum ermitteln.");
  Dist_MaxEbene_X=LastPoint.x-FirstPoint.x;
  Dist_MaxEbene_Y=LastPoint.y-FirstPoint.y;
  Dist_MaxEbene_Z=LastPoint.z-FirstPoint.z;
  println("Distanz auf X-Achse: ",Dist_MaxEbene_X);
  println("Distanz auf Y-Achse: ",Dist_MaxEbene_Y);
  println("Distanz auf Z-Achse: ",Dist_MaxEbene_Z);
  Ideal_Line_XZ_Length=sqrt((Dist_MaxEbene_X*Dist_MaxEbene_X)+(Dist_MaxEbene_Z*Dist_MaxEbene_Z));
  println("Die Ideal_Line_XZ ist ",Ideal_Line_XZ_Length, " [cm] lang.");
  Ideal_Line_Degree=Dist_MaxEbene_Y/Ideal_Line_XZ_Length;
  println("Curve_Degree der Idealen Verbindungslinie ",Ideal_Line_Degree);
  println("Aktuell geforderte barriefreiheit:",Curve_Degree);
  Possible_Y=Curve_Degree*Ideal_Line_XZ_Length;
  println("Maximaler Y-Höhen unteschied darf ",Possible_Y," betragen");
  println("Aktueller Y-Höhenunterschied beträgt: ",abs(Dist_MaxEbene_Y));
  if (abs(Dist_MaxEbene_Y ) > Curve_Degree*Ideal_Line_XZ_Length))
  {
    println("Barrierefreiheit ist nicht möglich!!!");
    return CHECKED=False;
  }
  else println("Barrierefreiheit ist möglich!!!");

  return CHECKED=True,FirstPoint,LastPoint,Ideal_Line_Degree,Ideal_Line_XZ_Length,Dist_MaxEbene_X,Dist_MaxEbene_Y
}

//X12-Länge und X34-Länge bestimmen. -> Für spätere Entscheidung, welche Geradengleichung gewählt werden muss
//Barrierefrei Grenzen ermitteln.
Get_X12_X34_Calc()
{
  println(" ");
  println("Ermittlung der Punkte X12 und X34 auf der XZ Ideal-Linie");
  println(" ");
  var d1,d2;
  d1=FirstPoint.y-Curve_Degree*FirstPoint.x;
  d2=LastPoint.y+Curve_Degree*(FirstPoint.x+Ideal_Line_XZ_Length);
  X12=((d2-d1)/(2*Curve_Degree))-FirstPoint.x;
  X34=Ideal_Line_XZ_Length-X12;
  println("d1: ",d1);
  println("d2: ",d2);
  println("X12: ",X12);//Länge an der XZ-Ebene der Ideal-Gerade
  println("X34: ",X34);//Länge an der XZ-Ebene der Ideal-Gerade
  return X12,X34;
}

//PointLenght_on_Ideal_Line_Calc
PointLenght_on_Ideal_Line_Calc(SplinePunkte)
{
  println(" ");
  println("Neuen Länge für den aktuellen Punkt auf Idealgeraden ausrechnen");
  println(" ");
  PointLenght_on_XZ_Ideal_Line=Dist_current_Point*cos(V_Angle*(pi/180)); //winkel muss in rad umgerechnet werden =
  println("cos(",V_Angle,")*Länge zwischen dem Anfangspunt und dem Akutellen Punkt: ",Dist_current_Point," = Ankathete:
  return PointLenght_on_XZ_Ideal_Line;
}

//Y-Abstand des akuellen Punktes zur Ideal-Geraden berstimmen --> Um später mitbestimmen zu können welche Geraden
Y_Divergence_to_Ideal_Line_Calc(cnt,Ideal_Line_Degree,pt,PointLenght_on_XZ_Ideal_Line,Point_Cord_on_Ideal_Line)
{
  println(" ");
  println("Y-Abstand des akuellen Punktes zur Ideal-Geraden berstimmen");
  println(" ");
  Y_Divergence_to_Ideal_Line=pt.y-(FirstPoint.y+PointLenght_on_XZ_Ideal_Line*Ideal_Line_Degree);
  println("Aktueller Abstand von Idealgeraden: ",Y_Divergence_to_Ideal_Line);
  Point_Cord_on_Ideal_Line[cnt].y=FirstPoint.y+PointLenght_on_XZ_Ideal_Line*Ideal_Line_Degree;
  return Y_Divergence_to_Ideal_Line,Point_Cord_on_Ideal_Line;
}
//Spline Y-Abwechuungen in den barrierefrei_machbaren Bereich verschieben
Force_Spline_into_BarrierFree_Bounds(cnt,Bounds_Checked_Spline_Points)
{
  println(" ");
  println("Spline Y-Abweichuungen in den barrierefrei_machbaren Bereich verschieben");
  println(" ");
  var Actuall_Curve_Degree;
  if (Curve_Degree<Ideal_Line_Degree)
  {
    Actuall_Curve_Degree=Ideal_Line_Degree;
  }
  else Actuall_Curve_Degree=Curve_Degree;
  println("Aktuelle Punkt Y.Koordinate: ",pt.y);
  println("Y-Abweichung zur IdealLinie 'POSITIV' oder 'NEGATIV'?");
  println("Y-Abweichung von IdealLinie: ",Y_Divergence_to_Ideal_Line);
  println("PunktvektorLänge auf IdealGeraden: ",PointLenght_on_XZ_Ideal_Line);
  println("Größer oder kleiner X12 oder X34?");
  println("X12: ",X12);
  println("X34: ",X34);
  //Entscheidung welche Geradengleichung als Vergleich hergenommen wird.
  if(Y_Divergence_to_Ideal_Line>=0) //Wenn Y_Divergence_to_Ideal_Line positiv ist!!!
  {
    if(PointLenght_on_XZ_Ideal_Line<=X12)
    {
      println("1. Geradengleichung");
      if(pt.y>FirstPoint.y+Actuall_Curve_Degree*PointLenght_on_XZ_Ideal_Line) //Anfangspunkt+Punktlänge*Curve_Degree
      {
        println("Der Y-Wert aktuellen Punktes beträgt ",pt.y);
        pt.y=FirstPoint.y+Actuall_Curve_Degree*PointLenght_on_XZ_Ideal_Line;
        println("Der aktuelle Punkt wird auf ",pt.y," korrigiert!!!");
      }
```

```
      else println("Der aktuelle Punkt muss nicht korrigiert werden!");
    }
    else
    {
      println("2. Geradengleichung");
      if(pt.y>LastPoint.y+Actuall_Curve_Degree*(Ideal_Line_XZ_Length-PointLenght_on_XZ_Ideal_Line))
      {
        println("Der Y-Wert aktuellen Punktes beträgt ",pt.y);
        pt.y=LastPoint.y+Actuall_Curve_Degree*(Ideal_Line_XZ_Length-PointLenght_on_XZ_Ideal_Line);
        println("Der aktuelle Punkt wird auf ",pt.y," korrigiert!!!");
      }
      else println("Der aktuelle Punkt muss nicht korrigiert werden!");
    }
  }
  else //Wenn Y_Divergence_to_Ideal_Line negativ ist!!!
  {
    if(PointLenght_on_XZ_Ideal_Line<=X34)
    {
      println("3. Geradengleichung");
      if(pt.y<FirstPoint.y-Actuall_Curve_Degree*PointLenght_on_XZ_Ideal_Line)
      {
        println("Der Y-Wert aktuellen Punktes beträgt ",pt.y);
        pt.y=FirstPoint.y-Actuall_Curve_Degree*PointLenght_on_XZ_Ideal_Line;
        println("Der aktuelle Punkt wird auf ",pt.y," korrigiert!!!");
      }
      else println("Der aktuelle Punkt muss nicht korrigiert werden!");
    }
    else
    {
      println("4. Geradengleichung");
      if(pt.y<LastPoint.y-Actuall_Curve_Degree*(Ideal_Line_XZ_Length-PointLenght_on_XZ_Ideal_Line))
      {
        println("Der Y-Wert aktuellen Punktes beträgt ",pt.y);
        pt.y=LastPoint.y-Actuall_Curve_Degree*(Ideal_Line_XZ_Length-PointLenght_on_XZ_Ideal_Line);
        println("Der aktuelle Punkt wird auf ",pt.y," korrigiert!!!");
      }
      else println("Der aktuelle Punkt muss nicht korrigiert werden!");
    }
  }
  return Bounds_Checked_Spline_Points[cnt]=pt;
}

//Y_Abstand zwischen 2 Splines ermitteln
Get_Distance_Between_Splines(SplinePunkte,Original_Spline_Points,Bounds_Checked_Spline_Points)
{
  println(" ");
  println("Y-Abstände zischen Original und VergleichsSpline ermitteln");
  println(" ");
  var i,Y_Dist=new(array,SplinePunkte);
  for (i=0;i<SplinePunkte;i++)
  {
    Y_Dist[i]=Original_Spline_Points[i].y-Bounds_Checked_Spline_Points[i].y;
    println(i,".Punkt OriginalSpline            : ",Original_Spline_Points[i].y);
    println(i,".Punkt BoundCheckedSpline: ",Bounds_Checked_Spline_Points[i].y);
    println(i,".Punkt Y-Distanz: ",Y_Dist[i]);
  }
  Y_DISTANCE=Y_Dist;
  return Y_DISTANCE;
}

//Größten Wert eines Array-Inhaltes ausgeben
Biggest_Value_Calc(SplinePunkte)
{
  println(" ");
  println("Größte Y-Abweichung berechnen.");
  println(" ");
  var i,Biggest_Y,PointNumber;
  Biggest_Value=Y_DISTANCE[0];
  PointNumber=0;
  for (i=1;i<SplinePunkte;i++)
  {
    println(i,". Vorheriger Wert: ",abs(Y_DISTANCE[i-1]));
    println(i,". Momentaner Wert: ",abs(Y_DISTANCE[i]));
    if( Biggest_Value < abs(Y_DISTANCE[i]))
    {
      Biggest_Value=abs(Y_DISTANCE[i]);
      PointNumber=i;
    }
    println(i,". größte Abweichung: ",Biggest_Value);
  }
    println("Punkt ",PointNumber," hat mit einem Betrag von: ",Biggest_Value," die größte Y-Abweichung!");
    Biggest_Point=PointNumber;
    return Biggest_Value,Biggest_Point;
}


//Spline anhand von maximaler Abweichung von Bounds_Checked_Spline_Points skallieren
Scale_Spline(SplinePunkte,Scaled_Spline_Points,Bounds_Checked_Spline_Points,Original_Spline_Points,Point_Cord_on_Ideal_Line)
{
  println (" ");
  println ("SPLINE SKALLIEREN!");
  println (" ");

  var i;
  if(Biggest_Value!=0)
  {
    //Stecke Punkt auf Idealgeraden-BoundaryPoint / Punkt auf Idealgeraden_OriginalSpline
    Scale_Faktor=(Point_Cord_on_Ideal_Line[Biggest_Point].y-Bounds_Checked_Spline_Points[Biggest_Point].y)/(Point_Cord_on_Idea
Original_Spline_Points[Biggest_Point].y);
  }
  else Scale_Faktor=0;
  println ("Skallierfaktor:",Scale_Faktor);
  for(i=0;i<SplinePunkte;i++)
  {
    println("Y_DISTANCE: ",Y_DISTANCE[i]);
    Y_DISTANCE[i]=Y_DISTANCE[i]*Scale_Faktor;
    println("Y_DISTANCE:",Y_DISTANCE[i]," Neu");
  }
  for(i=0;i<SplinePunkte;i++)
  {
    Scaled_Spline_Points[i].y=Point_Cord_on_Ideal_Line[i].y+Y_DISTANCE[i];
  }
  Scaled_Spline_Points=Bounds_Checked_Spline_Points;
  return Scaled_Spline_Points;
}

//SplinePunkt barrierefrei machen
Make_Spline_Barrier_Free(SplinePunkte,Bounds_Checked_Spline_Points, Barrier_Free_Spline_Points)
{
  println (" ");
  println ("SPLINE BARRIEREFREI MACHEN!");
  println (" ");

  var Cord,Cord_before;
  var Dist_X1,Dist_Y1,Dist_Z1,XZ_Dist1;
  var cnt=1,Entscheidung;

  Cord_before=Bounds_Checked_Spline_Points[0];   //Hier wird auch entschieden welche Spline gechecked wird!!!
                                                 //z.B.:SplineObjekt,Bounds_Checked_Spline,Scaled_Spline

  pt=Cord; //pt wird hier ein Array, das später die veränderten Punktkoordinaten vergibt.
  for(cnt=1;cnt<(SplinePunkte-1);cnt++)
  {
    println (" ");
    println ("Barrierefrei Check Punkt",cnt);
    Cord=Bounds_Checked_Spline_Points[cnt];

    //Distanzen ausrechnen
    //Zwischen dem Punkt davor und dem Momentanen (Cnt) Punkt
    //Mögliche Vorzeichenprobleme müssen erst (progrmmiert) ausgeschlossen werden!!!

    println ("Y-Distanz zwischen dem aktuellen Punkt ",cnt," dem Punkt ",cnt-1," davor!");
    Dist_X1=Cord.x-Cord_before.x;
    Dist_Z1=Cord.z-Cord_before.z;
    Dist_Y1=Cord.y-Cord_before.y; //--> positive oder negative Curve_Degree!
    XZ_Dist1=sqrt((Dist_X1*Dist_X1)+(Dist_Z1*Dist_Z1)); //Distanz in der Ebene

    println("Punkt ",cnt-1," Koordinaten: ",Cord_before);
    println("Punkt ",cnt," Koordinaten: ",Cord);
    println("Abstand in XZ-Ebene: ",XZ_Dist1);
    println("DIST_Y1: ",Dist_Y1);
    println("Curve_Degree: ",Curve_Degree);

    //je nach negativen oder positiven Höhenunterschied, wieder entschieden ob positive oder negative Curve_Degree!
    if (Dist_Y1>=0)Entscheidung=1; //positive Richtung, positive Curve_Degree

    if (Dist_Y1<0) Entscheidung=2; //positive Richtung, negative Curve_Degree

    //Y-Vergleich - eventuell korrektur!!!
    switch(Entscheidung)
    {
      case 1:
      println("Wenn Y_Distanz ",abs(Dist_Y1)," größer ist als ",XZ_Dist1* Curve_Degree," dann Punkt barrierefrei machen!");
      if (Dist_Y1> (XZ_Dist1* Curve_Degree))
      {
        println("Punkt wird barrierefrei gemacht");
        Cord.y=Cord_before.y+XZ_Dist1*Curve_Degree;
      }
      break;
```

```
      case 2:
      println("Wenn Y_Distanz ",abs(Dist_Y1)," größer ist als ",XZ_Dist1* Curve_Degree," dann Punkt barrierefrei
      if (Dist_Y1< (XZ_Dist1*(-Curve_Degree)))
      {
        println("Punkt wird barrierefrei gemacht");
        Cord.y=Cord_before.y+XZ_Dist1*(-Curve_Degree);
      }
      break;
    }
    Barrier_Free_Spline_Points[cnt]=Cord; //korriegierten "Boundary_Checked_Spline_Point" in Barriere_Free
    Cord_before=Barrier_Free_Spline_Points[cnt];
  }
  return Barrier_Free_Spline_Points;
}

//-------------------------------------------------------------------------------------------
//-------------------------------------------------------------------------------------------
main()
{
  var OriginalSpline;
  var SplinePunkte,cnt;
  var Original_Spline_Points,Bounds_Checked_Spline_Points, Barrier_Free_Spline_Points,Scaled_Spline_Point
  var doc=GetActiveDocument();
  var a,SplineName; //Variblen für den SplineChooser
  var LoftObjekt;
  if (ONOFF==TRUE && doc->FindObject("LoftObjekt"))
  {
    //---------------------------------------Momentan werden alle Splines barrierefrei gemacht----muss veränder
    for(a=1;a<=3;a++)
    {
      if(a==1) SplineName=("Pedestrian_Path_Spline");
      if(a==2) SplineName=("Biker_Path_Spline");
      if(a==3) SplineName=("Barrier_Free_Path_Spline");
      if(a==1) Curve_Degree=0.30; //zulässige Steigunsgrade in Prozent (Fußgängerweg)
      if(a==2) Curve_Degree=0.15; //zulässige Steigunsgrade in Prozent (Fahrradweg)
      if(a==3) Curve_Degree=Steigung; //zulässige Steigunsgrade in Prozent (barrierefreier Weg)
      if(doc->FindObject(SplineName))
      {
        println("Checking Path: ",SplineName);
        OriginalSpline=doc->FindObject(SplineName);
        SplinePunkte=OriginalSpline->GetPointCount();
        println("Überprüfe Spline:",SplineName," auf Barrierefreiheit!");
      }
      else println("Haven't found yet");

      Original_Spline_Points=OriginalSpline->GetPoints(); //Für späteren Vergleich;
      Bounds_Checked_Spline_Points=OriginalSpline->GetPoints();
      Barrier_Free_Spline_Points=OriginalSpline->GetPoints();
      SplinePunkte=OriginalSpline->GetPointCount();
      Scaled_Spline_Points=OriginalSpline->GetPoints();
      Point_Cord_on_Ideal_Line=OriginalSpline->GetPoints();
      Check_if_Barrier_Free_is_Possible(OriginalSpline,SplinePunkte);

      if(CHECKED=True)
      {
        println(" ");
        println ("BARRIEREFREI MACHEN!");
        println("-----------------------------------------");

        // 1. Teil: SplinePunkte auf den maximal möglichen Y-Abweichungsbereich verkleinern.
        println("1. Teil: SplinePunkte auf den maximal möglichen Y-Abweichungsbereich verkleinern");
        //X12 und X34 bestimmen. -> Für spätere Entscheidung, welche Geradengleichung gewählt werden muss
        Get_X12_X34_Calc();
        //1.Teil Schleife
        for (cnt=1;cnt<SplinePunkte;cnt++)
        {
          println(" ");
          println("--------------------Berechnung SplinePunkt_",cnt,"-------------------------------");
          pt=OriginalSpline->GetPoint(cnt); //Aktuelle Punktkoordinaten einlesen.
          //WINKEL zwischen den Vektoren ermitteln ermitteln--------
          var v0=FirstPoint,v1= LastPoint, v2=pt;
          Angle_between_Vectors_Calc(v0,v1,v2); // Ergebnis in "V_Angle"
          //XZ-Abstand zwischen dem AktuellenPunkt und dem Anfangspunkt berechnen
          var p1=FirstPoint, p2=pt;
          XZ_Dist_2_Points_Calc(p1,p2);// Ergebnis in "Dist_current_Point"
          //Punkt Länge auf XZ-Idealgerade umrechnen -> Ankathete
          PointLenght_on_Ideal_Line_Calc(SplinePunkte); //Ergebnis in "PointLenght_on_XZ_Ideal_Line"
          //Y-Abweichung von Ideallienie Berechnen um festzustellen ob positive oder negative Abweichung.
          Y_Divergence_to_Ideal_Line_Calc(cnt,Ideal_Line_Degree,pt,PointLenght_on_XZ_Ideal_Line,Point_Cord_
"Y_Divergence_to_Ideal_Line","Bounds_Checked_Spline"
          //Wenn momentane Y-Abwechuungen nicht barrierefrei dann in den barrierefrei_machbaren Bereich ve
          Force_Spline_into_BarrierFree_Bounds(cnt,Bounds_Checked_Spline_Points);//Ergebnis in "Bounds_Ch
        }
        //-------------------------------------------------------------------
        // 2. Teil: Y-Abstände zwischen Original und VergleichsSpline ermitteln und die Originalspline eventuell skalieren.
        Get_Distance_Between_Splines(SplinePunkte,Original_Spline_Points,Bounds_Checked_Spline_Points);//Ergebnis in "
        //größte Y-Abweichung zwischen Original und der projezierten Spline ermitteln
        Biggest_Value_Calc(SplinePunkte); //Ergebnis in "Biggest_Value","Biggest_Point"
        //Spline Skallieren
        Scaled_Spline_Points=Barrier_Free_Spline_Points;
        Scale_Spline(SplinePunkte,Scaled_Spline_Points,Bounds_Checked_Spline_Points,Original_Spline_Points,Point_Cord_
        //-------------------------------------------------------------------
        // 3. TEIL: BARRIERE FREI MACHEN
        //Bounds_Checked_Spline_Points=Scaled_Spline_Points; //Wenn das aktiviert ist, wir die spline skalliert
        Make_Spline_Barrier_Free(SplinePunkte,Bounds_Checked_Spline_Points, Barrier_Free_Spline_Points); //Ergebnis i
        //SPLINE KORRIGIEREN-ZEICHENEN
        var new_Points;
        for (cnt=0;cnt<SplinePunkte;cnt++)
        {
          //new_Points=Bounds_Checked_Spline_Points[cnt];//Hier wird entschieden welches PunktArray die Spline bestimn
          new_Points=Scaled_Spline_Points[cnt];
          //new_Points=Barrier_Free_Spline_Points[cnt];//Hier wird entschieden welches PunktArray die Spline bestimmt.
          println("Spline zeichnen:",new_Points);
          OriginalSpline->SetPoint(cnt,new_Points);
        }
        //SPLINE-Updaten
        var SplineNEW;
        SplineName=OriginalSpline->GetName();
        SplineNEW=doc->FindObject(SplineName);
        SplineNEW#ID_BASEOBJECT_GENERATOR_FLAG=False;
        SplineNEW#ID_BASEOBJECT_GENERATOR_FLAG=True;
        //LoftNurbs-Updaten
        var LoftnurbsNEW,Loftname;
        Loftname=("LoftObjekt");
        LoftnurbsNEW=doc->FindObject(Loftname);
        LoftnurbsNEW#ID_BASEOBJECT_GENERATOR_FLAG=False;
        LoftnurbsNEW#ID_BASEOBJECT_GENERATOR_FLAG=True;
        OFF=False; //OFF=True;
      }//if Checked=True ENDE
    }//Ende der Pfade Wählen-Schleife
  }//ENDE ONOFF=TRUE
  else
  {
    if (ONOFF==TRUE)
    {
      println("Kann nicht barrierefrei gemacht werden, da kein LoftObjekt und keine Splines vorhanden sind!!!");
      OFF=False;
    }
  }
}



//-------------------------------
//MAKE_PEDESTRIAN_PATH  |
//-------------------------------
//by Christian Pichlkastner
//Fußgängerweg erzeugen.
//
//Eingang: Path_Punkte, ONOFF, Anzahl_Splines, SplinePunkte
//Ausgang: OFF

//GLOBALE VARIABLEN
var doc;
var BRIDGE_POINTS_ARRAY;
var Path_Points;

//-------------------------------------------------------------------------------------------
//-------------------------------------------------------------------------------------------
main()
{
  doc=GetActiveDocument();
  if(ONOFF==TRUE)
  {
    var Pedestrian_Path_Spline,Pedestrian_Path_Name;
    Pedestrian_Path_Name=("Pedestrian_Path_Spline");
    if(doc->FindObject(Pedestrian_Path_Name)!=True);
    {
      BRIDGE_POINTS_ARRAY = new (array,Anzahl_Splines,SplinePunkte);
      println("Fußgänger_Weg wird generiert");

      //Punkte_Array von der Brückenflächengeompetrie erzeugen
      GENERATE_BRIGDE_GEOMETRY_ARRAY(); //ERGEBNIS: BRIDGE_POINTS_ARRAY[][];->[SplineAnzahl][Anzahl_Splin

      //Pfadpunkte Berechnen
      PATH_POINTS_CALC();

      //Pfad zeichen
```

```
      DRAW_PATH_SPLINE();
    }
  }
  OFF=FALSE;
}

//-----------------------------------
//MAKE_BARRIER_FREE_PATH  |
//-----------------------------------
//by Christian Pichlkastner
//Barrierefeien Weg erzeugen
//
//Eingang: Anzahl_Splines, SplinePunkte, Path_Punkte, ONOFF
//Ausgang: OFF
//GLOBALE VARIABLEN
var doc;
var BRIDGE_POINTS_ARRAY;
var Path_Points;

//-------------------------------------------------------------------------------------------------------
//-------------------------------------------------------------------------------------------------------
main()
{
  doc=GetActiveDocument();
  if(ONOFF==TRUE)
  {
    var Pedestrian_Path_Spline,Pedestrian_Path_Name;
    Pedestrian_Path_Name=("Pedestrian_Path_Spline");
    if(doc->FindObject(Pedestrian_Path_Name)!=True);
    {
      BRIDGE_POINTS_ARRAY = new (array,Anzahl_Splines,SplinePunkte);
      println("Fußgänger_Weg wird generiert");

      //Punkte_Array von der Brückenflächengeompetrie erzeugen
      GENERATE_BRIDGE_GEOMETRY_ARRAY(); //ERGEBNIS: BRIDGE_POINTS_ARRAY[][];->[SplineAnzahl][A

      //Pfadpunkte Berechnen
       PATH_POINTS_CALC();

       //Pfad zeichen
       DRAW_PATH_SPLINE();
    }
  }
  OFF=FALSE;
}



//--------------------------
//PATHFINDER_SCRIPT  |
//--------------------------
//by Christian Pichlkastner
//
//Diese Skript erzeugt eine Weg-Spline über eine Oberfläche
//
//Eingang: Anzahl_Splines, SplinePunkte, Grade_Factor, ONOFF
//Ausgang: OFF

//GLOBALE VARIABLEN
var doc; // Damit Objekte in aktueller Szene benützt/gesucht werden können
var BRIDGE_POINTS_ARRAY;
var Dist_current_Points;
var Path_Points,PathPointNum;
var path_cnt;

//Kürzester Weg von Punkt A zu Punkt B suchen -> Dijkstra Algorithmus Methode
SHORTEST_WAY_POINT_A_TO_B(Point_A_SplineNum,Point_A_PointNum,Point_B_SplineNum,Point_B_Poin
{
  var Min_Path_Length;
  var Node_Count=SplinePunkte*Anzahl_Splines;
  var PreviousNode_Name=new(array,Anzahl_Splines,SplinePunkte);
  var Current_Spline,Current_Point;
  var Previous_Spline,Previous_Point;
  var a,cnt,Neighbours,i,Schleifnr=1;
  var Neighbour_SplineNum=new(array,8); //Es sind maximal 8 NachbarPunkte möglich
  var Neighbour_PointNum=new(array,8); //Es sind maximal 8 NachbarPunkte möglich
  var CurrentNode_Coordinates;
  var PreviousNode_Coordinates;
  var New_Link_Distance=0;

  //jeder Knoten hat die 4 Daten:
  //KnotenNamen;Permantent/NonPermantent;Path_Distance;Before_Knot;
  var NODEDATA=new(array,Anzahl_Splines,SplinePunkte,4); //SplineNummer/PunktNummer/Daten


  //PART I - Initialisieren | für alle Punkt-Startwerte vergeben
  for(a=0;a<Anzahl_Splines;a++)
  {
    for(cnt=0;cnt<SplinePunkte;cnt++)
    {
      NODEDATA[a][cnt][0]="NonPermantent";
      NODEDATA[a][cnt][1]=-1.0; //Null
      //NODEDATA[a][cnt][2]=Before_Knot_SplineNum;
      //NODEDATA[a][cnt][3]=Before_Knot_PointNum;
    }
  }
  //StartPunkt
  Current_Spline=Point_A_SplineNum;
  Current_Point=Point_A_PointNum;
  NODEDATA[Current_Spline][Current_Point][0]="PERMANENT-FIXED"; //Start als FixPunkt festlegen.
  NODEDATA[Current_Spline][Current_Point][1]=0; //Start als FixPunkt festlegen.
  NODEDATA[Current_Spline][Current_Point][2]=Current_Spline; //Start als FixPunkt festlegen.
  NODEDATA[Current_Spline][Current_Point][3]=Current_Point; //Start als FixPunkt festlegen.
  //-------------Distanz auf einen großen wert setzen
  var Point_A_Coordinates,Point_B_Coordinates;
  var Smallest_Distance,Distance_Reset;

  Point_A_Coordinates=BRIDGE_POINTS_ARRAY[Current_Spline][Current_Point];
  Point_B_Coordinates=BRIDGE_POINTS_ARRAY[Point_B_SplineNum][Point_B_PointNum];

  //Dist_current_Points - Große vergleichs Distanz berechnen
  XYZ_Dist_2_Points_Calc(Point_B_Coordinates,Point_A_Coordinates);
  Smallest_Distance=abs(Dist_current_Points); //Startwert -> sollte großer wert sein, der sicher unterboten wird.
  // Distance_Reset=Smallest_Distance*Point_A_SplineNum*100*Grade_Faktor;//10*Smallest_Distance;
  //FEHLERQUELLE - Wenn der Wert zu klein ist, dann kann eine EndlosSchleifeEntstehen
  Distance_Reset=Smallest_Distance; //*Node_Count; //10000000000000;
  println("StartPunkt:",Point_A_SplineNum,"|",Point_A_PointNum," ZielPunkt:",Point_B_SplineNum,"|",Point_B_PointNum);
  println("Die Distanz zwischen den 2 Punkten beträgt: ",Smallest_Distance);

  //PART II - Bis Zielpunkt erreicht und Permanent Fixed ist.
  while (Current_Spline!=Point_B_SplineNum | Current_Point!=Point_B_PointNum)
  {
    //println("-----------------------------------------------");
    println("IN SCHLEIFE Nr.:",Schleifnr);
    Neighbours=0;
    Smallest_Distance =Distance_Reset*(Schleifnr+1);

    //größte Distanz finden
    for(a=0;a<Anzahl_Splines;a++)
    {
      for(cnt=0;cnt<SplinePunkte;cnt++)
      {
        if(Smallest_Distance <= NODEDATA[a][cnt][1])
        {
          Smallest_Distance=NODEDATA[a][cnt][1]*NODEDATA[a][cnt][1]*120*Node_Count+Distance_Reset;
        }
      }
    }
    println("ResetDistanz:",Smallest_Distance);
    //den Anfagsvergleichswert immer sehr hoch machen
    //1. Anzahl der möglichen NachbarPunkte berechnen.

    //Eckpunkte
    if(Current_Spline == 0 && Current_Point == 0)
    {
      Neighbours =3;
      Neighbour_SplineNum[0]=Current_Spline;
      Neighbour_PointNum[0]=Current_Point+1;
      Neighbour_SplineNum[1]=Current_Spline+1;
      Neighbour_PointNum[1]=Current_Point+1;
      Neighbour_SplineNum[2]=Current_Spline+1;
      Neighbour_PointNum[2]=Current_Point;
    }
    if(Current_Spline == Anzahl_Splines-1 && Current_Point == 0)
    {
      Neighbours =3;
      Neighbour_SplineNum[0]=Current_Spline-1;
      Neighbour_PointNum[0]=Current_Point;
      Neighbour_SplineNum[1]=Current_Spline-1;
      Neighbour_PointNum[1]=Current_Point+1;
      Neighbour_SplineNum[2]=Current_Spline;
      Neighbour_PointNum[2]=Current_Point+1;
    }
    if(Current_Spline == 0 && Current_Point == SplinePunkte-1)
    {
      Neighbours =3;
      Neighbour_SplineNum[0]=Current_Spline+1;
      Neighbour_PointNum[0]=Current_Point;
      Neighbour_SplineNum[1]=Current_Spline+1;
```

```
    Neighbour_PointNum[1]=Current_Point-1;
    Neighbour_SplineNum[2]=Current_Spline;
    Neighbour_PointNum[2]=Current_Point-1;
  }
  if(Current_Spline == Anzahl_Splines-1 && Current_Point == SplinePunkte-1)
  {
    Neighbours =3;
    Neighbour_SplineNum[0]=Current_Spline;
    Neighbour_PointNum[0]=Current_Point-1;
    Neighbour_SplineNum[1]=Current_Spline-1;
    Neighbour_PointNum[1]=Current_Point-1;
    Neighbour_SplineNum[2]=Current_Spline-1;
    Neighbour_PointNum[2]=Current_Point;
  }
  //Randbereiche 5
  if(Current_Spline == 0 && Current_Point > 0 && Current_Point < SplinePunkte-1)
  {
    Neighbours =5;
    Neighbour_SplineNum[0]=Current_Spline;
    Neighbour_PointNum[0]=Current_Point+1;
    Neighbour_SplineNum[1]=Current_Spline+1;
    Neighbour_PointNum[1]=Current_Point+1;
    Neighbour_SplineNum[2]=Current_Spline+1;
    Neighbour_PointNum[2]=Current_Point;
    Neighbour_SplineNum[3]=Current_Spline+1;
    Neighbour_PointNum[3]=Current_Point-1;
    Neighbour_SplineNum[4]=Current_Spline;
    Neighbour_PointNum[4]=Current_Point-1;
  }
  if(Current_Spline == Anzahl_Splines-1 && Current_Point > 0 && Current_Point < SplinePunkte-1)
  {
    Neighbours =5;
    Neighbour_SplineNum[0]=Current_Spline-1;
    Neighbour_PointNum[0]=Current_Point;
    Neighbour_SplineNum[1]=Current_Spline-1;
    Neighbour_PointNum[1]=Current_Point+1;
    Neighbour_SplineNum[2]=Current_Spline;
    Neighbour_PointNum[2]=Current_Point+1;
    Neighbour_SplineNum[3]=Current_Spline;
    Neighbour_PointNum[3]=Current_Point-1;
    Neighbour_SplineNum[4]=Current_Spline-1;
    Neighbour_PointNum[4]=Current_Point-1;
  }
  if(Current_Spline > 0 && Current_Spline < Anzahl_Splines-1 && Current_Point == 0)
  {
    Neighbours =5;
    Neighbour_SplineNum[0]=Current_Spline-1;
    Neighbour_PointNum[0]=Current_Point;
    Neighbour_SplineNum[1]=Current_Spline-1;
    Neighbour_PointNum[1]=Current_Point+1;
    Neighbour_SplineNum[2]=Current_Spline;
    Neighbour_PointNum[2]=Current_Point+1;
    Neighbour_SplineNum[3]=Current_Spline+1;
    Neighbour_PointNum[3]=Current_Point+1;
    Neighbour_SplineNum[4]=Current_Spline+1;
    Neighbour_PointNum[4]=Current_Point;
  }
  if(Current_Spline > 0 && Current_Spline < Anzahl_Splines-1 && Current_Point == SplinePunkte-1)
  {
    Neighbours =5;
    Neighbour_SplineNum[0]=Current_Spline-1;
    Neighbour_PointNum[0]=Current_Point;
    Neighbour_SplineNum[1]=Current_Spline+1;
    Neighbour_PointNum[1]=Current_Point;
    Neighbour_SplineNum[2]=Current_Spline+1;
    Neighbour_PointNum[2]=Current_Point-1;
    Neighbour_SplineNum[3]=Current_Spline;
    Neighbour_PointNum[3]=Current_Point-1;
    Neighbour_SplineNum[4]=Current_Spline-1;
    Neighbour_PointNum[4]=Current_Point-1;
  }

  //Zwischenbereich 8 //Anhand dieser Liste müssten sich eigentlich die if-Bedingungen davor kürzen lasser
  if(Current_Spline > 0 && Current_Spline < Anzahl_Splines-1 && Current_Point > 0 && Current_Point < Spl
  {
    Neighbours =8;
    Neighbour_SplineNum[0]=Current_Spline-1;
    Neighbour_PointNum[0]=Current_Point;
    Neighbour_SplineNum[1]=Current_Spline-1;
    Neighbour_PointNum[1]=Current_Point+1;
    Neighbour_SplineNum[2]=Current_Spline;
    Neighbour_PointNum[2]=Current_Point+1;
    Neighbour_SplineNum[3]=Current_Spline+1;
    Neighbour_PointNum[3]=Current_Point+1;
    Neighbour_SplineNum[4]=Current_Spline+1;
    Neighbour_PointNum[4]=Current_Point;
    Neighbour_SplineNum[5]=Current_Spline+1;
    Neighbour_PointNum[5]=Current_Point-1;
    Neighbour_SplineNum[6]=Current_Spline;
    Neighbour_PointNum[6]=Current_Point-1;
    Neighbour_SplineNum[7]=Current_Spline-1;
    Neighbour_PointNum[7]=Current_Point-1;
  }
  /*
  println("Nach der Nachbarzuordnung.");
  println("Der Punkt ",Current_Spline,"|",Current_Point," hat ",Neighbours," Nachbarpunkte.");
  */
  Previous_Spline=Current_Spline;
  Previous_Point=Current_Point;
  PreviousNode_Coordinates=BRIDGE_POINTS_ARRAY[Current_Spline][Current_Point];
  //Die Differenz zwischen dem Vorgänger_Punkt und seinen NachbarPunkten berechnen
  println("DIFFERENZ zwischen NACHBARPUNKT UND VORGÄNGERPUNKTEN berechnen");
  for(a=0;a<Neighbours;a++)
  {
    Current_Spline=Neighbour_SplineNum[a]; //Die Nachbarpunkt in Glob.Position converiert
    Current_Point=Neighbour_PointNum[a]; //Die Nachbarpunkt in Glob.Position converiert

    if(NODEDATA[Current_Spline][Current_Point][0]!="PERMANENT-FIXED")
    {
      CurrentNode_Coordinates=BRIDGE_POINTS_ARRAY[Current_Spline][Current_Point];
      //Dist_current_Points - Gewichtung der Strecken (Graphen)
      XYZ_Dist_2_Points_Calc(CurrentNode_Coordinates,PreviousNode_Coordinates);
      New_Link_Distance=NODEDATA[Previous_Spline][Previous_Point][1]+Dist_current_Points;
       /*
       println(" ");
       println("Distanz zwischen Current_Node:",Current_Spline,"|",Current_Point," und Pervious_Node: ",Previous_Spline,
",Dist_current_Points);
       println("Aktuelle Distanz zum Anfangspunkt A = ",New_Link_Distance);
       */
       //Diesen Punkte veränderen wenn Dist_current_Points < bisheriger Distanzwert in dem Punktdatenfeld.
       if (NODEDATA[Current_Spline][Current_Point][1]==-1.0 || New_Link_Distance <= NODEDATA[Current_Spline][Curr
       {
         /*
         println("Alter Distanz-Wert:",NODEDATA[Current_Spline][Current_Point][1]);
         println("Neuer Distanz-Wert:",New_Link_Distance);
         println("Dieser Nachbar wurde mit einem kleinerem Distanzwert überschrieben.");
         */
         NODEDATA[Current_Spline][Current_Point][1]= New_Link_Distance;
         //Eintragen von welchem Punkt aus gemessen wurde //Da der momentan kürzeste Weg über diesen Punkt führt.
         NODEDATA[Current_Spline][Current_Point][2]=Previous_Spline;
         NODEDATA[Current_Spline][Current_Point][3]=Previous_Point;
         //println("VORGÄNGER wird auf ",Previous_Spline,"|",Previous_Point," gesetzt");println(" ");
       }
       //else ("Der aktuelle Distanz-Wert wird nicht eingetragen, da dieser größer als der alte Wert ist.");
     }
     else //println("Der Punkt ",Current_Spline,"|",Current_Point,"kann nicht verwendet werden!!!!!!!");
    }

    //PART III
    //Welcher Punkt aller bisherigen Punkte, der noch nicht als Permanent-Fixed markiert ist, hat die kleinste Distanz?
    //alle Punktwerte durchchecken.
    println("Welcher Punkt aller bisher bearbeiteten Punkte, der noch nicht als Permanent-Fixed markiert ist, hat die kleins
    var currentDistance_Value;
    var i,PointNumber;
    for(a=0;a<Anzahl_Splines;a++)
    {
      for(cnt=0;cnt<SplinePunkte;cnt++)
      {
        //println("Check Point: ",a,"|",cnt,"");
        //Wenn Notedate ist nicht Permanent-Fixed oder die Distanz nicht -1.0- oder der Knoten nicht der Startknoten dann
        if(NODEDATA[a][cnt][0]!="PERMANENT-FIXED" && NODEDATA[a][cnt][1]!=-1.0)
        {
          currentDistance_Value=NODEDATA[a][cnt][1];
          println("Punkt ",a,"|",cnt," ist nicht PERMANENT-FIXED und enthält als Distanz nicht den Wert -1.0 sondern: ",NODE
          println("Die Momentan kürzeste Distanze die in einem Punkt eingetragen ist lautet: ",Smallest_Distance);
          if(Smallest_Distance >= currentDistance_Value)
          {
            Current_Spline=a; //Aktuelle Spline mit kleinster Distanz
            Current_Point=cnt;//Aktueller Punkt mit kleinster Distanz
            println(currentDistance_Value," erstetzt größern alten Wert ", Smallest_Distance);
           Smallest_Distance= currentDistance_Value;
          }
        }
        // else println("Punkt ",a,"|",cnt," ist PERMANENT_FIXED oder enthält Distanz-Wert:-1.0 oder ist Startpunkt");
      }
    }
    println(" ");
    println("Punkt ",Current_Spline,"|",Current_Point," wird zu aktuellem Punkt mit einem Betrag von: ",Smallest_Distance,
```

```
      //Dieser Knoten wird als definitiv markiert
      NODEDATA[Current_Spline][Current_Point][0]="PERMANENT-FIXED";
      Schleifnr=Schleifnr+1;
      //NOTBREMSE BEI wahrscheinlichem FEHLER - WhileSchleife beenden
      if(Schleifnr>500)
      {
        println("MÖGLICHER FEHLER ODER ZU KOMPLEXER PFAD!!!");
        Current_Spline=Point_B_SplineNum;
        Current_Point=Point_B_PointNum;
        break;
      }
    }  //SCHLEIFE ENDE BIS ENDPUNKT=PERMANENT-FIXED!
   println("Letzter aktueller außerhalb der Scheife Punkt war ",Current_Spline,"|",Current_Point,"");
   /*
   //Überprüfung: Ausgabe der Punktdaten
   for(a=0;a<Anzahl_Splines;a++)
   {
     for(cnt=0;cnt<SplinePunkte;cnt++)
     {
       println("Knoten ",a,"|",cnt," Status: ",NODEDATA[a][cnt][0]);
       println("Knoten ",a,"|",cnt," Distanz: ",NODEDATA[a][cnt][1]);
       println("Knoten ",a,"|",cnt," vorhergehende Knoten: ",NODEDATA[a][cnt][2],"|",NODEDATA[a][cnt][3]);
       println(" ");
       Node_Count=Node_Count+1;
     }
   }//Überprüfung: Ausgabe der Punktdaten ENDE
   */

   //PART 4 - Pfad bis zu A zurück verfolgen.
   PathPointNum=0;
   while (Current_Spline!=Point_A_SplineNum | Current_Point!=Point_A_PointNum)
   {
     Current_Spline=NODEDATA[Current_Spline][Current_Point][2];
     Current_Point=NODEDATA[Current_Spline][Current_Point][3];
     PathPointNum=PathPointNum+1;
   }
   PathPointNum=PathPointNum+1;
   println("Es wurden ",PathPointNum," Punkte als Pfad ermittelt!");
   Path_Points=new (array,PathPointNum);
   Current_Spline=Point_B_SplineNum;
   Current_Point=Point_B_PointNum;
   Min_Path_Length=NODEDATA[Current_Spline][Current_Point][1];
   for(i=PathPointNum-1;i>=0;i--)
   {
     Path_Points[i]=BRIDGE_POINTS_ARRAY[Current_Spline][Current_Point];
     /*
     println("Pfad ",i,". Punkt:",Current_Spline,"|",Current_Point);
     println("Pfad ",i,". Koordinaten:",Path_Points[i]);
     */
     Current_Spline=NODEDATA[Current_Spline][Current_Point][2];
     Current_Point=NODEDATA[Current_Spline][Current_Point][3];
   }
   println("Die kürzeste Strecke beträgt: ",Min_Path_Length);
   return Path_Points;
}

//-------------------------------------------------------------------------------------------
//-------------------------------------------------------------------------------------------
main()
{
  doc=GetActiveDocument();
  if (ONOFF==TRUE && doc->FindObject("LoftObjekt"))
  {
    var a; // Zählvariablen
    var Point_A_SplineNum,Point_A_PointNum,Point_B_SplineNum,Point_B_PointNum;
    BRIDGE_POINTS_ARRAY = new (array,Anzahl_Splines,SplinePunkte);

    println("START PATHFINDER!!!");

    //Punkte_Array von der Brückenflächengeompetrie erzeugen
    GENERATE_BRIDE_GEOMETRY_ARRAY(); //ERGEBNIS: BRIDGE_POINTS_ARRAY[][];->[SplineAnzahl][Anz

    //Anfangs und Endpunkt für den kürzesten Weg definieren
    Point_A_SplineNum=int(Anzahl_Splines/2);
    Point_A_PointNum=0;
    Point_B_SplineNum=int(Anzahl_Splines/2);
    Point_B_PointNum=SplinePunkte-1;
    println("Anfangspunkt: ",Point_A_SplineNum,"|",Point_A_PointNum);
    println("Endpunkt: ",Point_B_SplineNum,"|",Point_B_PointNum);

    //Kürzester Weg von Punkt A zu Punkt B suchen -> Dijkstra Algorithmus Methode
    SHORTEST_WAY_POINT_A_TO_B(Point_A_SplineNum,Point_A_PointNum,Point_B_SplineNum,Point_B_P

    //Punkte_Array von der Brückenflächengeompetrie erzeugen
    DRAW_PATH_SPLINE();
  }
  else
  {
    if(ONOFF==TRUE)
    {
      println("Es kann kein Pfad bzw. Weg generiert werden, da keine Oberfläche vorhanden ist!!!");
    }
  }
  OFF==FALSE;
}


//-----------------------------------
//MAKE_ATTRACTOR_SPHERES  |
//-----------------------------------
//by Christian Pichlkastner
//
// Erzeugt Kugelobjekte. Wenn der Zufallsgenerator aktiviert ist, dan werden Kugeln
// mit unterschiedlichen Radien und unterschiedlichen Positionen erzeugt.
//
//Eingang: ON,Num_of_Attraktors,Random_ON
//Ausgang: OFF

//GLOBALE VARIABLEN
var doc;
var Dist_current_Points;
var Approx_Bridge_Middle_Point,MiddleSpline;

//MAKE_SPHERES
MAKE_SPHERES()
{
  var Parent=doc->FindObject("ATTRACTORS");
  var Attractor_Name,Attractor_Radius,Attraktor_Position;
  var Zufalls_Zahl_1;
  var X_Zufall,Y_Zufall,Z_Zufall,Attractor_Size_Zufall;
  var Field_Radius,Spreading_Field_Size,Y_Dist;
  var Zufall;
  var i,half_Attractors_Num;
  var SplineObjekt;
  var SplinePunkte;
  var SplineName=("OriginalSpline"+tostring(MiddleSpline));
  var SplinePunkt_0=SplineObjekt->GetPoint(0);
  SplineObjekt=doc->FindObject(SplineName);

  Zufall=new(Random);
  Zufall->Init(time());

  XYZ_Dist_2_Points_Calc(Approx_Bridge_Middle_Point,SplinePunkt_0);
  Field_Radius=Dist_current_Points;
  Spreading_Field_Size=Field_Radius*8;
  for(i=0;i<Num_of_Attraktors;i++)
  {
   if(Random_ON==True)
   {
     //Zufällige Positive oder Negative Attrakroen vergeben;
     Zufalls_Zahl_1=Zufall->Get01();
     if(Zufalls_Zahl_1<0.5) Attractor_Name=("NEGATIVE_ATTRACTOR_Sphere"+tostring(i));
     if(Zufalls_Zahl_1)=0.5) Attractor_Name=("POSITIVE_ATTRACTOR_Sphere"+tostring(i));
     Attractor_Size_Zufall=Zufall->Get01();
     Attractor_Radius=Field_Radius*Attractor_Size_Zufall;

     X_Zufall=Zufall->Get01();
     Y_Zufall=Zufall->Get01();
     Z_Zufall=Zufall->Get01();
     Attraktor_Position=Approx_Bridge_Middle_Point;
     Attraktor_Position.x=Attraktor_Position.x-Spreading_Field_Size/2+Spreading_Field_Size*X_Zufall;
     Attraktor_Position.y=Attraktor_Position.y-Spreading_Field_Size/6+Spreading_Field_Size/3*Y_Zufall;
     Attraktor_Position.z=Attraktor_Position.z-Spreading_Field_Size/2+Spreading_Field_Size*Z_Zufall;
   }
   else
   {
     Attractor_Radius=(2*Field_Radius)/Num_of_Attraktors;
     Attraktor_Position=Approx_Bridge_Middle_Point;
     half_Attractors_Num=int(Num_of_Attraktors/2);
     if(i<half_Attractors_Num) Attractor_Name=("NEGATIVE_ATTRACTOR_Sphere"+tostring(i));
     if(i>=half_Attractors_Num) Attractor_Name=("POSITIVE_ATTRACTOR_Sphere"+tostring(i));
   }
   MAKE_SPHERE(Attractor_Radius,Attractor_Name,Attraktor_Position,Parent);
  }
}

//-------------------------------------------------------------------------------------------
```

```
//-------------------------------------------------------------------------------------------
main()
{
  if(ON==TRUE) //Warum ich die 2 Startbedingungen ( if(ON==TRUE && doc->FindObject("LoftObjekt")) nicht in
Modulen ist mir nicht klar.
  {
    doc=GetActiveDocument();
    if(doc->FindObject("LoftObjekt"))
    {
      println(" ");
      println("ATTRAKTOREN ERZEUGEN");

      //ATTRACTORS - Ordner ERSTELLEN
      var Attrators_NullObject;
      if(!doc->FindObject("ATTRACTORS"))
      {
        var ATTRAKTORS_Folder_Name=("ATTRACTORS");
        Attrators_NullObject=new(NullObject);
        Attrators_NullObject->SetName(ATTRAKTORS_Folder_Name);
        doc->InsertObject(Attrators_NullObject,Null,Null);
      }
      //Anzahl_Splines feststellen;
      var i,Anzahl_Splines=1;
      var SplineName;
      for(i=1;i<=Anzahl_Splines;i++)
      {
        SplineName=("OriginalSpline"+tostring(i));
        if(doc->FindObject(SplineName))Anzahl_Splines=i+1;;
        if(!doc->FindObject(SplineName))break;
      }
      Anzahl_Splines=Anzahl_Splines-1;
      println("Es gibt ",Anzahl_Splines," Splines.");
      //Spline im Mittleren Bereich auswählen
      MiddleSpline=int(Anzahl_Splines/2);
      println("Mittlere Spline_Nr: ",MiddleSpline);
      //Mittlere Spline als Objekt auslesn und Mittleren SplinePunkt bestimmen
      var SplineObjekt;
      var SplinePunkte;
      var Middle_SplinePoint;
      SplineName=("OriginalSpline"+tostring(MiddleSpline));
      SplineObjekt=doc->FindObject(SplineName);
      SplinePunkte=SplineObjekt->GetPointCount();
      Middle_SplinePoint=int(SplinePunkte/2);
      Approx_Bridge_Middle_Point=SplineObjekt->GetPoint(Middle_SplinePoint);
      println("Der Brückenmittelpunkt befindet sich an der Stelle: ",Approx_Bridge_Middle_Point);

      MAKE_SPHERES();
    }// if(doc->FindObject("LoftObjekt) ENDE
  }// if(ON==TRUE) ENDE
  OFF=FALSE;
}



//-----------------------------------------
//KNEAD_SPLINES_TO_ATTRATORES  |
//-----------------------------------------
//by Christian Pichlkastner
//
//Die Kontrollpunkte einer ausgewählte Spline werden über den Einfluss der
//Attraktorenkugeln verformt.
//
//Eingang: ON, SPLINE
//Ausgang: OFF
//GLOBALE VARIABLEN
var doc,SplinePunkte;
var Dist_current_Points;
MOVE_SPLINE_POINTS()
{
  //Alle SplinePunkte bis auf die Randpunkte durchgehen
  var i,a;
  var current_Spline_Point;
  var NEW_X=0.0,NEW_Y=0.0,NEW_Z=0.0,New_Point_Position=vector(0,0,0);
  var Attraktor_Num=0,Attractor_Object, Negative_Attraktor_Name,Positive_Attraktor_Name,Attraktor_Radius,
  var Attraktor_Grav_X,Attraktor_Grav_Y,Attraktor_Grav_Z,X_Dist,Y_Dist,Z_Dist;
  var Prefix;
  for(i=1;i<SplinePunkte-1;i++)
  {
    //Festellen welchen Punkten der aktuelle SplinePunkt am nächsten ist
    current_Spline_Point=SPLINE->GetPoint(i);
    //Entfernung zu allen Attraktoren messen und Stärke berücksichtigen
    for(a=0;a<=Attraktor_Num;a++)
    {
      Negative_Attraktor_Name=("NEGATIVE_ATTRACTOR_Sphere"+tostring(a));
      Positive_Attraktor_Name=("POSITIVE_ATTRACTOR_Sphere"+tostring(a));
      if(doc->FindObject(Negative_Attraktor_Name) || doc->FindObject(Positive_Attraktor_Name)) Attraktor_Num=a+1;
      else break;
    }
    Attraktor_Num=Attraktor_Num;
    Attraktor_Grav_X=new(array,Attraktor_Num+1);
    Attraktor_Grav_Y=new(array,Attraktor_Num+1);
    Attraktor_Grav_Z=new(array,Attraktor_Num+1);
    for(a=0;a<Attraktor_Num;a++)
    {
      Negative_Attraktor_Name=("NEGATIVE_ATTRACTOR_Sphere"+tostring(a));
      Positive_Attraktor_Name=("POSITIVE_ATTRACTOR_Sphere"+tostring(a));
      if(doc->FindObject(Negative_Attraktor_Name))
      {
        Attractor_Object=doc->FindObject(Negative_Attraktor_Name);
        Prefix=-1;
      }
      if(doc->FindObject(Positive_Attraktor_Name))
      {
        Attractor_Object=doc->FindObject(Positive_Attraktor_Name);
        Prefix=1;
      }
      Attractor_Position=Attractor_Object->GetPosition();
      Attraktor_Radius=Attractor_Object#PRIM_SPHERE_RAD;
      //Kraft nimmt mit der Entfernung zur 3. ab.
      X_Dist=current_Spline_Point.x-Attractor_Position.x;
      Attraktor_Grav_X[a]=(X_Dist/pow(X_Dist,2)*Attraktor_Radius)*Prefix;
      Y_Dist=current_Spline_Point.y-Attractor_Position.y;
      Attraktor_Grav_Y[a]=(Y_Dist/pow(Y_Dist,2)*Attraktor_Radius)*Prefix;
      Z_Dist=current_Spline_Point.z-Attractor_Position.z;
      Attraktor_Grav_Z[a]=(Z_Dist/pow(Z_Dist,2)*Attraktor_Radius)*Prefix;
      NEW_X=NEW_X+Attraktor_Grav_X[a];
      NEW_Y=NEW_Y+Attraktor_Grav_Y[a];
      NEW_Z=NEW_Z+Attraktor_Grav_Z[a];
    }
    New_Point_Position.x=current_Spline_Point.x+NEW_X;
    New_Point_Position.y=current_Spline_Point.y+NEW_Y;
    New_Point_Position.z=current_Spline_Point.z+NEW_Z;
    println(i,"Point Old_Position: ",current_Spline_Point);
    println(i,"Point New_Position: ",New_Point_Position);
    SPLINE->SetPoint(i,New_Point_Position);
  }
  println("Es gibt ",Attraktor_Num," Attraktoren");
}

//-------------------------------------------------------------------------------------------
//-------------------------------------------------------------------------------------------
main()
{
  if(ON==TRUE) //Warum ich die 2 Startbedingungen ( if(ON==TRUE && doc->FindObject("LoftObjekt")) nicht in eine Zeile
Modulen ist mir nicht klar.
  {
    doc=GetActiveDocument();
    if(SPLINE!=NULL)
    {
      println(" ");
      println("SPLINE AN ATTRAKTOREN AUSRICHTEN");
      SplinePunkte=SPLINE->GetPointCount();
      MOVE_SPLINE_POINTS();
      SPLINE->Message(MSG_UPDATE);
    }
    else println("Es ist kein Spline_Objekt bzw. Weg_Objekt zum Ausrichten vorhanden!!!");
  }  // if(ON==TRUE) ENDE
  OFF=FALSE;
}



//--------------------------------
//KNEAD_TO_PATHS_SCRIPT  |
//--------------------------------
//by Christian Pichlkastner
//
//Diese Skript verformt die Brückenoberfläche nach zwei Pfaden
//
//Eingang:  Anzahl_Splines, SplinePunkte, Pedestrian_Width, Biker_Width, WheelChair_Width, Scale_Factor, ONOFF
//Ausgang: OFF

//GLOBALE VARIABLEN
var doc; // Damit Objekte in aktueller Szene benützt/gesucht werden können
var BRIDGE_POINTS_ARRAY;
var Dist_current_Points;
var Path_Points,PathPointNum;
var path_cnt;
```

```
var Path_Points_Cnt;
var Nearst_Spline,Nearst_Point,Reset_Distance;

//NEAREST BRIDGPOINT TO PATHPOINT
DEFORM_BASICSURFACE_TO_PATHSPLINES(Path_Object)
{
  var Path_1_Object,Path_2_Object;
  var Path_1_Name,Path_2_Name;
  var Path_1_Position_Point,Path_2_Position_Point;
  var Section_Percent_Step,Percent_Position,Percent_Position_Interpolated,Percent_Position_Path_1,Percent_
  var Path_1_Lenght=0.0,Path_2_Lenght=0.0,Path_Middle_Lenght=0.0;
  var i,a,cnt,c,Points;
  var Distance_12=0;
  var Biggest_Distance=0,Dist_X,Dist_Y,Dist_Z;
  var X_Degree,Y_Degree,Z_Degree;
  var Path_Cnt=0,BetweenPaths_Points=0.0;
  var Mid_Point,HalfSplines=0.0,Scale=0.0;
  var Path_1_Pos_Temp1, Path_2_Pos_Temp1, Path_1_Pos_Temp2, Path_2_Pos_Temp2;
  var tolerance;
  var Path_1_Full_Length=0,Path_2_Full_Length=0;
  var X_Degree_L,Y_Degree_L,Z_Degree_L,X_Degree_R,Y_Degree_R,Z_Degree_R;
  var Left_Half_Width,Right_Half_Width;
  Path_1_Name=("Pedestrian_Path_Spline");
  Path_2_Name=("Barrier_Free_Path_Spline");

  //PfadObjekte erstellen, Anzahl ermittlen, Gesamtlänge ermitteln
  if(doc->FindObject(Path_1_Name))
  {
    Path_1_Object=doc->FindObject(Path_1_Name);
    Path_1_Object->InitLength(0);
    Path_1_Full_Length=Path_1_Object->GetLength();
    Path_Cnt=Path_Cnt+1;
  }
  if(doc->FindObject(Path_2_Name))
  {
    Path_2_Object=doc->FindObject(Path_2_Name);
    Path_2_Object->InitLength(0);
    Path_2_Full_Length=Path_2_Object->GetLength();
    Path_Cnt=Path_Cnt+1;
  }
  println("Path_1_Full_Length: ",Path_1_Full_Length);
  println("Path_1_Full_Length/SplinePunkte: ",Path_1_Full_Length/SplinePunkte);
  println("Path_2_Full_Length: ",Path_2_Full_Length);
  println("Path_2_Full_Length/SplinePunkte: ",Path_2_Full_Length/SplinePunkte);
  Section_Percent_Step=1.0/(SplinePunkte-1); //Alle X Prozent wird geschnitten.
  Percent_Position=0.0;
  Percent_Position_Path_1=0.0;
  Percent_Position_Path_2=0.0;

  //SCHNITTSCHLEIFE BEGINNEN - Schnitt-Anzahl = SplinePunkte-Anzahl
  for(i=0;i<SplinePunkte;i++)
  {
    println("SCHNITT ",i+1);
    //ProzentPositionen auf Durchschnittsprozentposition korrigieren
    Percent_Position_Interpolated=0;
    Path_1_Lenght=0.0;
    Path_2_Lenght=0.0;
    //-------------------------------------------------------KORREKTURTEIL ANFANG-----------------------------
    for(c=0;c<i;c++)
    {
      //println("PROZENTPOSITIONEN WERDEN AUF DURCHSCNITTSWERT KORRIGIERT");
      //println("Distanz zwischen 1. Punkt bei ",Percent_Position_Interpolated,"% und dem 2.Punkt bei ",Percent_
berechnen!");
      //Wikipedia Traindriver Beispiel: var BodyPos = PathObj->GetMg()->GetMulP(PathObj->GetSplinePoint(Patl
      //Path_1_Pos_Temp1=Path_1_Object->GetSplinePoint(Path_1_Object->UniformToNatural(Percent_Positior
      if(doc->FindObject(Path_1_Name)) Path_1_Pos_Temp1=Path_1_Object->GetSplinePoint(Path_1_Object->U
//println("Pedestrian_Path_Spline gefunden");
      if(doc->FindObject(Path_2_Name)) Path_2_Pos_Temp1=Path_2_Object->GetSplinePoint(Path_2_Object->
//println("Barrier_Free_Path_Spline gefunden");
      Percent_Position_Interpolated=Percent_Position_Interpolated+Section_Percent_Step;

      if(doc->FindObject(Path_1_Name)) Path_1_Pos_Temp2=Path_1_Object->GetSplinePoint(Path_1_Object->U
//println("Pedestrian_Path_Spline gefunden");
      if(doc->FindObject(Path_2_Name)) Path_2_Pos_Temp2=Path_2_Object->GetSplinePoint(Path_2_Object->
//println("Barrier_Free_Path_Spline gefunden");
      if(doc->FindObject(Path_1_Name)) XYZ_Dist_2_Points_Calc(Path_1_Pos_Temp1,Path_1_Pos_Temp2); Pat
      if(doc->FindObject(Path_2_Name)) XYZ_Dist_2_Points_Calc(Path_2_Pos_Temp1,Path_2_Pos_Temp2); P
    }
    //println("Pfadlänge1: ",Path_1_Lenght); //Path_1_Lenght ist die Weglänge vom Ausgangspunkt bis bis zum a
    //println("Pfadlänge2: ",Path_2_Lenght);
    //Wenn nicht 1.Punkt dann
    if(i>0)
    {
      Path_Middle_Lenght=(Path_1_Lenght+Path_2_Lenght)/Path_Cnt; //Mittlere Pfadlänge
      println("Mittlere Pfadlänge: ",Path_Middle_Lenght);
      if(doc->FindObject(Path_1_Name)) Percent_Position_Path_1=Path_Middle_Lenght/Path_1_Lenght;
      if(doc->FindObject(Path_2_Name)) Percent_Position_Path_2=Path_Middle_Lenght/Path_2_Lenght;
      //println("Mittlerer Pfad/Path1_Lenght: ",Percent_Position_Path_1);
      //println("Mittlerer Pfad/Path2_Lenght: ",Percent_Position_Path_2);
      if(doc->FindObject(Path_1_Name)) Percent_Position_Path_1=(Path_Middle_Lenght/Path_1_Lenght)*Percent_Position_Interpolated;
      if(doc->FindObject(Path_2_Name)) Percent_Position_Path_2=(Path_Middle_Lenght/Path_2_Lenght)*Percent_Position_Interpolatec
    }
    /*
    println("MOMENTANE POSITION: ",Percent_Position_Interpolated);
    println("Pfad1_Position: ",Percent_Position_Path_1);
    println("Pfad2_Position: ",Percent_Position_Path_2);
    */

    //-------------------------------------------------------KORREKTURTEIL ENDE-------------------------------------------------------
     //Den vorhergehenden Korrekturteil kann man ignorieren und die originalen Prozentwerte verwenden.
     //Beide Pfade werden eben nicht angeglichen sondern erhalten, die Selbe Prozentposition
     Percent_Position_Path_1=Percent_Position;
     Percent_Position_Path_2=Percent_Position;
     //---------------------------------------
     //KOORDINATEN_POSITION AN EINEM BESTIMMTEN SCHNITTPUNKT AUSLESEN
     if(doc->FindObject(Path_1_Name)) Path_1_Position_Point=Path_1_Object->GetSplinePoint(Path_1_Object->UniformToNatural(Percer
//println("Pedestrian_Path_Spline gefunden");
     if(doc->FindObject(Path_2_Name)) Path_2_Position_Point=Path_2_Object->GetSplinePoint(Path_2_Object->UniformToNatural(Perc
//println("Barrier_Free_Path_Spline gefunden");
    //1. Schnitt und distanz zwischen 2 od 3 Punkten Ausrechnen.
    if(doc->FindObject("Pedestrian_Path_Spline") && doc->FindObject("Barrier_Free_Path_Spline"))
    {
      XZ_Dist_2_Points_Calc(Path_1_Position_Point,Path_2_Position_Point);
      Distance_12=Dist_current_Points;
      //println("Die Distanz zwischen Spline1 und Spline 2 beträgt: ",Distance_12);
    }
    //2. die größte Distanz ermitteln -> nur bei 3 pfaden notwendig. -> Die 2 am weitesten entfernten Punkte als Randpunkte definieren.
    a=0;
    Points=0;
    Biggest_Distance=0;
    var SphereName;
    var Sphere_Scale;
    if(Distance_12>=Biggest_Distance)
    {
      Biggest_Distance=Distance_12;
      Points=12; //Bedeutet weitesten Entfernung besteht zwischen 1 und 2. Spline.
      //println("Biggest_Distance=12");
    }
    println("Strecke ",Points," hat die größte Distnaz mit=",Biggest_Distance);
    /*
    //MAKE_SPHERE(Scale,Name,Position,Parent) --zur Kontrolle
    Sphere_Scale=0.2;
    if(doc->FindObject(Path_1_Name))
    {
      SphereName=("1_PATH_Point_"+tostring(i));
      MAKE_SPHERE(Sphere_Scale,SphereName,Path_1_Position_Point,Null);
    }
    if(doc->FindObject(Path_2_Name))
    {
      SphereName=("2_PATH_Point_"+tostring(i));
      MAKE_SPHERE(Sphere_Scale,SphereName,Path_2_Position_Point,Null);
     }
    */
    //Bei 3 Stecken müssten hier mehere Kombinations-Fälle unterschieden werden (12,13,23)
    //3. Steigung ermittlen
    if(Points==12)
    {
      Dist_X=Path_1_Position_Point.x-Path_2_Position_Point.x;
      Dist_Y=Path_1_Position_Point.y-Path_2_Position_Point.y;
      Dist_Z=Path_1_Position_Point.z-Path_2_Position_Point.z;
      Mid_Point=(Path_1_Position_Point+Path_2_Position_Point)/2;
    }
    //Scale Faktor Distance
    //Scale=Biggest_Distance*Scale_Factor; //Die Matrix Überlappt den Weg um z.B.: ca. 20% der größten Wegdistanz.
    //Gesamtbreiten mit den jeweiligen Wegen mal die Skallierung
    Scale=Scale_Factor+1;
    //STEIGUNG IN ALLE 3 ACHSRICHTUNGEN ERMITTELN
    X_Degree_L=(Dist_X+(Pedestrian_Width)*(Dist_X/Biggest_Distance))*Scale/(Anzahl_Splines-1);
    Y_Degree_L=(Dist_Y+(Pedestrian_Width)*(Dist_Y/Biggest_Distance))*Scale/(Anzahl_Splines-1);
    Z_Degree_L=(Dist_Z+(Pedestrian_Width)*(Dist_Z/Biggest_Distance))*Scale/(Anzahl_Splines-1);
    X_Degree_R=(Dist_X+(WheelChair_Width)*(Dist_X/Biggest_Distance))*Scale/(Anzahl_Splines-1);
    Y_Degree_R=(Dist_Y+(WheelChair_Width)*(Dist_Y/Biggest_Distance))*Scale/(Anzahl_Splines-1);
    Z_Degree_R=(Dist_Z+(WheelChair_Width)*(Dist_Z/Biggest_Distance))*Scale/(Anzahl_Splines-1);
    //Für die if-Beurteilung
    X_Degree=(Dist_X)/(Anzahl_Splines-1)*Scale;
    Y_Degree=(Dist_Y)/(Anzahl_Splines-1)*Scale;
    Z_Degree=(Dist_Z)/(Anzahl_Splines-1)*Scale;
    if(i==SplinePunkte-1) println("Achtung Fehler");
```

```
    println("Pfad1_Position: ",Percent_Position_Path_1);
    //println("Pfad2_Position: ",Percent_Position_Path_2);
    println("X_Degree_L: ",X_Degree_L);
    println("Y_Degree_L: ",Y_Degree_L);
    println("Z_Degree_L: ",Z_Degree_L);
    println("X_Degree_R: ",X_Degree_R);
    println("Y_Degree_R: ",Y_Degree_R);
    println("Z_Degree_R: ",Z_Degree_R);
    //println("X_Degree: ",X_Degree);
    //println("Y_Degree: ",Y_Degree);
    //println("Z_Degree: ",Z_Degree);
    //BetweenPaths_Points=(Anzahl_Splines-1)-Scale;

    //4. BRIDGE_POINTS_ARRAY Punkte anpassen mit Scalefaktor
    HalfSplines=Anzahl_Splines/2;
    //println("HalfSplines= ",HalfSplines);
    //NUN WIRD DIE FLÄCHE ZWISCHEN DEN 2 ÄUßERSTEN SPLINES GEKNETET
    //UM BEI BEDARF DIE KONTUR NUR IM GRUNDRISS ZU KNETEN MUSS DIE Y-BERECHNUNG DEAKTIVIEI
    for(cnt=0;cnt<Anzahl_Splines;cnt++)
    {
      //Es wird festgestellt ob der eine Punkt über oder unter dem anderen liegen
      //Je nachdem ob die Steigungswerte positiv oder negativ sind wird
      if(cnt<HalfSplines)
      {
        if(X_Degree<=0 && Z_Degree>=0) BRIDGE_POINTS_ARRAY[cnt][i].z=Mid_Point.z+Z_Degree_L*(HalfSpli
        if(X_Degree<=0 && Z_Degree>=0) BRIDGE_POINTS_ARRAY[cnt][i].x=Mid_Point.x+X_Degree_L*(HalfSpli
        if(X_Degree<=0 && Z_Degree>=0) BRIDGE_POINTS_ARRAY[cnt][i].y=Mid_Point.y+Y_Degree_L*(HalfSpli
deaktivieren
        if(X_Degree<=0 && Z_Degree<=0) BRIDGE_POINTS_ARRAY[cnt][i].z=Mid_Point.z-Z_Degree_L*(HalfSpli
        if(X_Degree<=0 && Z_Degree<=0) BRIDGE_POINTS_ARRAY[cnt][i].x=Mid_Point.x-X_Degree_L*(HalfSpli
        if(X_Degree<=0 && Z_Degree<=0) BRIDGE_POINTS_ARRAY[cnt][i].y=Mid_Point.y-Y_Degree_L*(HalfSpli
deaktivieren
        //Dieser Teil wurde nicht überprüft, da dieser Fall nie eingetreten ist!!
        if(X_Degree>0 && Z_Degree<=0) BRIDGE_POINTS_ARRAY[cnt][i].z=Mid_Point.z-Z_Degree_L*(HalfSplin
        if(X_Degree>0 && Z_Degree<=0) BRIDGE_POINTS_ARRAY[cnt][i].x=Mid_Point.x-X_Degree_L*(HalfSplin
        if(X_Degree>0 && Z_Degree<=0) BRIDGE_POINTS_ARRAY[cnt][i].y=Mid_Point.y-Y_Degree_L*(HalfSplin
        if(X_Degree>0 && Z_Degree>=0) BRIDGE_POINTS_ARRAY[cnt][i].z=Mid_Point.z+Z_Degree_L*(HalfSplin
Radausen und Fußgänger außen
        if(X_Degree>0 && Z_Degree>=0) BRIDGE_POINTS_ARRAY[cnt][i].x=Mid_Point.x+X_Degree_L*(HalfSplin
Radausen und Fußgänger außen
        if(X_Degree>0 && Z_Degree>=0) BRIDGE_POINTS_ARRAY[cnt][i].y=Mid_Point.y+Y_Degree_L*(HalfSplin
        //Ende nicht überprüfte Fläle
      }
      if(cnt>=HalfSplines)
      {
        if(X_Degree<=0 && Z_Degree>=0) BRIDGE_POINTS_ARRAY[cnt][i].z=Mid_Point.z+Z_Degree_R*(HalfSpl
        if(X_Degree<=0 && Z_Degree>=0) BRIDGE_POINTS_ARRAY[cnt][i].x=Mid_Point.x+X_Degree_R*(HalfSpl
        if(X_Degree<=0 && Z_Degree>=0) BRIDGE_POINTS_ARRAY[cnt][i].y=Mid_Point.y+Y_Degree_R*(HalfSpl
deaktivieren
        if(X_Degree<=0 && Z_Degree<=0) BRIDGE_POINTS_ARRAY[cnt][i].z=Mid_Point.z-Z_Degree_R*(HalfSpli
        if(X_Degree<=0 && Z_Degree<=0) BRIDGE_POINTS_ARRAY[cnt][i].x=Mid_Point.x-X_Degree_R*(HalfSpli
        if(X_Degree<=0 && Z_Degree<=0) BRIDGE_POINTS_ARRAY[cnt][i].y=Mid_Point.y-Y_Degree_R*(HalfSpli
deaktivieren
        //Dieser Teil wurde nicht überprüft, da dieser Fall nie eingetreten ist!!
        if(X_Degree>0 && Z_Degree<=0) BRIDGE_POINTS_ARRAY[cnt][i].z=Mid_Point.z-Z_Degree_R*(HalfSplin
        if(X_Degree>0 && Z_Degree<=0) BRIDGE_POINTS_ARRAY[cnt][i].x=Mid_Point.x-X_Degree_R*(HalfSplin
        if(X_Degree>0 && Z_Degree<=0) BRIDGE_POINTS_ARRAY[cnt][i].y=Mid_Point.y-Y_Degree_R*(HalfSplin
        if(X_Degree>0 && Z_Degree>=0) BRIDGE_POINTS_ARRAY[cnt][i].z=Mid_Point.z+Z_Degree_R*(HalfSplin
Radausen und Fußgänger außen
        if(X_Degree>0 && Z_Degree>=0) BRIDGE_POINTS_ARRAY[cnt][i].x=Mid_Point.x+X_Degree_R*(HalfSplin
Radausen und Fußgänger außen
        if(X_Degree>0 && Z_Degree>=0) BRIDGE_POINTS_ARRAY[cnt][i].y=Mid_Point.y+Y_Degree_R*(HalfSplin
      }
    }
    Percent_Position=Percent_Position+Section_Percent_Step;
  }
  //Spline Updaten -> Neuzeichen
  println("Splines Updaten");
  UPDATE_BRIDGE_SPLINES();
}

//-------------------------------------------------------------------------------------
//-------------------------------------------------------------------------------------
main()
{
  doc=GetActiveDocument();
  if (ONOFF==TRUE && doc->FindObject("Path_Splines") && doc->FindObject("Pedestrian_Path_Spline") && d
  {
    var a; // Zählvariablen
    var Path_Object,Path_Name;
    BRIDGE_POINTS_ARRAY = new (array,Anzahl_Splines,SplinePunkte);

    //Punkte_Array von der Brückenflächengeompetrie erzeugen
    GENERATE_BRIDE_GEOMETRY_ARRAY(); //ERGEBNIS: BRIDGE_POINTS_ARRAY[][];->[SplineAnzahl][Anz


    //DEFORM_BASICSURFACE_TO_PATHSPLINES
    DEFORM_BASICSURFACE_TO_PATHSPLINES(Path_Object);

    //LoftNurbs-Updaten
    var LoftnurbsNEW,Loftname;
    Loftname=("LoftObjekt");
    LoftnurbsNEW=doc->FindObject(Loftname);
    LoftnurbsNEW#ID_BASEOBJECT_GENERATOR_FLAG=False;
    LoftnurbsNEW#ID_BASEOBJECT_GENERATOR_FLAG=True;
  }
  else
  {
    if(ONOFF==TRUE) println("Es gibt keine 2 Pfade!!!");
  }
  OFF==FALSE;
}



//-------------------------------
//PATHCLEARANCE_SCRIPT  |
//-------------------------------
//by Christian Pichlkastner
//
//Diese Skript ebnet die Brückenoberfläche in den Wegbereichen
//
//Eingang: Anzahl_Splines, SplinePunkte, Pedestrian_Width, Biker_Width, WheelChair_Width, ONOFF
//Ausgang: OFF
//GLOBALE VARIABLEN
var doc; // Damit Objekte in aktueller Szene benützt/gesucht werden können
var BRIDGE_POINTS_ARRAY;
var Dist_current_Points;
var Path_Points,PathPointNum;
var path_cnt;
var Path_Points_Cnt;
var Nearst_Spline,Nearst_Point,Reset_Distance;

//Correct_Path_Overlapping
CORRECT_PATH_OVERLAPPING()
{
  //Wenn die Brücken geometrie im Pfadlichtenbereich andere Y-Werte als der Pfadaufweist,dann
  //überlappt ein anderer Pfad diesen und dieser muss an dieser Stelle nachkorrigiert werden.
  //MAN KANN ABER LEIDER NUR AUF KONTROLLPUNKTE DER PFADSPLINE ZUGREIFEN!!!
  //VORSCHLAG: FIKTIVE NACHBARPUNKTE (über Prozentrechnung) ERMITTELN UND AUF ÜBERLAPPUNG CHECKEN.
  //JE NACH STÄRTKE DER ÜBERLAPPUNG WERDEN DIE KONTROLLPUNKTE MEHR ODER WENIG STARK VERSCHOBI
  //einfachste Variante: stimmt ein y-Wert an einer prozentstelle in der lichte nicht mit der Brückengeometie
  //.....dann wird der näheste Kontrollpunkt auf den Y-Brückengeometiewert gesetzt.
  println("Correct_Path_Overlapping");
  var i,a,cnt,c,j;
  var Path_Point_Coord;
  var CurrentBridgePoint;
  var Percent_Position=0.0,Section_Percent_Step;
  var Path_Object,Path_Name,Path_Control_Point,Current_Control_Point,Path_Width,Path_Points_Cnt,PointNum;
  var SamplingResolution,Sampling_Sections;
  var Height_Difference,Smallest_Distance,Tolerance_Value;

  //var NODEDATA=new(array,Anzahl_Splines,SplinePunkte); //SplineNummer/PunktNummer/Daten
  SamplingResolution=(SplinePunkte)*10;
  Sampling_Sections=SamplingResolution/(SplinePunkte);
  Section_Percent_Step=1.0/SamplingResolution;
  //Make Path Clear
  for(j=1;j<=2;j++)
  {
    if(j==1) Path_Name=("Biker_Path_Spline");
    if(j==2) Path_Name=("Pedestrian_Path_Spline");
    if(j==1) Path_Width=Biker_Width;
    if(j==2) Path_Width=Pedestrian_Width;
    println(" ");
    println(j," Überprüfe ",Path_Name);
    if(doc->FindObject(Path_Name))
    {
      Path_Object=doc->FindObject(Path_Name);
      for(a=0;a<Anzahl_Splines;a++)
      {
        for (cnt=0;cnt<SplinePunkte;cnt++)
        {
          CurrentBridgePoint=BRIDGE_POINTS_ARRAY[a][cnt];
          //println("Punkt [",a,"][",cnt,"] wird verglichen");
          Percent_Position=0;
          for (i=0;i< SamplingResolution;i++)
          {
            Path_Point_Coord=Path_Object->GetSplinePoint(Percent_Position,0);
            XZ_Dist_2_Points_Calc(Path_Point_Coord,CurrentBridgePoint);
```

```
          Dist_current_Points=abs(Dist_current_Points);

          if(Dist_current_Points<=(Path_Width/2)) //Wenn im Pfadbereich dann check Y-Wert
          {
            //println(i," Der Brückenpunkt ",Percent_Position," ist im Pfadbereich");
            Height_Difference=Path_Point_Coord.y-CurrentBridgePoint.y;
            BRIDGE_POINTS_ARRAY[a][cnt].y=Path_Point_Coord.y;
            Tolerance_Value=20;
             if(abs(Height_Difference)>=Tolerance_Value)
             {
               //Nähesten PfadKontrollPunkt finden und verändern.
               //desto weiter dieser Punkt entfernt ist desto stärker
               //wird dieser Konrollpunkt verändert.
               //println("Höhendifferenz beträgt: ",Height_Difference);
               Path_Points_Cnt=Path_Object->GetPointCount();
               Smallest_Distance=10000000000;
               PointNum=0;
               for(c=0;c<Path_Points_Cnt;c++)
               {
                 Path_Control_Point=Path_Object->GetPoint(c);
                 XYZ_Dist_2_Points_Calc(Path_Control_Point,CurrentBridgePoint);
                 Dist_current_Points=abs(Dist_current_Points);
                 if(Dist_current_Points<Smallest_Distance)
                 {
                   Smallest_Distance=Dist_current_Points;
                   Current_Control_Point=Path_Control_Point;
                   PointNum=c;
                 }
               }
               //println("Nähester Control Punkt von ist ",Smallest_Distance," entfernt.");
               //println("Nähester Control Punkt von ",Path_Name," ist Punkt",PointNum," mit den Koordinaten: ",Cu
               if (abs(Smallest_Distance/200)>= 0) Height_Difference=Height_Difference*(1/(Smallest_Distance/1
abgeschwächt.

               //println("Neue Höhendifferenz beträgt: ",Height_Difference);
               Current_Control_Point.y=Current_Control_Point.y-Height_Difference;  //Hier wird der Pfadpunkt ko
               Path_Object->SetPoint(PointNum,Current_Control_Point);
               Path_Object#ID_BASEOBJECT_GENERATOR_FLAG=False;
               Path_Object#ID_BASEOBJECT_GENERATOR_FLAG=True;
               //println("Neuer Control Punkt ",PointNum," mit den Koordinaten: ",Current_Control_Point);
             }
          }
          Percent_Position=Percent_Position+Section_Percent_Step;
        }
       }
      }
    }//IF PATH FOUND ENDE
  }//PATH CHOOSER ENDE
} //CORRECT OVERLAPPING ENDE

//Make Path Clear
MAKE_PATH_CLEAR(Path_Height, Path_Width,Path_Name)
{
  println(" ");
  println("Make ",Path_Name," clear");
  println(" ");
  var i,a,cnt;
  var Path_Point_Coord;
  var CurrentBridgePoint;
  var Percent_Position=0.0,Section_Percent_Step;
  var Path_Object,Path_Full_Lenght;
  var SamplingResolution,Sampling_Sections,Sampling_Error_Dist;
  //var NODEDATA=new(array,Anzahl_Splines,SplinePunkte); //SplineNummer/PunktNummer/Daten
  Path_Object=doc->FindObject(Path_Name);
  SamplingResolution=(SplinePunkte)*10;
  Sampling_Sections=SamplingResolution/(SplinePunkte);
  Section_Percent_Step=1.0/SamplingResolution;
  Path_Object->InitLength(0);
  Path_Full_Lenght=Path_Object->GetLength();
  Sampling_Error_Dist=Path_Full_Lenght/SamplingResolution;
  println("SamplingResolution: ",SamplingResolution);
  println("Sampling_Sections: ", Sampling_Sections);
  println("Section_Percent_Step: ",Section_Percent_Step);
  println("Sampling_Error_Dist: ",Sampling_Error_Dist);
  for(a=0;a<Anzahl_Splines;a++)
  {
    for (cnt=0;cnt<SplinePunkte;cnt++)
    {
      CurrentBridgePoint=BRIDGE_POINTS_ARRAY[a][cnt];
      //println("Punkt [",a,"][",cnt,"] wird verglichen");
      Percent_Position=0;
      for (i=0;i< SamplingResolution;i++)
      {
        //println("Punkt [",a,"][",cnt,"] wird mit Pfadpunkt bei ",Percent_Position,"% verglichen.");
        //println("Pfadbreite: ",Path_Width);
        Path_Point_Coord=Path_Object->GetSplinePoint(Percent_Position,0);
        XZ_Dist_2_Points_Calc(Path_Point_Coord,CurrentBridgePoint);
        //println(i,"Distanz: ",Dist_current_Points);
        Dist_current_Points=abs(Dist_current_Points);
        if(Dist_current_Points<=((Path_Width/2)+Sampling_Error_Dist))
        {
          //println("Y-Wert wird verändert.");
          BRIDGE_POINTS_ARRAY[a][cnt].y=Path_Point_Coord.y;
        }
        Percent_Position=Percent_Position+Section_Percent_Step;
      }
    }
  }
  UPDATE_BRIDGE_SPLINES();
}

//-------------------------------------------------------------------------------------
//-------------------------------------------------------------------------------------
main()
{
  doc=GetActiveDocument();
  if (ONOFF==TRUE && doc->FindObject("LoftObjekt") && doc->FindObject("Path_Splines"))
  {
    var a; // Zählvariablen
    var Biker_Height,Biker_Width, Pedestrian_Height, Pedestrian_Width,  Wheelchair_Height,   Wheelchair_Width;
    var Path_Height,Path_Width;
    var Path_Name;
    BRIDGE_POINTS_ARRAY = new (array,Anzahl_Splines,SplinePunkte);
    //Weg_Lichte_Profil (Rechteck)
    Biker_Height=180;
    Pedestrian_Height=200;
    Wheelchair_Height=200;
    println("START TO MAKE CLEARANCE OF PATHS!!!");

    //Punkte_Array von der Brückenflächengeompetrie erzeugen
    GENERATE_BRIDE_GEOMETRY_ARRAY(); //ERGEBNIS: BRIDGE_POINTS_ARRAY[][];->[SplineAnzahl][Anzahl_Splines]
    //Make Path Clear
    for(a=1;a<=3;a++)
    {
      if(a==1) Path_Name=("Pedestrian_Path_Spline");
      if(a==2) Path_Name=("Biker_Path_Spline");
      if(a==3) Path_Name=("Barrier_Free_Path_Spline");
      if(doc->FindObject(Path_Name))
      {
        println("Checking Path: ",Path_Name);
        Path_Height=Biker_Height;
        Path_Width=Biker_Width;
        if(a==1)    Path_Height= Pedestrian_Height; Path_Width=Biker_Width;
        if(a==2)Path_Height=Biker_Height; Path_Width=Pedestrian_Width;
        if(a==3)Path_Height=   Wheelchair_Height; Path_Width=WheelChair_Width;
        MAKE_PATH_CLEAR(Path_Height, Path_Width,Path_Name);
      }
    }
    //Correct_Path_Overlapping
    CORRECT_PATH_OVERLAPPING();
    //LoftNurbs-Updaten
    var LoftnurbsNEW,Loftname;
    Loftname=("LoftObjekt");
    LoftnurbsNEW=doc->FindObject(Loftname);
    LoftnurbsNEW#ID_BASEOBJECT_GENERATOR_FLAG=False;
    LoftnurbsNEW#ID_BASEOBJECT_GENERATOR_FLAG=True;
  }
  else
  {
    if(ONOFF==TRUE) println("Es gibt kein Pfad!!!");
  }
  OFF==FALSE;
}



//-----------------------------------------
//STRUCTURE_EVALUATING_SCRIPT    |
//-----------------------------------------
//by Christian Pichlkastner
//Dieses Cinema4d Coffee Skript überprüft die Brüche an mehreren Stellen auf ihre
//Tragfähigkeit und versucht eine Vorbemessung zu erzielen.
//Wenn die Brücke nicht Tragfähig ist, dann kann die Brücke tragfähig gemacht werden.
//
//Eingang: Anzahl_Splines, SplinePunkte, Wasserpegel, Material_Typ, Structure_Load,
//ServiceLoad, Diagram_Factor, Sigma_Zul, Torque_Calc_Mode, Support_Position, ONOFF,
//Beam_Num_In, Pedestrian_Width, Biker_Width, WheelChair_Width, Spline_Deformer_Factor_In,
//Barrier_Free_Path_Degree_In, Path_Overlapping_Factor_In
```

```
//
//Ausgang: OFF, SPLINE_DEFORMER_ON, CHECK_SPLINE_DEGREES_ON, Barrier_Free_Path_Degree_Out,
//KNEAD_TO_PATHS_ON, PATH_CLEARANCE_ON, CREATE_HANDRAIL_ON, Anz_Splines_Out, Anz_Splines
//Pedestrian_Width_Out, Biker_Width_Out, WheelChair_Width_Out, Spline_Deformer_Factor_Out,
//Path_Overlapping_Factor_Out, Beam_Num_Out, KNEAD_TO_SPLINE_TO_ATTRAKTORES_ON,
//Attractor_Spline_Object_Out

//GLOBALE VARIABLEN
var doc; // Damit Objekte in aktueller Szene benützt/gesucht werden können
var Dist_current_Points;
var BRIDGE_POINTS_ARRAY;
var EntirePlane,Sector_Plane,SinglePlanes,SinglePlane_BalancePoint,Planes_Count,Surface_Balance_Point; //
var SinglePlanes_Load,Single_Volume_Cubes_Load,Structure_Load_NEW,Support_Res_Force_A,Support_Res
Support_Z_Force_A,Support_Z_Force_B;
var Section_Torque_X,Section_Torque_Z,Res_Section_Torque,Section_Torque_RES;
var System_Axis_Points;
var Ideal_Beam_Heights;//Hier ist die Dicke des Trägers gespeichert
var BRIDGE_VOLUME_POINTS_ARRAY;
var Cube_Volume_Balance_Point,Single_Cube_Volume,Pyramid_Angle,Pyramind_Base_Area;
var Positiv_Cubes_Volume, Positiv_Entire_Volume;
var Negativ_Cubes_Volume, Negativ_Entire_Volume;
var PROFILETYPE;
var Sum_Cubes_Balance_Points,Entire_Structure_Load_Balance_Point, Entire_Service_Load_Balance_Point,E
var Entire_Structure_Volume,Sum_of_Pyramides_Volumes;
var Entire_Bridge_Load,Single_Bridge_Load_Points,Single_Bridge_Balance_Points,Entire_Bridge_Balance_Po
var Minimal_Construction_Height;
var OG_Height,UG_Height,Left_Width,Right_Width,OG_Height_Percent,UG_Height_Percent,Left_Width_Percent
Untergurtweite,...in Prozent;
var Beam_Proportion_Faktor,Obergurt_Mindesthoehe,Untergurt_Mindesthoehe;
var Section_Shape_Mode; //Definiert wie sich das Profil zum Rand hin verläuft
var Stegbreiten_Faktor; //Definiert, wie Breit die Stege im Verhältnis zur Höhe sein müssen.
var Rectangle_Area_Value,Rectangle_Area_Balance_Point,Triangle_Area_Value,Triangle_Area_Balance_Point;
var section_Load_pos,Section_Load;
var Section_Moments_of_Inertia_Y,Section_Moments_of_Inertia_XZ,Section_Area_Balance_Point,Section_Are
var Formula_Divident_Arm_1,Formula_Divident_Arm_2,Faktor_1_Enable,Faktor_2_Enable,Faktor_2_Proportio
Faktor_2_Enable_Extern,Section_Shape_Mode_Extern,Change_Bridge_Symmetry;
var Global_Safty_Factor,Beam_Number_Intern; //Stabilitäts_Check
var OG_Restiance_Torque, UG_Restiance_Torque, Left_Side_Restiance_Torque, Right_Side_Restiance_Torque
var Sigma_OG_Actual,Sigma_UG_Actual,Sigma_Left_Actual,Sigma_Right_Actual;
var OG_Efficiency, UG_Efficiency, Left_Efficiency, Right_Efficiency,Bending_of_Arm_1_Efficiency,Bending_of_
Bending_Section; //Trägerauslastung
var Waterlevel,Waterlevel_Point;
var Torque_Curve_Calc,Torque_Curve_Calc_Mode; //Trägervordimensionierung anhand des Momenten verlau
var Change_Bridge_Thickness; //0: bedeutet Brückendicke soll nicht verändert werden; 1: bedeutet eine Veränd
nach oben erlaubt.
var Formula_Divident_Arm_1_Extern,Formula_Divident_Arm_2_Extern;
var Path_Area_Point;
var Section_Balance_Extern;//Kann den Wert 0 oder 1 haben; 0=kein Wegbereich;1=Wegbereich;
var Cantilever_Arm_1,Cantilever_Arm_2,Bridge_Lenght,Bridge_Segment_Lenght;
var Angle_Weight; //Winkelgewicht
var Fittest_Beam_Number;

//MAKE_PYRAMID
MAKE_PYRAMID(Size,Name,Position,Parent)
{
  var Pyramid_Object;
  if(!doc->FindObject(Name))
  {
    Pyramid_Object = new(PyramidObject);
    Pyramid_Object->SetName(Name);
    Pyramid_Object#PRIM_PYRAMID_LEN=vector(Size,Size,Size);
    Position.y= Position.y-Size/2;
    Pyramid_Object->SetPosition(Position);
    doc->InsertObject(Pyramid_Object,Parent, null);
  }
  else
  {
    Pyramid_Object = doc->FindObject(Name);
    Position.y= Position.y-Size/2;
    Pyramid_Object->SetPosition(Position);
  }
}

//MAKE_CUBE
MAKE_CUBE(Size,Name,Position,Parent)
{
  var Cube_Object;
  if(!doc->FindObject(Name))
  {
    Cube_Object = new(CubeObject);
    Cube_Object->SetName(Name);
    Cube_Object#PRIM_CUBE_LEN=Size;
    Cube_Object->SetPosition(Position);
    doc->InsertObject(Cube_Object,Parent, null);
  }
  else
  {
    Cube_Object = doc->FindObject(Name);
    Cube_Object#PRIM_CUBE_LEN=Size;
    Cube_Object->SetPosition(Position);
  }
}

//MAKE A 4 Points defined RECTANGLE
MAKE_4_POINT_RECTANGLE(UL,UR,DR,DL,Name,Parent)
{
  var Rectangle_Object;
  var Rectangle_Points=new(array,4);

  if(!doc->FindObject(Name))
  {
    Rectangle_Object=new(SplineObject);
    var vc=new(VariableChanged);
    var bt=new(BackupTags);
    bt->Init(Rectangle_Object);    // required because we are changing the number of points
    vc->Init(0,4); // it used to have 0 points, now has 5
    if (Rectangle_Object->Message(MSG_POINTS_CHANGED, vc))
    {
      bt->Restore();
    }
    doc->InsertObject(Rectangle_Object,Parent, null);
    Rectangle_Object->SetName(Name);
    Rectangle_Object#SPLINEOBJECT_TYPE=0; // set the type of spline here
    Rectangle_Object#SPLINEOBJECT_CLOSED=True;
  }
  Rectangle_Points[0]=UL;
  Rectangle_Points[1]=UR;
  Rectangle_Points[2]=DR;
  Rectangle_Points[3]=DL;
  //println(Path_Punkte);
  Rectangle_Object=doc->FindObject(Name);
  Rectangle_Object->SetPoints(Rectangle_Points);
  Rectangle_Object->Message(MSG_UPDATE);
}

//MAKE_3RD_BEAM
MAKE_3RD_BEAM_PROFILE(Section,Ideal_Beam_Heights,Name,Parent)
{
  var c,cnt,a,i;
  var Profil_Object;
  var Profil_Punkte_Anzahl=Anzahl_Splines*2;
  var Profile_Points=new(array,Profil_Punkte_Anzahl);
  if(!doc->FindObject(Name))
  {
    Profil_Object=new(SplineObject);
    var vc=new(VariableChanged);
    var bt=new(BackupTags);
    bt->Init(Profil_Object);    // required because we are changing the number of points
    vc->Init(0,Profil_Punkte_Anzahl); // it used to have 0 points, now has 5
    if (Profil_Object->Message(MSG_POINTS_CHANGED, vc))
    {
      bt->Restore();
    }
    doc->InsertObject(Profil_Object,Parent, null);
    Profil_Object->SetName(Name);
    Profil_Object#SPLINEOBJECT_TYPE=0; // set the type of spline here
    Profil_Object#SPLINEOBJECT_CLOSED=True;
  }
  for(c=0;c<Profil_Punkte_Anzahl;c++)
  {
    Profile_Points[c]=BRIDGE_VOLUME_POINTS_ARRAY[c][Section];
  }
  //println(Path_Punkte);
  Profil_Object=doc->FindObject(Name);
  Profil_Object->SetPoints(Profile_Points);
  Profil_Object->Message(MSG_UPDATE);
}

//MAKE_PARABLE_KONTUR(Ideal_Beam_Heights)
MAKE_PARABLE_KONTUR(Ideal_Beam_Heights)
{
  //-------------------------------------------------------------------------
  //-----------------Parabelförmige Verjüngung zum Rand hin-----------------------
  //-------------------------------------------------------------------------
  //Pro Segment verjüngt sich die Höhe nach einer Quadratischen Parabel
  //Die Verjüngung müsste in Richtunge der idealline zwischen ufer a und b des
  //Brückenschwerpunktes berücksichtigt werden um eine gering torsion zu erhalten.
  var a,cnt;
```

```
var Mittlere_Spline_1,Mittlere_Spline_2,Abweichung;
var Dist,current_Spline,Middle_Spline,Max_DIST_1,Max_DIST_2,Max_Percent_1,Max_Percent_2,Faktor_1,Fak

Abweichung=Anzahl_Splines/2;
if(Abweichung!=int(Abweichung))
{ //ungerade Splineanzahl
  Mittlere_Spline_1=int(Anzahl_Splines/2)+1; //wenn immer abgerundet wird...dann stimmt dies //Ansonsten.
  Mittlere_Spline_2=int(Anzahl_Splines/2)+1; //= Spline 1
}
else
{
  //gerade Splineanzahl=2 Mittelsplines
  Mittlere_Spline_1=int(Anzahl_Splines/2)-1;
  Mittlere_Spline_2=int(Anzahl_Splines/2);
}
//println("Mittlere_Spline_1: ",Mittlere_Spline_1);
//println("Mittlere_Spline_2: ",Mittlere_Spline_2);

for (a=0;a<SplinePunkte;a++)
{
  //1.Hälfte
  for (cnt=0;cnt<=Mittlere_Spline_1;cnt++)
  {
    //println("------------------------------------------------- ");
    //println(cnt,".Spline Schleife 1 Schnitt: ",a);
    current_Spline=BRIDGE_POINTS_ARRAY[0][a];
    Middle_Spline=BRIDGE_POINTS_ARRAY[Mittlere_Spline_1][a];
    XYZ_Dist_2_Points_Calc(current_Spline,Middle_Spline);
    Max_DIST_1=abs(Dist_current_Points);
    //println("Max_DIST_1: ",Max_DIST_1);
    current_Spline=BRIDGE_POINTS_ARRAY[cnt][a];
    Middle_Spline=BRIDGE_POINTS_ARRAY[Mittlere_Spline_1][a];
    XYZ_Dist_2_Points_Calc(current_Spline,Middle_Spline);
    Dist=abs(Dist_current_Points);
    //println("DIST: ",Dist);
    //je nach Modus wird der Schnitt gegen Rand verlaufen
    if(Section_Shape_Mode==1) //Konkav
    {
      Max_Percent_1=sqrt(Max_DIST_1)/2;
      Faktor_1=(sqrt(Dist)/2)/Max_Percent_1;
      Faktor_1=abs(1-Faktor_1);
    }
    if(Section_Shape_Mode==2) //Konvex
    {
      Max_Percent_1=(Max_DIST_1*Max_DIST_1)/2;
      Faktor_1=((Dist*Dist)/2)/Max_Percent_1;
      Faktor_1=abs(1-Faktor_1);
    }
    Ideal_Beam_Heights[cnt][a]=(Ideal_Beam_Heights[cnt][a])*Faktor_1+(Minimal_Construction_Height/3.5);
  }
  //2.Hälfte
  for (cnt=Anzahl_Splines-1;cnt>=Mittlere_Spline_2;cnt--)
  {
    //println(cnt,".Spline Schleife 2 Schnitt: ",a);
    current_Spline=BRIDGE_POINTS_ARRAY[Anzahl_Splines-1][a];
    Middle_Spline=BRIDGE_POINTS_ARRAY[Mittlere_Spline_2][a];
    XYZ_Dist_2_Points_Calc(current_Spline,Middle_Spline);
    Max_DIST_2=abs(Dist_current_Points);
    //println("Max_DIST_2 ",Max_DIST_2);
    current_Spline=BRIDGE_POINTS_ARRAY[cnt][a];
    Middle_Spline=BRIDGE_POINTS_ARRAY[Mittlere_Spline_2][a];
    XYZ_Dist_2_Points_Calc(current_Spline,Middle_Spline);
    Dist=abs(Dist_current_Points);
    //println("DIST ",Dist);
    //je nach Modus wird der Schnitt gegen Rand verlaufen
    if(Section_Shape_Mode==1) //Konkav
    {
      Max_Percent_2=sqrt(Max_DIST_2)/2;
      Faktor_2=(sqrt(Dist)/2)/Max_Percent_2;
      Faktor_2=abs(1-Faktor_2);
    }
    if(Section_Shape_Mode==2) //Konvex
    {
      Max_Percent_2=(Max_DIST_2*Max_DIST_2)/2;
      Faktor_2=((Dist*Dist)/2)/Max_Percent_2;
      Faktor_2=abs(1-Faktor_2);
    }
    //println("Faktor_2 ",Faktor_2);
    //println("ALT_Ideal_Beam_Heights[cnt][a]: ",Ideal_Beam_Heights[cnt][a]);
    //Ideal_Beam_Heights[cnt][a]=(Ideal_Beam_Heights[cnt][a]+Minimal_Height)*Faktor_2;
    Ideal_Beam_Heights[cnt][a]=(Ideal_Beam_Heights[cnt][a])*Faktor_2+(Minimal_Construction_Height/3.5);
    //if(Ideal_Beam_Heights[cnt][a]<Minimal_Height)  Ideal_Beam_Heights[cnt][a]=10;
    //println("NEU_Ideal_Beam_Heights[",cnt,"][",a,"]: ",Ideal_Beam_Heights[cnt][a]);
  }
  }
  return Ideal_Beam_Heights;
}

//Read_Bridge_Outlines;
READ_BRIDGE_SHAPE()
{
  println(" ");
  println("READ BRIDGE_SHAPE");
  var a,cnt;
  var Beamkontur_Object;
  var Beamkontur_Name;
  var Changed__Point,temp_Point_Upside,temp_Point_Downside;
  var temp_Height;
  var Profil_Points=2*Anzahl_Splines;
  var SplineNrDownside;
  for (a=0;a<SplinePunkte;a++)
  {
    for (cnt=0;cnt<Anzahl_Splines;cnt++)
    {
      Beamkontur_Name=(tostring(Fittest_Beam_Number)+"BEAMKONTUR"+tostring(a)); //bsp.: 1BEAMKONTUR18
      Beamkontur_Object=doc->FindObject(Beamkontur_Name);
      //Achtung die Konturpunkte laufen rund um den Träger->DownPoint ist nicht cnt+Anzahl_Splines
      temp_Point_Upside=Beamkontur_Object->GetPoint(cnt);
      //println("Oberflächenpunkt gelesen BRIDGE_POINTS_ARRAY[",cnt,"][",a,"]");
      BRIDGE_VOLUME_POINTS_ARRAY[cnt][a]=BRIDGE_POINTS_ARRAY[cnt][a];
      SplineNrDownside=(Profil_Points-1)-cnt;
      temp_Point_Downside=Beamkontur_Object->GetPoint(SplineNrDownside);
      //println("Dazugehörigen Unterseitenpunkt gelesen BRIDGE_POINTS_ARRAY[",SplineNrDownside,"][",a,"]");
      temp_Height=temp_Point_Upside.y-temp_Point_Downside.y;
      if(abs(temp_Height) < Minimal_Construction_Height) temp_Height=Minimal_Construction_Height;
      Ideal_Beam_Heights[cnt][a]=temp_Height;
      temp_Point_Downside=BRIDGE_POINTS_ARRAY[cnt][a];
      temp_Point_Downside.y=temp_Point_Downside.y-temp_Height;
      BRIDGE_VOLUME_POINTS_ARRAY[SplineNrDownside][a]=temp_Point_Downside;
    }
  }
  if(Beam_Number_Intern==2) //Träger 2 besitzt einen Rechtecksquerschnitt! Träger 3 wird später konkarv gezeichnet, daher muss fü
Rechtecksquerschnitt hier in einen konkaven Träger umgerechnet werden.
  {
    println("Träger 2 besitzt einen Rechtecksquerschnitt! Träger 3 wird später konkarv gezeichnet, daher muss für die Volumensberech
hier in einen konkaven Träger umgerechnet werden.");
    if(Section_Shape_Mode!=0)
    {
      MAKE_PARABLE_KONTUR(Ideal_Beam_Heights); //Man bekommt am ende die Veränderten "Ideal_Beam_Heights[cnt][a]" zurück
    }
    //Volumshöhenpunkte zuweisen
    var Profil_Punkte=2*Anzahl_Splines;
    for (a=0;a<SplinePunkte;a++)
    {
      for (cnt=0;cnt<Anzahl_Splines;cnt++)
      {
        BRIDGE_VOLUME_POINTS_ARRAY[cnt][a]=BRIDGE_POINTS_ARRAY[cnt][a];
        BRIDGE_VOLUME_POINTS_ARRAY[(Profil_Punkte-1)-cnt][a]=BRIDGE_POINTS_ARRAY[cnt][a];
        BRIDGE_VOLUME_POINTS_ARRAY[(Profil_Punkte-1)-cnt][a].y=BRIDGE_VOLUME_POINTS_ARRAY[(Profil_Punkte-1)-cnt][a].y-Id
        //println("HÖHEN->Ideal_Beam_Heights[",cnt,"][",a,"]:",Ideal_Beam_Heights[cnt][a]);
      }
    }
  }
}

//CHOOSE_FITTEST_BEAM
CHOOSE_FITTEST_BEAM()
{
  println(" ");
  println("Der fittetste Träger wird ausgewählt!");
  println("-----------------------------------------");
  var Temp_Beam_Number,a;
  var temp_Nr=Beam_Number_Intern+1;
  var Beam_Compare_Data=new(array,temp_Nr,3); //Beam_Compare_Date[Beam_Number,(0:Beam_Bending,1:Beam_Volume,2:Fit,No
  var Bending_Arm_1,Bending_Arm_2,Max_Bending_Arm,temp_Beam_Volume, max_Bending_Arm, temp_Volume;
  var temp_Value=0.0;
  var Status="NOT FIT";

  println("Beam_Number_Intern: ",Beam_Number_Intern);
  if(Beam_Number_Intern<=4)
  {
    println("Beam_Number: ",Beam_Number_Intern+1,". Es sind erst weniger als 4 Träger erzeugt worden, es wird einfach der Vorgäng
" mutiert.");
    Fittest_Beam_Number=Beam_Number_Intern;
  }

  //Alle Träger Daten einlesen (Durchbiegungen & Volumen)
  if(Beam_Number_Intern>=5)
```

```
{
  println("Es wird der aus ",Beam_Number_Intern-3," Trägern ausgesucht");
  println("Werte aus 3d Objekten auslesen");
  for(Temp_Beam_Number=4;Temp_Beam_Number<=Beam_Number_Intern;Temp_Beam_Number++)
  {
    println("Träger ",Temp_Beam_Number," wird überprüft");
    if(doc->FindObject("Beam_"+tostring(Temp_Beam_Number)+"_Bending_of_Arm_1_Efficiency_Extern"))
    {
      var Bending_of_Arm_1_Efficiency_Extern_Object;
      Bending_of_Arm_1_Efficiency_Extern_Object=doc->FindObject("Beam_"+tostring(Temp_Beam_Number)+
      Bending_Arm_1=Bending_of_Arm_1_Efficiency_Extern_Object#PRIM_SPHERE_RAD;
    }
    if(doc->FindObject("Beam_"+tostring(Temp_Beam_Number)+"_Bending_of_Arm_2_Efficiency_Extern"))
    {
      var Bending_of_Arm_2_Efficiency_Extern_Object;
      Bending_of_Arm_2_Efficiency_Extern_Object=doc->FindObject("Beam_"+tostring(Temp_Beam_Number)+
      Bending_Arm_2=Bending_of_Arm_2_Efficiency_Extern_Object#PRIM_SPHERE_RAD;
    }
    if(Bending_Arm_1>=Bending_Arm_2)
    {
      Max_Bending_Arm=Bending_Arm_1;
    }
    else Max_Bending_Arm=Bending_Arm_2;

    println(Temp_Beam_Number, " Max_Bending_Arm: ",Max_Bending_Arm);
    Beam_Compare_Data[Temp_Beam_Number][0]=Max_Bending_Arm;

    if(doc->FindObject("Beam_"+tostring(Temp_Beam_Number)+"_Entire_Structure_Volume"))
    {
      var Beam_Structure_Volume_Object;
      Beam_Structure_Volume_Object=doc->FindObject("Beam_"+tostring(Temp_Beam_Number)+"_Entire_Str
      temp_Volume=Beam_Structure_Volume_Object#PRIM_SPHERE_RAD;
      Beam_Compare_Data[Temp_Beam_Number][1]= temp_Volume;
    }
    println("temp_Volume: ",temp_Volume);
  }//for(Temp_Beam_Number=4;Temp_Beam_Number<=Beam_Number_Intern;Temp_Beam_Number++) End
  //Trägervergleichen und besten Wählen.
  for(Temp_Beam_Number=4;Temp_Beam_Number<=Beam_Number_Intern;Temp_Beam_Number++)
  {
    Beam_Compare_Data[Temp_Beam_Number][2]="NOT FIT";
  }

  println("Werte sortieren");
  for(Temp_Beam_Number=4;Temp_Beam_Number<Beam_Number_Intern;Temp_Beam_Number++)
  {
    println("Träger ",Temp_Beam_Number);
    temp_Value=Beam_Compare_Data[Temp_Beam_Number][0];
    //println("Träger ",Temp_Beam_Number," hat kurz vor der Entscheidung die Bewertung: ",Beam_Compare.
Wert:",temp_Value);
    if(temp_Value<=1.0)
    {
     Status="FIT";
     println("Träger ",Temp_Beam_Number," wurde als FIT erkannt!");
     Beam_Compare_Data[Temp_Beam_Number][2]=Status; //Duchbiegung im zulässigen Bereich
    }
    else
    {
      Status="NOT FIT";
      println("Träger ",Temp_Beam_Number," wurde als NOT FIT erkannt!");
      Beam_Compare_Data[Temp_Beam_Number][2]=Status;
    }
  }
  var Smallest_Beam_Volume=100000000;
  var temp_Volume=0;
  var temp_Name;
  println("Es wird aus den fittesten Träger ausgewählt.");
  for(Temp_Beam_Number=4;Temp_Beam_Number<=Beam_Number_Intern;Temp_Beam_Number++)
  {
    //Nun wird der Fitteste Träger unter den Fitten ausgewählt.
    //Der mit der Geringsten Durchbiegung ist am Effizientesten
    println("Ist Träger ",Temp_Beam_Number," fit?");
    temp_Volume=Beam_Compare_Data[Temp_Beam_Number][1];
    temp_Name=Beam_Compare_Data[Temp_Beam_Number][2];
    if(temp_Name=="FIT")
    {
      println("Träger ",Temp_Beam_Number," ist fit und wird auf sein Volumen überprüft.");
      if(Smallest_Beam_Volume>=temp_Volume)
      {
        Smallest_Beam_Volume=temp_Volume; //Aktuellkleinstes Volumen mit dem aktuellen Volumen vergleich
        Fittest_Beam_Number=Temp_Beam_Number;
      }
    }
  }
 if(Smallest_Beam_Volume==100000000)
 {
   Fittest_Beam_Number=Beam_Number_Intern; //Wenn kein Smallest vergeben wurde, dann gibt es keine fitten. Es wird
   println("ES GIBT NICHT WIRKLICH EINEN FITTEN TRÄGER -> Es wird einfach weiter gemacht!");
 }
 println("Smallest_Beam_Volume=",Smallest_Beam_Volume);
 println("Fittester Träger ist Träger Nr.:",Fittest_Beam_Number);
 }
}

//MAKE_VARIABLE_FOLDER
WRITE_VARIABLE_FOLDER()
{
 println(" ");
 println("VARIABLEN IN ORDNER SCHREIBEN");

 //NullObjekte als Ordner ERSTELLEN
 var a;
 var STRUCTURE_NullObject;
 var Variable_Container_Folder_Object;
 var Variable_Container_Name=("VARIABLE_CONTAINER");
 var Position=vector(0,0,0);
 STRUCTURE_NullObject=doc->FindObject("STRUCTURE_CALC");
 if(!doc->FindObject("VARIABLE_CONTAINER"))
 {
  STRUCTURE_NullObject=doc->FindObject("STRUCTURE_CALC");
  Variable_Container_Folder_Object=new(NullObject);
  Variable_Container_Folder_Object->SetName(Variable_Container_Name);
  Variable_Container_Folder_Object#ID_BASEOBJECT_VISIBILITY_EDITOR=False;
  Variable_Container_Folder_Object#ID_BASEOBJECT_VISIBILITY_RENDER=False;
  doc->InsertObject(Variable_Container_Folder_Object,STRUCTURE_NullObject,Null);
 }
 Variable_Container_Folder_Object=doc->FindObject("VARIABLE_CONTAINER");
 var OG_Height_Percent_Name=("Beam_"+tostring(Beam_Number_Intern)+"_OG_Height_Percent");
 MAKE_SPHERE(OG_Height_Percent,OG_Height_Percent_Name,Position,Variable_Container_Folder_Object);
 var UG_Height_Percent_Name=("Beam_"+tostring(Beam_Number_Intern)+"_OG_Height_Percent");
 MAKE_SPHERE(UG_Height_Percent,UG_Height_Percent_Name,Position,Variable_Container_Folder_Object);
 var Change_Bridge_Thickness_Name=("Beam_"+tostring(Beam_Number_Intern)+"_Change_Bridge_Thickness");
 MAKE_SPHERE(Change_Bridge_Thickness,Change_Bridge_Thickness_Name,Position,Variable_Container_Folder_Object
 var  Torque_Curve_Calc_Mode_Name=("Beam_"+tostring(Beam_Number_Intern)+"_Torque_Curve_Calc_Mode");
 MAKE_SPHERE(Torque_Curve_Calc_Mode,Torque_Curve_Calc_Mode_Name,Position,Variable_Container_Folder_Object
 var  Formula_Divident_Arm_1_Extern_Name=("Beam_"+tostring(Beam_Number_Intern)+"_Formula_Divident_Arm_1_Ex
 MAKE_SPHERE(Formula_Divident_Arm_1_Extern,Formula_Divident_Arm_1_Extern_Name,Position,Variable_Container_F
 var  Formula_Divident_Arm_2_Extern_Name=("Beam_"+tostring(Beam_Number_Intern)+"_Formula_Divident_Arm_2_E
 MAKE_SPHERE(Formula_Divident_Arm_2_Extern,Formula_Divident_Arm_2_Extern_Name,Position,Variable_Container_
 var  Bending_of_Arm_1_Efficiency_Extern_Name=("Beam_"+tostring(Beam_Number_Intern)+"_Bending_of_Arm_1_Effic
 MAKE_SPHERE(Bending_of_Arm_1_Efficiency,Bending_of_Arm_1_Efficiency_Extern_Name,Position,Variable_Container_
 var  Bending_of_Arm_2_Efficiency_Extern_Name=("Beam_"+tostring(Beam_Number_Intern)+"_Bending_of_Arm_2_Eff
 MAKE_SPHERE(Bending_of_Arm_2_Efficiency,Bending_of_Arm_2_Efficiency_Extern_Name,Position,Variable_Containe
 var  Faktor_1_Enable_Extern_Name=("Beam_"+tostring(Beam_Number_Intern)+"_Faktor_2_Enable_Extern");
 MAKE_SPHERE(Faktor_1_Enable_Extern,Faktor_1_Enable_Extern_Name,Position,Variable_Container_Folder_Object);
 var  Faktor_1_Enable_Extern_Name=("Beam_"+tostring(Beam_Number_Intern)+"_Faktor_2_Enable_Extern");
 MAKE_SPHERE(Faktor_1_Enable_Extern,Faktor_1_Enable_Extern_Name,Position,Variable_Container_Folder_Object);

 var  Section_Shape_Mode_Extern_Name=("Beam_"+tostring(Beam_Number_Intern)+"_Section_Shape_Mode_Extern");
 MAKE_SPHERE(Section_Shape_Mode_Extern,Section_Shape_Mode_Extern_Name,Position,Variable_Container_Folder_
 var  Entire_Structure_Volume_Name=("Beam_"+tostring(Beam_Number_Intern)+"_Entire_Structure_Volume");
 MAKE_SPHERE(Entire_Structure_Volume,Entire_Structure_Volume_Name,Position,Variable_Container_Folder_Object);

 var OG_Efficiency_Value,OG_Efficiency_Name;
 for (a=0;a<SplinePunkte;a++) //Einzelne Schnittewerte übergeben;
 {
  OG_Efficiency_Name=("Beam_"+tostring(Beam_Number_Intern)+"_"+tostring(a)+"_OG_Efficiency");
  OG_Efficiency_Value=OG_Efficiency[a];
  MAKE_SPHERE(OG_Efficiency_Value,OG_Efficiency_Name,Position,Variable_Container_Folder_Object);
 }

 var UG_Efficiency_Value,UG_Efficiency_Name;
 for (a=0;a<SplinePunkte;a++) //Einzelne Schnittewerte übergeben;
 {
  UG_Efficiency_Name=("Beam_"+tostring(Beam_Number_Intern)+"_"+tostring(a)+"_UG_Efficiency");
  UG_Efficiency_Value=UG_Efficiency[a];
  MAKE_SPHERE(UG_Efficiency_Value,UG_Efficiency_Name,Position,Variable_Container_Folder_Object);
 }
 var Section_Bending_Efficiency_Value,Section_Bending_Efficiency_Name;
 for (a=0;a<SplinePunkte;a++) //Einzelne Schnittewerte übergeben;
 {
  Section_Bending_Efficiency_Name=("Beam_"+tostring(Beam_Number_Intern)+"_"+tostring(a)+"_Section_Bending_Effic
  Section_Bending_Efficiency_Value=Section_Bending_Efficiency[a];
  MAKE_SPHERE(Section_Bending_Efficiency_Value,Section_Bending_Efficiency_Name,Position,Variable_Container_Fol
 }
 var Section_Balance_Value=1,Section_Balance_Value_Name;
 var Section_Balance_Extern;
 for (a=0;a<SplinePunkte;a++) //Einzelne Schnittewerte übergeben;
 {
```

```
    Section_Balance_Extern=Section_Area_Balance_Point[a];
    Section_Balance_Value_Name=("Beam_"+tostring(Beam_Number_Intern)+"_"+tostring(a)+"_Section_Balan
    MAKE_SPHERE(Section_Balance_Value,Section_Balance_Value_Name,Section_Balance_Extern,Variable_C
  }
}

//READ_VARIABLE_FOLDER
READ_VARIABLE_FOLDER()
{
  println(" ");
  println("VARIABLEN AUS ORDNER LESEN");

  var a;

  if(doc->FindObject("Beam_"+tostring(Fittest_Beam_Number)+"_OG_Height_Percent"))
  {
    var OG_Height_Percent_Object;
    OG_Height_Percent_Object=doc->FindObject("Beam_"+tostring(Fittest_Beam_Number)+"_OG_Height_Perc
    OG_Height_Percent=OG_Height_Percent_Object#PRIM_SPHERE_RAD;
  }
  else
  {
    if(Material_Typ==0) OG_Height_Percent=0.05;
    if(Material_Typ==1) OG_Height_Percent=0.10;
  }

  //println("OG_Height_Percent: ",OG_Height_Percent);
  if(doc->FindObject("Beam_"+tostring(Fittest_Beam_Number)+"_UG_Height_Percent"))
  {
    var UG_Height_Percent_Object;
    UG_Height_Percent_Object=doc->FindObject("Beam_"+tostring(Fittest_Beam_Number)+"_UG_Height_Perc
    UG_Height_Percent=UG_Height_Percent_Object#PRIM_SPHERE_RAD;
  }
  else
  {
    if(Material_Typ==0) UG_Height_Percent=0.1;
    if(Material_Typ==1) UG_Height_Percent=0.30;
  }

  //println("UG_Height_Percent: ",UG_Height_Percent);
  if(doc->FindObject("Beam_"+tostring(Fittest_Beam_Number)+"_Change_Bridge_Thickness"))
  {
    var Change_Bridge_Thickness_Object;
    Change_Bridge_Thickness_Object=doc->FindObject("Beam_"+tostring(Fittest_Beam_Number)+"_Change_B
    Change_Bridge_Thickness=Change_Bridge_Thickness_Object#PRIM_SPHERE_RAD;
  }

  //println("Change_Bridge_Thickness: ",Change_Bridge_Thickness);
  if(doc->FindObject("Beam_"+tostring(Fittest_Beam_Number)+"Torque_Curve_Calc_Mode"))
  {
    var Torque_Curve_Calc_Mode_Object;
    Torque_Curve_Calc_Mode_Object=doc->FindObject("Beam_"+tostring(Fittest_Beam_Number)+"_Torque_C
    Torque_Curve_Calc_Mode=Torque_Curve_Calc_Mode_Object#PRIM_SPHERE_RAD;
  }

  if(doc->FindObject("Beam_"+tostring(Fittest_Beam_Number)+"_Formula_Divident_Arm_1_Extern"))
  {
    var Formula_Divident_Arm_1_Extern_Object;
    Formula_Divident_Arm_1_Extern_Object=doc->FindObject("Beam_"+tostring(Fittest_Beam_Number)+"_For
    Formula_Divident_Arm_1_Extern=Formula_Divident_Arm_1_Extern_Object#PRIM_SPHERE_RAD;
  }

  if(doc->FindObject("Beam_"+tostring(Fittest_Beam_Number)+"_Formula_Divident_Arm_2_Extern"))
  {
    var Formula_Divident_Arm_2_Extern_Object;
    Formula_Divident_Arm_2_Extern_Object=doc->FindObject("Beam_"+tostring(Fittest_Beam_Number)+"_Fo
    Formula_Divident_Arm_2_Extern=Formula_Divident_Arm_2_Extern_Object#PRIM_SPHERE_RAD;
  }

  if(doc->FindObject("Beam_"+tostring(Fittest_Beam_Number)+"_Bending_of_Arm_1_Efficiency_Extern"))
  {
    var Bending_of_Arm_1_Efficiency_Extern_Object;
    Bending_of_Arm_1_Efficiency_Extern_Object=doc->FindObject("Beam_"+tostring(Fittest_Beam_Number)+"
    Bending_of_Arm_1_Efficiency=Bending_of_Arm_1_Efficiency_Extern_Object#PRIM_SPHERE_RAD;
  }

  if(doc->FindObject("Beam_"+tostring(Fittest_Beam_Number)+"_Bending_of_Arm_2_Efficiency_Extern"))
  {
    var Bending_of_Arm_2_Efficiency_Extern_Object;
    Bending_of_Arm_2_Efficiency_Extern_Object=doc->FindObject("Beam_"+tostring(Fittest_Beam_Number)+"
    Bending_of_Arm_2_Efficiency=Bending_of_Arm_2_Efficiency_Extern_Object#PRIM_SPHERE_RAD;
  }

  if(doc->FindObject("Beam_"+tostring(Fittest_Beam_Number)+"_Faktor_1_Enable_Extern"))
  {
    var Faktor_1_Enable_Extern_Object;
    Faktor_1_Enable_Extern_Object=doc->FindObject("Beam_"+tostring(Fittest_Beam_Number)+"_Faktor_1_Enable_Extern");
    Faktor_1_Enable_Extern=Faktor_1_Enable_Extern_Object#PRIM_SPHERE_RAD;
  }

  if(doc->FindObject("Beam_"+tostring(Fittest_Beam_Number)+"_Faktor_2_Enable_Extern"))
  {
    var Faktor_2_Enable_Extern_Object;
    Faktor_2_Enable_Extern_Object=doc->FindObject("Beam_"+tostring(Fittest_Beam_Number)+"_Faktor_2_Enable_Extern");
    Faktor_2_Enable_Extern=Faktor_2_Enable_Extern_Object#PRIM_SPHERE_RAD;
  }

  if(doc->FindObject("Beam_"+tostring(Fittest_Beam_Number)+"_Section_Shape_Mode_Extern"))
  {
    var Section_Shape_Mode_Extern_Object;
    Section_Shape_Mode_Extern_Object=doc->FindObject("Beam_"+tostring(Fittest_Beam_Number)+"_Section_Shape_Mode_Extern");
    Section_Shape_Mode_Extern=Section_Shape_Mode_Extern_Object#PRIM_SPHERE_RAD;
  }

  //println("Torque_Curve_Calc_Mode: ",Torque_Curve_Calc_Mode);
  var OG_Efficiency_Name,OG_Efficiency_Object;
  for (a=0;a<SplinePunkte;a++) //Einzelne Schnittewerte übergeben;
  {
    OG_Efficiency_Name=("Beam_"+tostring(Fittest_Beam_Number)+"_"+tostring(a)+"_OG_Efficiency");
    OG_Efficiency_Object=doc->FindObject(OG_Efficiency_Name);
    OG_Efficiency[a]=OG_Efficiency_Object#PRIM_SPHERE_RAD;
    //println("OG_Efficiency[",a,"] ",OG_Efficiency[a]);
  }

  var UG_Efficiency_Name,UG_Efficiency_Object;
  for (a=0;a<SplinePunkte;a++) //Einzelne Schnittewerte übergeben;
  {
    UG_Efficiency_Name=("Beam_"+tostring(Fittest_Beam_Number)+"_"+tostring(a)+"_UG_Efficiency");
    UG_Efficiency_Object=doc->FindObject(UG_Efficiency_Name);
    UG_Efficiency[a]=UG_Efficiency_Object#PRIM_SPHERE_RAD;
    //println("UG_Efficiency[",a,"] ",UG_Efficiency[a]);
  }

  var UG_Efficiency_Name,UG_Efficiency_Object;
  for (a=0;a<SplinePunkte;a++) //Einzelne Schnittewerte übergeben;
  {
    UG_Efficiency_Name=("Beam_"+tostring(Fittest_Beam_Number)+"_"+tostring(a)+"_UG_Efficiency");
    UG_Efficiency_Object=doc->FindObject(UG_Efficiency_Name);
    UG_Efficiency[a]=UG_Efficiency_Object#PRIM_SPHERE_RAD;
    //println("UG_Efficiency[",a,"] ",UG_Efficiency[a]);
  }

  var Section_Bending_Efficiency_Name,Section_Bending_Efficiency_Object;
  for (a=0;a<SplinePunkte;a++) //Einzelne Schnittewerte übergeben;
  {
    Section_Bending_Efficiency_Name=("Beam_"+tostring(Fittest_Beam_Number)+"_"+tostring(a)+"_Section_Bending_Efficiency");
    Section_Bending_Efficiency_Object=doc->FindObject(Section_Bending_Efficiency_Name);
    Section_Bending_Efficiency[a]=Section_Bending_Efficiency_Object#PRIM_SPHERE_RAD;
    //println("Section_Bending_Efficiency[",a,"] ", Section_Bending_Efficiency[a]);
  }
  for (a=0;a<SplinePunkte;a++) //Einzelne Schnittewerte übergeben;
  {
    Section_Area_Balance_Point[a]=vector(0,0,0);
  }

  var Section_Balance_Extern_Name,Section_Balance_Coord_Object,Section_Balance_Coord=vector(0,0,0);
  for (a=0;a<SplinePunkte;a++) //Einzelne Schnittewerte übergeben;
  {
    Section_Balance_Extern_Name=("Beam_"+tostring(Fittest_Beam_Number)+"_"+tostring(a)+"_Section_Balance_Extern");
    Section_Balance_Coord_Object=doc->FindObject(Section_Balance_Extern_Name);
    Section_Balance_Coord=Section_Balance_Coord_Object->GetPosition();
    Section_Area_Balance_Point[a]=Section_Balance_Coord;
    //println("Section_Area_Balance_Point[",a,"] ",Section_Area_Balance_Point[a]);
  }
}

//Vom tiefsten Uferpunkt das Wasserlevel abziehn
WATERLEVEL_POINT_CALC()
{
  println(" ");
  println("WASSERPEGEL MESSPUNKT BERECHNEN!");
  var a,cnt,i;
  var current_Point_Ufer_A,current_Point_Ufer_B;
  var current_deepest_Point;
  var PunktNr,SplineNr;
  //Den tiefsten Uferpunkt berechnen;
  current_deepest_Point=BRIDGE_POINTS_ARRAY[0][0];
  for (cnt=0;cnt<=Anzahl_Splines-1;cnt++)
```

```
{
  current_Point_Ufer_A=BRIDGE_POINTS_ARRAY[cnt][0];
  current_Point_Ufer_B=BRIDGE_POINTS_ARRAY[cnt][SplinePunkte-1];
  //Wenn der aktuelle y.Wert kleiner ist als der vorhergende, dann ist aktueller Punkt der neue Tiefpunkt
  if(current_Point_Ufer_A.y<=current_deepest_Point.y)
  {
    current_deepest_Point=current_Point_Ufer_A;
    Waterlevel_Point=current_Point_Ufer_A;
    Waterlevel_Point.y=Waterlevel_Point.y+Waterlevel;
    PunktNr=SplinePunkte-1;
    SplineNr=cnt;
  }
  if(current_Point_Ufer_B.y<=current_deepest_Point.y)
  {
    current_deepest_Point=current_Point_Ufer_B;
    Waterlevel_Point=current_Point_Ufer_B;
    Waterlevel_Point.y=Waterlevel_Point.y+Waterlevel;
    PunktNr=0;
    SplineNr=cnt;
  }
 println("Der Wasserpegel wird unter der Spline ",SplineNr," und dem Punkt ",PunktNr," an der Stelle ",Waterlev
}

//CHANGE_Bridge_SHAPE...CheckWaterlebel ect...
CHECK_AND_CHANGE_BRIDGE_SHAPE()
{
 println(" ");
 println("CHECK AND CHANGE BRIDGE_SHAPE");
 var a,cnt;
 var Beamkontur_Object;
 var Beamkontur_Name;
 var Changed_Point,temp_Point_Upside,temp_Point_Downside;
 var temp_Height_Upside,temp_Height_Downside, temp_High;
 var Profile_Points=Anzahl_Splines*2;
 var SplineDownSide;
 //Die Brücken dicke wird eventuel verändert;
 var Change_Faktor_Upside=1;
 var Change_Faktor_Downside=1;
 var Zufall, Zufallszahl_1,Zufallszahl_2,Zufallszahl_3,Cross_Efficiency;
 var Water_Dist;
 Zufall=new(Random);
 Zufall->Init(time());
 Zufallszahl_1=Zufall->Get01();
 Zufallszahl_2=Zufall->Get01();
 Zufallszahl_3=Zufall->Get01();
 var Fittest_Bending, Max_Efficieny;

 if(Bending_of_Arm_1_Efficiency>Bending_of_Arm_2_Efficiency)  Max_Efficieny=Bending_of_Arm_1_Efficien
 if(Bending_of_Arm_1_Efficiency<=Bending_of_Arm_2_Efficiency)  Max_Efficieny=Bending_of_Arm_2_Efficie

 if(Max_Efficieny>=1) Max_Efficieny=Max_Efficieny*1.3;
 if(Max_Efficieny<1) Max_Efficieny=Max_Efficieny*0.7;

 //Höhe des Trägers ändern
 if(Zufallszahl_2<=0.5)
 {
  for (a=0;a<SplinePunkte;a++)
  {
   Fittest_Bending=1+(1-(Section_Bending_Efficiency[SplinePunkte-a-1]));
   if(0.2<Zufallszahl_1<0.8)  Fittest_Bending=Fittest_Bending*Zufallszahl_1;
   //println("Fittest_Bending nach Zufall: ",Fittest_Bending);
   for (cnt=0;cnt<Anzahl_Splines;cnt++)
   {
    //println("Wegpunkte mutieren");
    SplineDownSide=(Profile_Points-1)-cnt;
    temp_Point_Upside=BRIDGE_VOLUME_POINTS_ARRAY[cnt][a];
    temp_Point_Downside=BRIDGE_VOLUME_POINTS_ARRAY[SplineDownSide][a];
    temp_Height_Upside=temp_Point_Upside.y-Section_Area_Balance_Point[a].y;
    temp_Point_Upside.y=Section_Area_Balance_Point[a].y+(temp_Height_Upside*Fittest_Bending*Max_Effi
    temp_Height_Downside=Section_Area_Balance_Point[a].y-temp_Point_Downside.y;
    temp_Point_Downside.y=Section_Area_Balance_Point[a].y-(temp_Height_Downside*Fittest_Bending*Ma
    temp_High=abs(temp_Point_Upside.y-temp_Point_Downside.y);
    //println("temp_Point_Upside.y: ",temp_Point_Upside.y);
    //println("temp_Point_Downside.y: ",temp_Point_Downside.y);
    //println("temp_High: ",temp_High);
    if(temp_High<Minimal_Construction_Height)
    {
     temp_High=Minimal_Construction_Height;
     temp_Point_Upside=BRIDGE_VOLUME_POINTS_ARRAY[cnt][a];
     temp_Point_Downside.y=temp_Point_Upside.y-temp_High;
     BRIDGE_VOLUME_POINTS_ARRAY[SplineDownSide][a]=temp_Point_Downside;
     //println("temp_Point_Upside.y: ",temp_Point_Upside.y);
     //println("temp_Point_Downside.y: ",temp_Point_Downside.y);
     //println("temp_High: ",temp_High);
    }
    if(Path_Area_Point[cnt][a]==0) //Veränderungen nach oben nur wenn keine Wege vorhanden sind.
    {
     if(Waterlevel_Point.y>temp_Point_Downside.y)
     {
      Water_Dist=abs(Waterlevel_Point.y-temp_Point_Downside.y);
      temp_Point_Upside.y=temp_Point_Upside.y*1.1+Water_Dist;
      temp_Point_Downside.y=temp_Point_Downside.y+Water_Dist;
     }
     BRIDGE_VOLUME_POINTS_ARRAY[cnt][a]=temp_Point_Upside;
    }
    if(0.2<Zufallszahl_3)
    {
     if(Waterlevel_Point.y>temp_Point_Downside.y) temp_Point_Downside.y=Waterlevel_Point.y; //erzwingt die Wasser
    }
    BRIDGE_VOLUME_POINTS_ARRAY[SplineDownSide][a]=temp_Point_Downside;
   }
  }
 }
 else //Aushöhlgrad ändern;
 {
  println("Aushöhlgrad ändern");
  if(Zufallszahl_1<0.3)  Max_Efficieny=Max_Efficieny*0.8;
  if(Zufallszahl_1<0.7)  Max_Efficieny=Max_Efficieny*1.2;
  if(0.3<=Zufallszahl_1<=0.7)  Max_Efficieny=Max_Efficieny;
  OG_Height_Percent=OG_Height_Percent*Max_Efficieny;//0.12
  UG_Height_Percent= Right_Width_Percent*Max_Efficieny;
  Left_Width_Percent=Right_Width_Percent*Max_Efficieny; //Der Wert darf nicht höher als über 0.5 sein
  Right_Width_Percent= Right_Width_Percent*Max_Efficieny;//Der Wert darf nicht höher als über 0.5 sein
  if(OG_Height_Percent>=0.5) OG_Height_Percent=0.49;
  if( UG_Height_Percent>=0.5)  UG_Height_Percent=0.49;
  if( Left_Width_Percent>=0.5)  Left_Width_Percent=0.49;
  if( Right_Width_Percent>=0.5) Right_Width_Percent=0.49;
 }
}

//SURFACE_CALC
BRIDGE_SURFACE_CALC()
{
 println(" ");
 println("BRÜCKENFLÄCHEN BRECHNUNG");
 var Plane_Triangle1,Plane_Triangle2,SinglePlane,Half_Outline; //Half_Outline = HalberUmfang
 var AB_Lenght,BC_Lenght,CD_Lenght,DA_Lenght,AC_Diagonale_Lenght; // Seitenlängen
 var A_Point,B_Point,C_Point,D_Point;
 var AC_PointSUM;
 var a,c=0,cnt,SectionNum,Section_PlaneNum;
 Planes_Count=(Anzahl_Splines-1)*(SplinePunkte-1);
 SinglePlane_BalancePoint=new (array,Anzahl_Splines-1,SplinePunkte-1); //Teilflächenschwerpunkte
 SinglePlanes=new (array,Anzahl_Splines-1,SplinePunkte-1);
 Sector_Plane=new (array,SplinePunkte-1);
 EntirePlane=0;
 SectionNum=SplinePunkte-1; //SchnittNr.
 Section_PlaneNum=Anzahl_Splines-1; //Punkte je Schnitt

 for (cnt=0;cnt<Section_PlaneNum;cnt++) //SplineNummer
 {
  for (a=0;a<SectionNum;a++) //PunktNr
  {
   //Distanz zwischen den Punkten ausrechnen
   A_Point=BRIDGE_POINTS_ARRAY[cnt][a];
   B_Point=BRIDGE_POINTS_ARRAY[cnt][a+1];
   C_Point=BRIDGE_POINTS_ARRAY[cnt+1][a+1];
   D_Point=BRIDGE_POINTS_ARRAY[cnt+1][a];
   XYZ_Dist_2_Points_Calc(A_Point,C_Point);
   AC_Diagonale_Lenght=Dist_current_Points;
   XYZ_Dist_2_Points_Calc(A_Point,B_Point);
   AB_Lenght=Dist_current_Points;
   XYZ_Dist_2_Points_Calc(B_Point,C_Point);
   BC_Lenght=Dist_current_Points;
   XYZ_Dist_2_Points_Calc(C_Point,D_Point);
   CD_Lenght=Dist_current_Points;
   XYZ_Dist_2_Points_Calc(D_Point,A_Point);
   DA_Lenght=Dist_current_Points;
   //Dreieck 1
   Half_Outline=(AB_Lenght+BC_Lenght+AC_Diagonale_Lenght)/2;
   Plane_Triangle1=sqrt(Half_Outline*(Half_Outline-AB_Lenght)*(Half_Outline-BC_Lenght)*(Half_Outline-AC_Diagonale
   //Dreieck 2
   Half_Outline=(AC_Diagonale_Lenght+CD_Lenght+DA_Lenght)/2;
   Plane_Triangle2=sqrt(Half_Outline*(Half_Outline-AC_Diagonale_Lenght)*(Half_Outline-CD_Lenght)*(Half_Outline-DA
   SinglePlane=Plane_Triangle1+Plane_Triangle2;
   EntirePlane=EntirePlane+SinglePlane;
   //EinzelFlächenschwerpunkt brechnen;
   SinglePlane_BalancePoint[cnt][a]=(vector(A_Point.x,A_Point.y,A_Point.z)+vector(C_Point.x,C_Point.y,C_Point.z))/2;
```

```
      SinglePlanes[cnt][a]=SinglePlane;
      /*
      println("Schwerpunkt ",cnt,"|",a," Koordinaten:",SinglePlane_BalancePoint[cnt][a]);
      println(" ");
      println("SEKTOR: ",a," | Fläche: ",cnt);
      println("Punkt_A: ",A_Point);
      println("Punkt_B: ",B_Point);
      println("Punkt_C: ",C_Point);
      println("Punkt_D: ",D_Point);
      println("AC_Diagonale_Lenght: ",AC_Diagonale_Lenght);
      println("AB: ",AB_Lenght);
      println("BC: ",BC_Lenght);
      println("CD: ",CD_Lenght);
      println("DA: ",DA_Lenght);
      println("Fläche 1.Dreieck: ",Plane_Triangle1);
      println("Fläche 2.Dreieck: ",Plane_Triangle2);
      println("SinglePlane: ",SinglePlane);
      println("Momentane Gesamtfläche: ",EntirePlane);
      */
    }//SplinePunkteSchleifeEnde
  }//SplineAnzahlSchleifeEnde

  //Flächenschwerpunkt der Geometrie ausrechnen ->Vielleicht brauche ich den -> bei gleichlast eventuell inter
  println("Es gibt ",Planes_Count," Oberflächen_Rechtecke welche eine Gesamtfläche von ",EntirePlane/10000,"
  Surface_Balance_Point=vector(0,0,0);
  for (cnt=0;cnt<SectionNum;cnt++)
  {
    for (a=0;a<SectionNum;a++)
    {
      Surface_Balance_Point=Surface_Balance_Point+(SinglePlane_BalancePoint[cnt][a]*SinglePlanes[cnt][a]);
      /*
      println("Plane ",a,"|",cnt," Schwerpunkt: ",SinglePlane_BalancePoint[cnt][a]);
      println("Aktuelle SchwepunktSumme: ",Surface_Balance_Point);
      println(" ");
       */
    }
  }
  Surface_Balance_Point=Surface_Balance_Point/EntirePlane;
  return EntirePlane,SinglePlane_BalancePoint,Sector_Plane,SinglePlanes;
}

//TORQUE_CALC
LOAD_CALC()
{
  println(" ");
  println("LASTFÄLLE BERECHNEN!");
  var Path_Point_Crd,Path_1_Object,Path_2_Object,Path_3_Object;
  var a,cnt,i,current_Balance_Point_Crd;
  var Path_to_Balance_Point_Distance;
  var SamplingResolution,Sampling_Sections,Sampling_Error_Dist1,Sampling_Error_Dist2,Sampling_Error_Dis
  var Section_Percent_Step,Percent_Position=0.0;
  var SectionNum,Section_PlaneNum; //Schnittnummer, PlanenNummer
  var Path_Width, Path_1_Full_Lenght, Path_2_Full_Lenght, Path_3_Full_Lenght;
  var temp_Load_Balance_Point;
  SinglePlanes_Load=new(array,Anzahl_Splines-1,SplinePunkte-1);  //Schnitt / Punkte-Schnitt
  Structure_Load_NEW=new (array,Anzahl_Splines-1,SplinePunkte-1);//Eigengewicht Annahme
  Single_Volume_Cubes_Load=new(array,Anzahl_Splines-1,SplinePunkte-1);
  Single_Bridge_Balance_Points=new(array,Anzahl_Splines-1,SplinePunkte-1);
  Single_Bridge_Load_Points=new(array,Anzahl_Splines-1,SplinePunkte-1);
  SamplingResolution=(SplinePunkte)*2;//alte einstellung war 4 , desto niedriger der Wert desto schneller, weil
  Sampling_Sections=SamplingResolution/(SplinePunkte);
  Section_Percent_Step=1.0/SamplingResolution;
  if(doc->FindObject("Pedestrian_Path_Spline"))
  {
    Path_1_Object=doc->FindObject("Pedestrian_Path_Spline");
    Path_1_Object->InitLength(0);
    Path_1_Full_Lenght=Path_1_Object->GetLength();
    Sampling_Error_Dist1=(Path_1_Full_Lenght/SamplingResolution)/4; //weil Abtastraten-Fehler wird lichte er
  }
  if(doc->FindObject("Biker_Path_Spline"))
  {
    Path_2_Object=doc->FindObject("Biker_Path_Spline");
    Path_2_Object->InitLength(0);
    Path_2_Full_Lenght=Path_2_Object->GetLength();
    Sampling_Error_Dist2=(Path_2_Full_Lenght/SamplingResolution)/4; //weil Abtastraten-Fehler wird lichte e
  }
  if(doc->FindObject("Barrier_Free_Path_Spline"))
  {
    Path_3_Object=doc->FindObject("Barrier_Free_Path_Spline");
    Path_3_Object->InitLength(0);
    Path_3_Full_Lenght=Path_3_Object->GetLength();
    Sampling_Error_Dist3=(Path_3_Full_Lenght/SamplingResolution)/4; //weil Abtastraten-Fehler wird lichte e
  }
  SectionNum=SplinePunkte-1; //SchnittNr.
  Section_PlaneNum=Anzahl_Splines-1; //Punkte im Schnitt

  //EIGENGEWICHTSBERECHNUNG für jedes Volumen;
  //Falls keine Weglasten hinzukommen dann wird hier schonmal das Eigengewicht berechnet
  //Bemerkung -> Es wird immer alles in [cm] umgewandelt
  for (a=0;a<SectionNum;a++) //PunktNr
  {
    for (cnt=0;cnt<Section_PlaneNum;cnt++) //SplineNummer
    {
      //2500kg/m³ *10 = 25000N/m³/100 = 25kN/m³ ===Structure_Load/100 || cm³ / 1000000 = m³ = Sum_of_Pyramides_Volumes/10
      Single_Volume_Cubes_Load[cnt][a]=(Structure_Load/100)*(Sum_of_Pyramides_Volumes[cnt][a]/1000000);
      //println("Single_Volume_Cubes_Load[",cnt,"][",a,"]= ",Single_Volume_Cubes_Load[cnt][a]);
    }
  }
  //Gesamtes Konstruktionsgewicht
  if(Entire_Structure_Volume==0) Entire_Structure_Volume=0.000001;
  Entire_Structure_Load=(Structure_Load/100)*(Entire_Structure_Volume/1000000);
  //println("Das gesamte Brücken_Eigengewicht beträgt:",Entire_Structure_Load, "kN");
  //println("Das gesamte Brücken_Eigengewicht beträgt:",Entire_Structure_Load*0.1, "t");
  //Nutzlasten aud Nullsetzen
  for (a=0;a<SectionNum;a++) //PunktNr
  {
    for (cnt=0;cnt<Section_PlaneNum;cnt++) //SplineNummer
    {
      // 5.1kN/m² -> 0.051kN/dm²-> 0.00051kN/cm² = /10000
      SinglePlanes_Load[cnt][a]=0;
      //Path_Area_Point[cnt][a]=0;
      //println("SinglePlanes_Load[",cnt,"][",a,"]: ",SinglePlanes_Load[cnt][a],"[kN]");
    }
  }

  //Nutzlastberechnung
  println("Wege für die Lastermittlung suchen");
  //Wenn irgendein Schwerpunkt in Weglichte, dann Nutzlast addieren.
  //if SinglePlane_BalancePoint[cnt][a] distanz weglichte entfernung
  for (a=0;a<SectionNum;a++) //PunktNr
  {
    for (cnt=0;cnt<Section_PlaneNum;cnt++) //SplineNummer
    {
      current_Balance_Point_Crd=SinglePlane_BalancePoint[cnt][a];
      Percent_Position=0;
      for (i=0;i< SamplingResolution;i++)
      {
        if(doc->FindObject("Pedestrian_Path_Spline"))
        {
          Path_Point_Crd=Path_1_Object->GetSplinePoint(Path_1_Object->UniformToNatural(Percent_Position),0);
          XZ_Dist_2_Points_Calc(current_Balance_Point_Crd,Path_Point_Crd);
          Path_to_Balance_Point_Distance=abs(Dist_current_Points);
          Path_Width=Pedestrian_Width;
          if(Path_to_Balance_Point_Distance<=((Path_Width/2)+Sampling_Error_Dist1))
          {
            // 5.1kN/m² -> 0.051kN/dm²-> 0.00051kN/cm² = /10000
            SinglePlanes_Load[cnt][a]= (Service_Load/10000)*SinglePlanes[cnt][a]; //Möglicherweise Einheitenfehler...die Kraft würde ich
          }
        }
        if(doc->FindObject("Biker_Path_Spline"))
        {
          Path_Point_Crd=Path_2_Object->GetSplinePoint(Path_2_Object->UniformToNatural(Percent_Position),0);
          XZ_Dist_2_Points_Calc(current_Balance_Point_Crd,Path_Point_Crd);
          Path_to_Balance_Point_Distance=abs(Dist_current_Points);
          Path_Width=Biker_Width;
          if(Path_to_Balance_Point_Distance<=((Path_Width/2)+Sampling_Error_Dist2))
          {
            SinglePlanes_Load[cnt][a]= (Service_Load/10000)*SinglePlanes[cnt][a];
          }
        }
        if(doc->FindObject("Barrier_Free_Path_Spline"))
        {
          Path_Point_Crd=Path_3_Object->GetSplinePoint(Path_3_Object->UniformToNatural(Percent_Position),0);
          XZ_Dist_2_Points_Calc(current_Balance_Point_Crd,Path_Point_Crd);
          Path_to_Balance_Point_Distance=abs(Dist_current_Points);
          Path_Width=WheelChair_Width;
          if( Path_to_Balance_Point_Distance<=((Path_Width/2)+Sampling_Error_Dist3))
          {
            SinglePlanes_Load[cnt][a]=(Service_Load/10000)*SinglePlanes[cnt][a];
          }
        }
        Percent_Position=Percent_Position+Section_Percent_Step;
      }
    }
  }

  //-------------------------------------------------------------------------------------------------
  //Überprüfen, welche Brückenpunkte im Wegbereich sind;
```

```
//ACHTUNG: DAS KANN NICHT MIT OBERER SCHLEIFE ZUSAMMENGEFASST WERDEN,
//DA NUN NICHT DIE SCHWERPUNKTE ZÄHLEN SONDERN DIE MATRIX PUNKTE.
//DAS KOSTET SEHR VIEL ZEIT IN DER BERECHNUNG UND
//SOLLTE SPÄTER NOCHEINMAL ÜBERDACHT WERDEN
var current_Point_Crd;
/*
Sampling_Error_Dist1=Sampling_Error_Dist1*0.1;
Sampling_Error_Dist2=Sampling_Error_Dist2*0.1;
Sampling_Error_Dist3=Sampling_Error_Dist3*0.1;
*/
Path_Area_Point=new(array,Anzahl_Splines,SplinePunkte);
for (cnt=0;cnt<Anzahl_Splines;cnt++) //PunktNr
{
  for (a=0;a<SplinePunkte;a++) //SplineNummer
  {
    Path_Area_Point[cnt][a]=0; //Brückenpunkt ist nicht i Wege bereich
  }
}
if(doc->FindObject("Pedestrian_Path_Spline")) Sampling_Error_Dist1=Sampling_Error_Dist1*1.15;
if(doc->FindObject("Biker_Path_Spline")) Sampling_Error_Dist2=Sampling_Error_Dist2*1.15;
if(doc->FindObject("Barrier_Free_Path_Spline")) Sampling_Error_Dist3=Sampling_Error_Dist3*1.15;
for (a=0;a<SplinePunkte;a++) //PunktNr
{
  for (cnt=0;cnt<Anzahl_Splines;cnt++) //SplineNummer
  {
    current_Point_Crd=BRIDGE_POINTS_ARRAY[cnt][a];
    Percent_Position=0;
    for (i=0;i<SamplingResolution;i++)
    {
      if(doc->FindObject("Pedestrian_Path_Spline"))
      {
        Path_Point_Crd=Path_1_Object->GetSplinePoint(Path_1_Object->UniformToNatural(Percent_Posit
        XZ_Dist_2_Points_Calc(current_Point_Crd,Path_Point_Crd);
        Path_to_Balance_Point_Distance=abs(Dist_current_Points);
        Path_Width=Pedestrian_Width;
        if(Path_to_Balance_Point_Distance<=((Path_Width/2)+Sampling_Error_Dist1))
        {
          Path_Area_Point[cnt][a]=1;
        }
      }
      if(doc->FindObject("Biker_Path_Spline"))
      {
        Path_Point_Crd=Path_2_Object->GetSplinePoint(Path_2_Object->UniformToNatural(Percent_Pos
        XZ_Dist_2_Points_Calc(current_Point_Crd,Path_Point_Crd);
        Path_to_Balance_Point_Distance=abs(Dist_current_Points);
        Path_Width=Biker_Width;
        if(Path_to_Balance_Point_Distance<=((Path_Width/2)+Sampling_Error_Dist2))
        {
          Path_Area_Point[cnt][a]=1;
        }
      }
      if(doc->FindObject("Barrier_Free_Path_Spline"))
      {
        Path_Point_Crd=Path_3_Object->GetSplinePoint(Path_3_Object->UniformToNatural(Percent_Pos
        XZ_Dist_2_Points_Calc(current_Point_Crd,Path_Point_Crd);
        Path_to_Balance_Point_Distance=abs(Dist_current_Points);
        Path_Width=WheelChair_Width;
        if( Path_to_Balance_Point_Distance<=((Path_Width/2)+Sampling_Error_Dist3))
        {
          Path_Area_Point[cnt][a]=1;
        }
      }
      Percent_Position=Percent_Position+Section_Percent_Step;
    }
  }
}
    //println("Masse-Schwerpunkt des Eigengewichts berechnen!");
    temp_Load_Balance_Point=vector(0,0,0);
    for (cnt=0;cnt<Section_PlaneNum;cnt++)
    {
      for (a=0;a<SectionNum;a++)
      {
        if(!Sum_of_Pyramides_Volumes[cnt][a]==0)
        {
          //println("Sum_Cubes_Balance_Points[cnt][a]",Sum_Cubes_Balance_Points[cnt][a]);
          temp_Load_Balance_Point=temp_Load_Balance_Point+(Sum_Cubes_Balance_Points[cnt][a]*Sin
        }
        else
        {
          temp_Load_Balance_Point=Surface_Balance_Point;
        }
      }
    }
    if(!Entire_Structure_Load==0)
    {
      temp_Load_Balance_Point=temp_Load_Balance_Point/Entire_Structure_Load;
    }
    else temp_Load_Balance_Point=Surface_Balance_Point;

    Entire_Structure_Load_Balance_Point=temp_Load_Balance_Point;

    //println("NUTZLASTSCHWERPUNKT berechnen!");
    Entire_Service_Load_Balance_Point=vector(0,0,0);
    Entire_Service_Load=0.0;

    for (cnt=0;cnt<Section_PlaneNum;cnt++)
    {
      for (a=0;a<SectionNum;a++)
      {
        if(!SinglePlanes_Load[cnt][a]==0)
        {
          Entire_Service_Load_Balance_Point=Entire_Service_Load_Balance_Point+(SinglePlane_BalancePoint[cnt][a]*Sin
          Entire_Service_Load=Entire_Service_Load+SinglePlanes_Load[cnt][a];
        }
        else
        {
          Entire_Service_Load_Balance_Point=Entire_Service_Load_Balance_Point;
          Entire_Service_Load=Entire_Service_Load+0;
        }
      }
    }
    if(!Entire_Service_Load==0)
    {
      Entire_Service_Load_Balance_Point=Entire_Service_Load_Balance_Point/Entire_Service_Load;
    }
    else Entire_Service_Load_Balance_Point=Surface_Balance_Point;
    Entire_Bridge_Load=Entire_Service_Load+Entire_Structure_Load;

    //BRÜCHEN GESAMT LASTEN BERECHNEN (Eigengewicht + Nutzlast)
   for (cnt=0;cnt<Section_PlaneNum;cnt++)
   {
     for (a=0;a<SectionNum;a++)
     {
       //SUMME der EINZEL BALANCE_PUNKTE
       Single_Bridge_Load_Points[cnt][a]=Single_Volume_Cubes_Load[cnt][a]+SinglePlanes_Load[cnt][a];
       Single_Bridge_Balance_Points[cnt][a]=(Sum_Cubes_Balance_Points[cnt][a]*Single_Volume_Cubes_Load[cnt][a]+Si
SinglePlanes_Load[cnt][a])/Single_Bridge_Load_Points[cnt][a];

//Single_Bridge_Balance_Points[cnt][a].x=((Single_Volume_Cubes_Load[cnt][a]*Sum_Cubes_Balance_Points[cnt][a].x)+(S
Point[cnt][a].x))/Single_Bridge_Load_Points[cnt][a];

//Single_Bridge_Balance_Points[cnt][a].z=((Single_Volume_Cubes_Load[cnt][a]*Sum_Cubes_Balance_Points[cnt][a].z)+(S
Point[cnt][a].z))/Single_Bridge_Load_Points[cnt][a];
       //GESAMT BALANCEPUNKT berechnen
       if(Entire_Service_Load!=0)
       {
         Entire_Bridge_Balance_Point=((Entire_Structure_Load_Balance_Point*Entire_Structure_Load)+(Entire_Service_L
Entire_Bridge_Load;
         //Entire_Bridge_Balance_Point.x=((Entire_Service_Load_Balance_Point.x*Entire_Service_Load)+(Entire_Structur
Entire_Structure_Load))/Entire_Bridge_Load;
         //Entire_Bridge_Balance_Point.z=((Entire_Service_Load_Balance_Point.z*Entire_Service_Load)+(Entire_Structur
Entire_Structure_Load))/Entire_Bridge_Load;
       }
       else Entire_Bridge_Balance_Point=Entire_Structure_Load_Balance_Point;

       //-----------------------------------------------Wenn Y-Achse nicht irgnoriert werden sollte.

//Entire_Bridge_Balance_Point[cnt][a]=((Single_Volume_Cubes_Load[cnt][a]*Sum_Cubes_Balance_Points[cnt][a])+(Singl
[cnt][a]))/2;
       //Entire_Bridge_Balance_Point=((Entire_Service_Load_Balance_Point*Entire_Service_Load)+(Entire_Structure_Lo
       //-----------------------------------------------
     }
   }
   println("Das Gesamt angenommene Brücken gewicht beträgt: ",Entire_Bridge_Load,"kN das entspricht ",Entire_Bridge_

   //-----------------------------------------------------------------
   //LASTEN ZEICHEN
   //-----------------------------------------------------------------
   //Teil 1: Eigengewicht
    var Structur_Folder_Name=("STRUCTURE_CALC"),STRUCTURE_NullObject;
    var Structure_Balance_Points_Folder_Name=("STRUCTURE_LOADS");
    var Structure_BalancePoints_Folder_Object;
    var current_Position,Size=10;

    //Support Ordner erstellen
    if(!doc->FindObject(Structure_Balance_Points_Folder_Name))
    {
      STRUCTURE_NullObject=doc->FindObject(Structur_Folder_Name);
```

```
    Structure_BalancePoints_Folder_Object=new(NullObject);
    Structure_BalancePoints_Folder_Object->SetName(Structure_Balance_Points_Folder_Name);
    doc->InsertObject(Structure_BalancePoints_Folder_Object,STRUCTURE_NullObject,Null);
  }
  //Eigengewicht_Schwerpunkt zeichen
  Structure_BalancePoints_Folder_Object=doc->FindObject(Structure_Balance_Points_Folder_Name);
  //Cubus_Eigengewicht_Schwerpunkte zeichen
  //println("Masse-Schwerpunkte der einzelnen Kubus_Volumen zeichnen!");
  var Cubes_Balance_Point_Object_Name;
  for (a=0;a<SplinePunkte-1;a++)
  {
    for (cnt=0;cnt<Anzahl_Splines-1;cnt++)
    {
      current_Position= Sum_Cubes_Balance_Points[cnt][a];
      //println("Sum_Cubes_Balance_Points[",cnt,"][",a,"]: ",Sum_Cubes_Balance_Points[cnt][a]);
      Cubes_Balance_Point_Object_Name=("|"+tostring(cnt)+"|"+tostring(a)+"|Cube_Volume_Load");
      //println("SIZE_Single_Volume_Cubes_Load[",cnt,"][",a,"]= ",Single_Volume_Cubes_Load[cnt][a]);
      //println("SIZE_Cubes_Balance_Point_Object_Name= ",Cubes_Balance_Point_Object_Name);
      Size=Single_Volume_Cubes_Load[cnt][a];
      MAKE_SPHERE(Size,Cubes_Balance_Point_Object_Name,current_Position,Structure_BalancePoints_Fo
    }
  }
  //println("Eigengewicht-Schwerpunkte zeichnen!");
  var Structure_Load_Balance_Point_Object_Name=("STRUCTURE_BALANCE_POINT");
  Size=Entire_Structure_Load/10; // durch 10 bedeutet das der Radius in Tonnen ausgegben wird
  current_Position=Entire_Structure_Load_Balance_Point;
  MAKE_SPHERE(Size,Structure_Load_Balance_Point_Object_Name,current_Position,Structure_BalancePo

  //Teil 2: Nutzlast
  var Service_Balance_Points_Folder_Name=("SERVICE_LOADS");
  var Service_BalancePoints_Folder_Object;
  var current_Position,Size=10;

 //Support Ordner erstellen
 if(!doc->FindObject(Service_Balance_Points_Folder_Name))
 {
   STRUCTURE_NullObject=doc->FindObject(Structur_Folder_Name);
   Service_BalancePoints_Folder_Object=new(NullObject);
   Service_BalancePoints_Folder_Object->SetName(Service_Balance_Points_Folder_Name);
   doc->InsertObject(Service_BalancePoints_Folder_Object,STRUCTURE_NullObject,Null);
 }
 Service_BalancePoints_Folder_Object=doc->FindObject(Service_Balance_Points_Folder_Name);
 //Cubus_Eigengewicht_Schwerpunkte zeichen
 //println("Einzlene Nutzlasten zeichnen!");
 var Balance_Point_Object_Name;
 for (a=0;a<SplinePunkte-1;a++)
 {
   for (cnt=0;cnt<Anzahl_Splines-1;cnt++)
   {
     current_Position= SinglePlane_BalancePoint[cnt][a];
     Balance_Point_Object_Name=("|"+tostring(cnt)+"|"+tostring(a)+"|Service_Load");
     Size=SinglePlanes_Load[cnt][a];
     MAKE_SPHERE(Size,Balance_Point_Object_Name,current_Position,Service_BalancePoints_Folder_Objec
   }
 }
 //Eigengewicht_Schwerpunkt zeichen
 //println("Nutzlast-Schwerpunkt zeichnen!");
 Size=Entire_Service_Load/10; // durch 10 bedeutet das der Radius in Tonnen ausgegben wird
 var Service_Load_Balance_Point_Object_Name=("ENTIRE_SERVICE_LOAD_POINT");
 current_Position=Entire_Service_Load_Balance_Point;
 MAKE_SPHERE(Size,Service_Load_Balance_Point_Object_Name,current_Position,Service_BalancePoints_
 //Teil 3: Gesamte Brückenlast
 var Service_Balance_Points_Folder_Name=("BRIDGE_LOADS");
 var Service_BalancePoints_Folder_Object;
 var current_Position,Size=10;
 //Support Ordner erstellen
 if(!doc->FindObject(Service_Balance_Points_Folder_Name))
 {
   STRUCTURE_NullObject=doc->FindObject(Structur_Folder_Name);
   Service_BalancePoints_Folder_Object=new(NullObject);
   Service_BalancePoints_Folder_Object->SetName(Service_Balance_Points_Folder_Name);
   doc->InsertObject(Service_BalancePoints_Folder_Object,STRUCTURE_NullObject,Null);
 }

 Service_BalancePoints_Folder_Object=doc->FindObject(Service_Balance_Points_Folder_Name);
 //Cubus_Eigengewicht_Schwerpunkte zeichen
 //println("Einzlene Brückenlasten zeichnen!");
 var Balance_Point_Object_Name;
 for (a=0;a<SplinePunkte-1;a++)
 {
   for (cnt=0;cnt<Anzahl_Splines-1;cnt++)
   {
     current_Position= Single_Bridge_Balance_Points[cnt][a];
     Balance_Point_Object_Name=("|"+tostring(cnt)+"|"+tostring(a)+"|Bridge_Load");
     //println("Size= ",Single_Bridge_Load_Points[cnt][a]);
     Size=Single_Bridge_Load_Points[cnt][a];
     MAKE_SPHERE(Size,Balance_Point_Object_Name,current_Position,Service_BalancePoints_Folder_Object);
   }
  }
 //Eigengewicht_Schwerpunkt zeichen
 //println("Gesamten-Brücken Schwerpunkt zeichnen!");
 Size=Entire_Bridge_Load/10; // durch 10 bedeutet das der Radius in Tonnen ausgegben wird
 var Service_Load_Balance_Point_Object_Name=("ENTIRE_BRIDGE_LOAD_POINT");
 current_Position=Entire_Bridge_Balance_Point;
 MAKE_SPHERE(Size,Service_Load_Balance_Point_Object_Name,current_Position,Service_BalancePoints_Folder_Object);
}

//Momenten berechnung - Kragarm (cantilever arm)
TORQUE_CALC()
{
 println(" ");
 println("MOMENTE AN EXTREMPUNKTEN BERECHNEN!");
 var i,c,cnt,a;
 var Section_Balance_Point=new(array,SplinePunkte);
 var current_Balance_Point;
 var SectionNum,Section_PlaneNum,Temp_Balance_Point;
 var X_Dist,Z_Dist,Temp_X,Temp_Z,Dist_RES,Temp_RES;
 var Average_Support_Position=vector(0,0,0);
 var Temp_Torque_X=new(array,SplinePunkte);
 var Temp_Torque_Z=new(array,SplinePunkte);
 var Temp_Torque_RES=new(array,SplinePunkte);
 var P1,P2;
 Section_Torque_X=new(array,SplinePunkte);
 Section_Torque_Z=new(array,SplinePunkte);
 Section_Torque_RES=new(array,SplinePunkte);
 System_Axis_Points=new(array,SplinePunkte);
 SectionNum=SplinePunkte-1;        //SchnittNr.
 Res_Section_Torque=new(array,SplinePunkte);
 //mittle AUFLAGER-POSITION berechnen
 for (cnt=0;cnt<Anzahl_Splines;cnt++)
 {
   Average_Support_Position=Average_Support_Position+BRIDGE_POINTS_ARRAY[cnt][Support_Position-1];
 }
 Average_Support_Position=Average_Support_Position/Anzahl_Splines;
 //Durchschnittsposition pro Schnitt bestimmen -> an Brücken-Geometrie (Splinepunkte) -> SystemAchse
 current_Balance_Point=vector(0,0,0);
 Temp_Balance_Point=vector(0,0,0);
 for (a=0;a<SplinePunkte;a++)
 {
   Section_Balance_Point[a]=vector(0,0,0);
   for (cnt=0;cnt<Anzahl_Splines;cnt++)
   {
     current_Balance_Point=BRIDGE_POINTS_ARRAY[cnt][a];
     Temp_Balance_Point=Temp_Balance_Point+current_Balance_Point;
   }
   Section_Balance_Point[a]=Temp_Balance_Point/(Anzahl_Splines);
   System_Axis_Points[a]=Section_Balance_Point[a]; //Die globale Variable mit den System_Achsen_Punkten wird belegt
   Temp_Balance_Point=0;
 }

 //SUMME DER SECTOR MOMENTE JE NACH AUFLAGERPOSITION
 //println(" ");
 //println("SUMME DER SECTOR MOMENTE JE NACH AUFLAGERPOSITION");
 section_Load_pos=new(array,SplinePunkte);
 Section_Load=new(array,SplinePunkte);
 var c,Distanz,P1,P2;
 var temp,temp_pos=vector(0,0,0);
 temp=0.0;
 Distanz=vector(0,0,0);
 for (a=SplinePunkte-1;a>=0;a--)
 {
   for (cnt=0;cnt<Anzahl_Splines-1;cnt++)
   {
     Section_Load[a]=0.0;
     section_Load_pos[a]=vector(0,0,0);
     Section_Torque_X[a]=0.0;//ALLE MOMENTE AUF NULL SETZEN
     Section_Torque_Z[a]=0.0;//ALLE MOMENTE AUF NULL SETZEN
     Res_Section_Torque[a]=0.0;//ALLE MOMENTE AUF NULL SETZEN
   }
 }

 var Structur_Folder_Name=("STRUCTURE_CALC");
 var STRUCTURE_NullObject,Section_Balance_Folder_Object;
 var Section_Balance_Points_Folder_Name=("Beam_"+tostring(Beam_Number_Intern)+"_SectionLoads");
 //Balance Punkte_Ordner erstellen
 if(!doc->FindObject(Section_Balance_Points_Folder_Name))
 {
   STRUCTURE_NullObject=doc->FindObject(Structur_Folder_Name);
```

```
  Section_Balance_Folder_Object=new(NullObject);
  Section_Balance_Folder_Object->SetName(Section_Balance_Points_Folder_Name);
  doc->InsertObject(Section_Balance_Folder_Object,STRUCTURE_NullObject,Null);
}
//--------------------------------verschiedene Momentenberechnungsmodi--------------------
if(Torque_Calc_Mode==1) //KRAGARME VOM AUFLAGER WEG
{
 if(Support_Position!=1)
 {
  //println("2 Kragarme von Auflager aus.");
    for (c=1;c<=Support_Position-1;c++) //Alle Punkte von 1  bis zum Auflager zählen z.B.: max 1-19
    {
     //println("SCHNITT ",c," Schleife A");
     //println("Section_Balance_Point[",c,"]=",Section_Balance_Point[c]);
     temp=0.0;
     temp_pos=vector(0,0,0);
     //Die Summe aller Resultierenden die im Schnitt betrachtet werden multipliziert mit deren Abstand zum
     for (a=c;a>=1;a--) //Vom aktuellen Schnitt bis zum Schnitt 1 herunter zählen (z.B.: Schnitt 1 x Load 0)
     {
      for (cnt=0;cnt<Anzahl_Splines-1;cnt++)   //Lasten und deren Position des aktuellen Schnittes berechne
      {
       temp=temp+Single_Bridge_Load_Points[cnt][a-1]; //Alle Lasten des temporären Schnittes mit den be
       //Gesammtschwerpontsposition berechnung -> verändert sich mit jeder addierten Last (in Richtung d
       Distanz.x=Single_Bridge_Balance_Points[cnt][a-1].x-Single_Bridge_Balance_Points[0][0].x;//der Bala
       Distanz.y=Single_Bridge_Balance_Points[cnt][a-1].y-Single_Bridge_Balance_Points[0][0].y;
       Distanz.z=Single_Bridge_Balance_Points[cnt][a-1].z-Single_Bridge_Balance_Points[0][0].z;
       temp_pos=temp_pos+Distanz*Single_Bridge_Load_Points[cnt][a-1];
      }
      Section_Load[c]=temp; //Temp erhöht sich mit jedem Durchlauf bis der aktuelle schnitt "C" erreicht ist.
      section_Load_pos[c]=Single_Bridge_Balance_Points[0][0]+temp_pos/temp; //jetzt wird die tatsächlich
     }
     //Überprüfung LASTPOSTION EINZEICHEN
     var Scale=Section_Load[c]; //Skallierung hat Auswirkung auf die SchwerpunktKugelgrößen
     var pos=section_Load_pos[c];
     var Kugel_Name;
     Kugel_Name=("BEAM_"+tostring(Beam_Number_Intern)+" SCHNITTLAST_"+tostring(c));
     MAKE_SPHERE(Scale,Kugel_Name,pos,Section_Balance_Folder_Object);
     //Momente brechenen
     //Distanz zum System_Axis_Points[a]
     X_Dist=System_Axis_Points[c].x-section_Load_pos[c].x;
     Z_Dist=System_Axis_Points[c].z-section_Load_pos[c].z;
     P1=System_Axis_Points[c];
     P2=section_Load_pos[c];
     XZ_Dist_2_Points_Calc(P1,P2);
     Dist_RES=Dist_current_Points;
     Section_Torque_X[c]=abs(X_Dist)*Section_Load[c];
     Section_Torque_Z[c]=abs(Z_Dist)*Section_Load[c];
     Res_Section_Torque[c]=abs(Dist_RES)*Section_Load[c];
     /*
     println("Section_Load[",c,"]",Section_Load[c]);
     println("section_Load_pos[",c,"]",section_Load_pos[c]);
     println("X_Dist: ",X_Dist);
     println("X-Moment an Schnitt ",c," beträgt: ",Section_Torque_X[c]);
     println("Z_Dist: ",Z_Dist);
     println("Z-Moment an Schnitt ",c," beträgt: ",Section_Torque_Z[c]);
     println("Dist_RES: ",Dist_RES);
     println("Res_Section_Torque[",c,"] beträgt: ",Res_Section_Torque[c]);
     */
    }
    for(c=SplinePunkte-2;c>=Support_Position;c--) //Alle Punkte von Oben nach Unten bis zum Auflager zäh
    {
     //println("SCHNITT ",c," Schleife B");
     temp=0.0;
     temp_pos=vector(0,0,0);
     //Die Summe aller Resultierenden die im Schnitt betrachtet werden multipliziert mit deren Abstand zum
     for (a=c-1;a<=SplinePunkte-2;a++) //Vom aktuellen Schnitt bis zum vorletzen Schnitt rechnen
     {
      for (cnt=0;cnt<Anzahl_Splines-1;cnt++)  //Lasten des aktuellen Schnittes berechnen
      {
       temp=temp+Single_Bridge_Load_Points[cnt][a];
       Distanz.x=Single_Bridge_Balance_Points[cnt][a].x-Single_Bridge_Balance_Points[0][0].x;
       Distanz.y=Single_Bridge_Balance_Points[cnt][a].y-Single_Bridge_Balance_Points[0][0].y;
       Distanz.z=Single_Bridge_Balance_Points[cnt][a].z-Single_Bridge_Balance_Points[0][0].z;
       temp_pos=temp_pos+Distanz*Single_Bridge_Load_Points[cnt][a];
      }
      Section_Load[c]=temp;
      section_Load_pos[c]=Single_Bridge_Balance_Points[0][0]+temp_pos/temp;
     }
     //Überprüfung LASTPOSTION EINZEICHEN
     var Scale=Section_Load[c]; //Skallierung hat Auswirkung auf die SchwerpunktKugelgrößen
     var pos=section_Load_pos[c];
     var Kugel_Name;
     Kugel_Name=("BEAM_"+tostring(Beam_Number_Intern)+" SCHNITTLAST_"+tostring(c));
     MAKE_SPHERE(Scale,Kugel_Name,pos,Section_Balance_Folder_Object);
     //Momente berechenen
     //Distanz zum System_Axis_Points[a]
     X_Dist=System_Axis_Points[c].x-section_Load_pos[c].x;
     Z_Dist=System_Axis_Points[c].z-section_Load_pos[c].z;
     P1=System_Axis_Points[c]; //Akuteller Schnitt
     P2=section_Load_pos[c]; //Akutelle Schnittlastposition
     XZ_Dist_2_Points_Calc(P1,P2);
     Dist_RES=abs(Dist_current_Points);
     Section_Torque_X[c]=abs(X_Dist)*Section_Load[c];
     Section_Torque_Z[c]=abs(Z_Dist)*Section_Load[c];
     Res_Section_Torque[c]=Dist_RES*Section_Load[c];
     /*
     println("Section_Load[",c,"]",Section_Load[c]);
     println("section_Load_pos[",c,"]",section_Load_pos[c]);
     println("X_Dist: ",X_Dist);
     println("X-Moment an Schnitt ",c," beträgt: ",Section_Torque_X[c]);
     println("Z_Dist: ",Z_Dist);
     println("Z-Moment an Schnitt ",c," beträgt: ",Section_Torque_Z[c]);
     println("Dist_RES: ",Dist_RES);
     println("Res_Section_Torque[",c,"] beträgt: ",Res_Section_Torque[c]);
     */
    }
  }
 } //if(Torque_Calc_Mode==1) ENDE

 if(Torque_Calc_Mode==0)  //2 Kragarme von der Uferzone weg
 {
  //println("2 Kragarme von Uferzonen -> Momentenberechnung");
  //1. ARM
  for(c=Support_Position-2;c>=0;c--) //Alle Punkte von Oben nach Unten zählen z.B.: max 19-1
  {
```

```
    //println("SCHNITT ",c," Schleife C");
    temp=0.0;
    temp_pos=vector(0,0,0);
    //Die Summe aller Resultierenden die im Schnitt betrachtet werden multipliziert mit deren Abstand zum Sc
    for (a=c;a<=Support_Position-2;a++)
    {
      for (cnt=0;cnt<Anzahl_Splines-1;cnt++)   //Lasten des aktuellen Schnittes berechnen
      {
        temp=temp+Single_Bridge_Load_Points[cnt][a];
        Distanz.x=Single_Bridge_Balance_Points[cnt][a].x-Single_Bridge_Balance_Points[0][0].x;
        Distanz.y=Single_Bridge_Balance_Points[cnt][a].y-Single_Bridge_Balance_Points[0][0].y;
        Distanz.z=Single_Bridge_Balance_Points[cnt][a].z-Single_Bridge_Balance_Points[0][0].z;
        temp_pos=temp_pos+Distanz*Single_Bridge_Load_Points[cnt][a];
        //println("Single_Bridge_Load_Points[",cnt,"][",a,"]",Single_Bridge_Load_Points[cnt][a]);
      }
      Section_Load[c]=temp;
      section_Load_pos[c]=Single_Bridge_Balance_Points[0][0]+temp_pos/temp;
    }

    //Überprüfung LASTPOSTION EINZEICHEN
    var Scale=Section_Load[c]; //Skallierung hat Auswirkung auf die SchwerpunktKugelgrößen
    var pos=section_Load_pos[c];
    var Kugel_Name;
    Kugel_Name=("BEAM_"+tostring(Beam_Number_Intern)+" SCHNITTLAST_"+tostring(c));
    MAKE_SPHERE(Scale,Kugel_Name,pos,Section_Balance_Folder_Object);
    //Momente brechenen
    //Distanz zum System_Axis_Points[a]
    X_Dist=System_Axis_Points[c].x-section_Load_pos[c].x;
    Z_Dist=System_Axis_Points[c].z-section_Load_pos[c].z;
    P1=System_Axis_Points[c];
    P2=section_Load_pos[c];
    XZ_Dist_2_Points_Calc(P1,P2);
    Dist_RES=Dist_current_Points;
    Section_Torque_X[c]=abs(X_Dist)*Section_Load[c];
    Section_Torque_Z[c]=abs(Z_Dist)*Section_Load[c];
    Res_Section_Torque[c]=abs(Dist_RES)*Section_Load[c];
    /*
    println("Section_Load[",c,"]",Section_Load[c]);
    println("section_Load_pos[",c,"]",section_Load_pos[c]);
    println("X_Dist: ",X_Dist);
    println("X-Moment an Schnitt ",c," beträgt: ",Section_Torque_X[c]);
    println("Z_Dist: ",Z_Dist);
    println("Z-Moment an Schnitt ",c," beträgt: ",Section_Torque_Z[c]);
    println("Dist_RES: ",Dist_RES);
    println("Res_Section_Torque[",c,"] beträgt: ",Res_Section_Torque[c]);
    */
  } //1. Arm Ende

  //2. ARM
  for (c=Support_Position;c<=SplinePunkte-1;c++) //invertiert -> Alle Punkte von Unten nach Oben zählen z.B.
  {
    //println("SCHNITT ",c," Schleife D");
    //println("Section_Balance_Point[",c,"]=",Section_Balance_Point[c]);
    //Die Summe aller Resultierenden die im Schnitt betrachtet werden multipliziert mit deren Abstand zum Sc
    temp=0.0;
    temp_pos=vector(0,0,0);
    for (a=c-1;a>=Support_Position-1;a--)  //Alle Punkte von C nach Oben-1 zählen z.B.: (max 0-19) max 0-18=
Schnitte.
    {
      //println("SCHNITT ",c," Schleife D");
      for (cnt=0;cnt<Anzahl_Splines-1;cnt++) //Lasten des aktuellen Schnittes berechnen
      {
        temp=temp+Single_Bridge_Load_Points[cnt][a];
        Distanz.x=Single_Bridge_Balance_Points[cnt][a].x-Single_Bridge_Balance_Points[0][0].x;
        Distanz.y=Single_Bridge_Balance_Points[cnt][a].y-Single_Bridge_Balance_Points[0][0].y;
        Distanz.z=Single_Bridge_Balance_Points[cnt][a].z-Single_Bridge_Balance_Points[0][0].z;
        temp_pos=temp_pos+Distanz*Single_Bridge_Load_Points[cnt][a];
        //println("Single_Bridge_Load_Points[",cnt,"][",a,"]",Single_Bridge_Load_Points[cnt][a]);
      }
      Section_Load[c]=temp;
      section_Load_pos[c]=Single_Bridge_Balance_Points[0][0]+temp_pos/temp;
    }
    //Überprüfung LASTPOSTION EINZEICHEN
    var Scale=Section_Load[c]; //Skallierung hat Auswirkung auf die SchwerpunktKugelgrößen
    var pos=section_Load_pos[c];
    var Kugel_Name;
    Kugel_Name=("BEAM_"+tostring(Beam_Number_Intern)+" SCHNITTLAST_"+tostring(c));
    MAKE_SPHERE(Scale,Kugel_Name,pos,Section_Balance_Folder_Object);
    //Momente brechenen
    //Distanz zum System_Axis_Points[a]
    X_Dist=System_Axis_Points[c].x-section_Load_pos[c].x;
    Z_Dist=System_Axis_Points[c].z-section_Load_pos[c].z;
    P1=System_Axis_Points[c];
    P2=section_Load_pos[c];
    XZ_Dist_2_Points_Calc(P1,P2);
    Dist_RES=Dist_current_Points;
    Section_Torque_X[c]=abs(X_Dist)*Section_Load[c];
    Section_Torque_Z[c]=abs(Z_Dist)*Section_Load[c];
    Res_Section_Torque[c]=abs(Dist_RES)*Section_Load[c];
    /*
    println("Section_Load[",c,"]",Section_Load[c]);
    println("section_Load_pos[",c,"]",section_Load_pos[c]);
    println("X_Dist: ",X_Dist);
    println("X-Moment an Schnitt ",c," beträgt: ",Section_Torque_X[c]);
    println("Z_Dist: ",Z_Dist);
    println("Z-Moment an Schnitt ",c," beträgt: ",Section_Torque_Z[c]);
    println("Dist_RES: ",Dist_RES);
    println("Res_Section_Torque[",c,"] beträgt: ",Res_Section_Torque[c]);
    */
  } //2.Arm Ende
}//  if(Torque_Calc_Mode==0) ENDE
if(Torque_Calc_Mode==2)  //Einfeldträger
{
  var Dist_ALL,X_Dist_ALL,Z_Dist_ALL;
  println("Es wird ein Einfeldträger berechnet");
  //Auflagerkräfte berechnen;
  //Auflagerkraft B=Summe aller Momente um A
  //Achtung nachstehender teil muss mit folgenden 2 Variablen kürzer programmiert werden...
  //Entire_Bridge_Load...Entire_Bridge_Balance_Point
  P1=System_Axis_Points[0];
  P2=System_Axis_Points[SplinePunkte-1];
  XZ_Dist_2_Points_Calc(P1,P2);
  Dist_ALL=Dist_current_Points;
  P1=System_Axis_Points[0];
  P2=Entire_Bridge_Balance_Point;
  XZ_Dist_2_Points_Calc(P1,P2);
  Dist_RES=Dist_current_Points;
  Support_Res_Force_B=(Entire_Bridge_Load*Dist_RES)/Dist_ALL;
  Support_Res_Force_A=Entire_Bridge_Load-Support_Res_Force_B;
  X_Dist_ALL=abs(System_Axis_Points[0].x-System_Axis_Points[SplinePunkte-1].x);
  X_Dist=abs(System_Axis_Points[0].x-Entire_Bridge_Balance_Point.x);
  Support_X_Force_B=(Entire_Bridge_Load*X_Dist)/X_Dist_ALL;
  Support_X_Force_A=Entire_Bridge_Load-Support_X_Force_B;
  Z_Dist_ALL=abs(System_Axis_Points[0].z-System_Axis_Points[SplinePunkte-1].z);
  Z_Dist=abs(System_Axis_Points[0].z-Entire_Bridge_Balance_Point.z);
  Support_Z_Force_B=(Entire_Bridge_Load*Z_Dist)/Z_Dist_ALL;
  Support_Z_Force_A=Entire_Bridge_Load-Support_Z_Force_B;
  /*
  println("Entire_Bridge_Load= ",Entire_Bridge_Load);
  println("Auflager A_x= ",Support_X_Force_A);
  println("Auflager A_z= ",Support_Z_Force_A);
  println("Auflager A_Res= ",Support_Res_Force_A);
  println("Auflager B_x= ",Support_X_Force_B);
  println("Auflager B_z= ",Support_Z_Force_B);
  println("Auflager B_Res= ",Support_Res_Force_B);
  */
  var Res_Armlenght=0.0,X_Armlenght=0.0,Z_Armlenght=0.0;
  var Res_Segment_Lenght=0.0,X_Segment_Lenght=0.0,Z_Segment_Lenght=0.0;

  //MOMENTENVERLAUFBERECHNEN unter Berücksichtigung des entgegenwirkenden Auflagermomentes berechnen.
  for(a=SplinePunkte-2;a>=0;a--) //Alle Schnitte von Oben nach Unten zählen z.B.: max 18-0
  {
    temp=0.0;
    temp_pos=vector(0,0,0);
    section_Load_pos[a]=vector(0,0,0);
    //Hebelsarmlängen berechnen
    X_Dist=System_Axis_Points[a].x-section_Load_pos[a+1].x;
    X_Segment_Lenght=X_Segment_Lenght+X_Dist;
    Z_Dist=System_Axis_Points[a].z-section_Load_pos[a+1].z;
    Z_Segment_Lenght=Z_Segment_Lenght+Z_Dist;
    P1=System_Axis_Points[a];
    P2=System_Axis_Points[a+1];
    XZ_Dist_2_Points_Calc(P1,P2);
    Res_Segment_Lenght=Dist_current_Points;
    Res_Armlenght=Res_Armlenght+Res_Segment_Lenght;
    for(c=a;c<=SplinePunkte-2;c++) //Alle Schnitte von Oben nach Unten zählen z.B.: max 18-0
    {
      //Segment Last berechnen
      for (cnt=0;cnt<Anzahl_Splines-1;cnt++) //Lasten des aktuellen Schnittes berechnen
      {
        temp=temp+Single_Bridge_Load_Points[cnt][c];
        Distanz.x=Single_Bridge_Balance_Points[cnt][c].x-Single_Bridge_Balance_Points[0][0].x;
        Distanz.y=Single_Bridge_Balance_Points[cnt][c].y-Single_Bridge_Balance_Points[0][0].y;
        Distanz.z=Single_Bridge_Balance_Points[cnt][c].z-Single_Bridge_Balance_Points[0][0].z;
        temp_pos=temp_pos+Distanz*Single_Bridge_Load_Points[cnt][c];
      }
    }
    Section_Load[a]=temp;
```

```
    section_Load_pos[a]=Single_Bridge_Balance_Points[0][0]+temp_pos/temp;
    //Überprüfung LASTPOSTION EINZEICHEN
    var Scale=Section_Load[a]; //Skallierung hat Auswirkung auf die SchwerpunktKugelgrößen
    var pos=section_Load_pos[a];
    var Kugel_Name;
    Kugel_Name=("BEAM_"+tostring(Beam_Number_Intern)+" SCHNITTLAST_"+tostring(a));
    MAKE_SPHERE(Scale,Kugel_Name,pos,Section_Balance_Folder_Object);
    X_Dist=System_Axis_Points[a].x-section_Load_pos[a].x;
    Z_Dist=System_Axis_Points[a].z-section_Load_pos[a].z;
    P1=System_Axis_Points[a];
    P2=section_Load_pos[a];
    XZ_Dist_2_Points_Calc(P1,P2);
    Dist_RES=Dist_current_Points;
    //Auflager B Abstand zu aktuellem Schnitt
    var Sup_X_Dist=0.0, Sup_Z_Dist=0.0, Sup_Res_Dist=0.0;

    Sup_X_Dist=System_Axis_Points[a].x-System_Axis_Points[SplinePunkte-1].x;
    Sup_Z_Dist=System_Axis_Points[a].z-System_Axis_Points[SplinePunkte-1].z;
    P1=System_Axis_Points[a];
    P2=System_Axis_Points[SplinePunkte-1];
    XZ_Dist_2_Points_Calc(P1,P2);
    Sup_Res_Dist=Dist_current_Points;

    Section_Torque_X[a]=abs(X_Dist*Section_Load[a])-abs(Sup_X_Dist*Support_X_Force_B);
    Section_Torque_Z[a]=abs(Z_Dist*Section_Load[a])-abs(Sup_Z_Dist*Support_Z_Force_B);
    Res_Section_Torque[a]=abs(Dist_RES*Section_Load[a])-abs(Sup_Res_Dist*Support_Res_Force_B);
    Section_Torque_X[a]=abs(Section_Torque_X[a]);
    Section_Torque_Z[a]=abs(Section_Torque_Z[a]);
    Res_Section_Torque[a]=abs(Res_Section_Torque[a]);
    Section_Torque_X[SplinePunkte-1]=0;
    Section_Torque_Z[SplinePunkte-1]=0;
    Res_Section_Torque[SplinePunkte-1]=0;
    /*
    println("Section_Load[",a,"]",Section_Load[c]);
    println("section_Load_pos[",a,"]",section_Load_pos[a]);
    println("X_Dist: ",X_Dist);
    println("X-Moment an Schnitt ",a," beträgt: ",Section_Torque_X[a]);
    println("Z_Dist: ",Z_Dist);
    println("Z-Moment an Schnitt ",a," beträgt: ",Section_Torque_Z[a]);
    println("Dist_RES: ",Dist_RES);
    println("Res_Section_Torque[",a,"] beträgt: ",Res_Section_Torque[a]);
    */
  }
 }
 //-------------------------GRAFISCHE DARSTELLUNG-------------------------
 //AUFLAGERPOSITION zeichenen
 var Structur_Folder_Name=("STRUCTURE_CALC");
 var Support_Folder_Name=("SUPPORT");
 var Support_Folder_Object,current_Support_Position,STRUCTURE_NullObject;
 var Support_Object_Name,Pyramid_Size=20;
 //Support Ordner erstellen
 if(!doc->FindObject(Support_Folder_Name))
 {
  STRUCTURE_NullObject=doc->FindObject(Structur_Folder_Name);
  Support_Folder_Object=new(NullObject);
  Support_Folder_Object->SetName(Support_Folder_Name);
  doc->InsertObject(Support_Folder_Object,STRUCTURE_NullObject,Null);
 }
 Support_Folder_Object=doc->FindObject(Support_Folder_Name);
 for (cnt=0;cnt<Anzahl_Splines;cnt++)
 {
  current_Support_Position=BRIDGE_POINTS_ARRAY[cnt][Support_Position-1];
  Support_Object_Name=("Support"+tostring(cnt));
  MAKE_SPHERE(Pyramid_Size,Support_Object_Name, current_Support_Position,Support_Folder_Object);
 }
 //Zeichne Durchnschitt Auflager
 var Average_Support_Name=(" Average_Support");
 STRUCTURE_NullObject=doc->FindObject(Structur_Folder_Name);
 MAKE_PYRAMID(Pyramid_Size*5,Average_Support_Name,Average_Support_Position, STRUCTURE_NullObj
 //MOMENTE zeichen
 var Size=vector(0,0,0);
 var Torque_Object_Name;
 var Torque_Folder_Name=("Beam_"+tostring(Beam_Number_Intern)+"_Torque_Folder");
 //var Torque_Folder_Name=("Torque_Folder");
 var Torque_Folder_Object;
 var Torque_Scale_Factor=Diagram_Factor;
 var Result_Torque_Folder_Object_Name=("Beam_"+tostring(Beam_Number_Intern)+"_Result_Torque");
 //var Result_Torque_Folder_Object_Name=("Result_Torque");
 var Result_Torque_Folder_Object;
 if(!doc->FindObject(Torque_Folder_Name))
 {
  STRUCTURE_NullObject=doc->FindObject(Structur_Folder_Name);
  Torque_Folder_Object=new(NullObject);
  Torque_Folder_Object->SetName(Torque_Folder_Name);
  doc->InsertObject(Torque_Folder_Object,STRUCTURE_NullObject,Null);
 }
 if(!doc->FindObject(Result_Torque_Folder_Object_Name))
 {
  Torque_Folder_Object=doc->FindObject(Torque_Folder_Name);
  Result_Torque_Folder_Object=new(NullObject);
  Result_Torque_Folder_Object->SetName(Result_Torque_Folder_Object_Name);
  doc->InsertObject(Result_Torque_Folder_Object,Torque_Folder_Object,Null);
 }
 Result_Torque_Folder_Object=doc->FindObject(Result_Torque_Folder_Object_Name);
 Torque_Folder_Object=doc->FindObject(Torque_Folder_Name);
 //Momente zeichnen

 //Aufgeteilt in 2 Schleifen, damit Objektordnerstruktur beider Kraagarme geich bleibt.
 for (a=SectionNum;a>=Support_Position-1;a--)
 {
  Torque_Object_Name=("Beam_"+tostring(Beam_Number_Intern)+"_X_Torque_Section_"+tostring(a));
  Size=vector(abs(Section_Torque_X[a])/Torque_Scale_Factor,5,5);//Würfelabbmessungen
  Section_Balance_Point[a].x=Section_Balance_Point[a].x+(Section_Torque_X[a]/Torque_Scale_Factor)/2;//Würfelpositi
  MAKE_CUBE(Size,Torque_Object_Name,Section_Balance_Point[a], Torque_Folder_Object);
  Torque_Object_Name=("Beam_"+tostring(Beam_Number_Intern)+"_Z_Torque_Section_"+tostring(a));
  Size=vector(5,5,abs(Section_Torque_Z[a])/Torque_Scale_Factor); //Würfelabbmessungen
  Section_Balance_Point[a].x=Section_Balance_Point[a].x-(Section_Torque_X[a]/Torque_Scale_Factor)/2;//Würfelpositi
  Section_Balance_Point[a].z=Section_Balance_Point[a].z+(Section_Torque_Z[a]/Torque_Scale_Factor)/2;//Würfelpositi
  MAKE_CUBE(Size,Torque_Object_Name,Section_Balance_Point[a], Torque_Folder_Object);
  Torque_Object_Name=("Beam_"+tostring(Beam_Number_Intern)+"_Res_Torque_Section_"+tostring(a));
  Size=vector(5,Res_Section_Torque[a]/Torque_Scale_Factor,5); //Würfelabbmessungen
  Section_Balance_Point[a].y=Section_Balance_Point[a].y+(Res_Section_Torque[a]/Torque_Scale_Factor)/2;//Würfelpos
  Section_Balance_Point[a].z=Section_Balance_Point[a].z-(Section_Torque_Z[a]/Torque_Scale_Factor)/2;
  MAKE_CUBE(Size,Torque_Object_Name,Section_Balance_Point[a], Result_Torque_Folder_Object);
  Section_Balance_Point[a].z=Section_Balance_Point[a].z+(Section_Torque_Z[a]/Torque_Scale_Factor)/2;
 }
 if(Support_Position!=1)
 {
  for (a=Support_Position-2;a>=0;a--)//          for (a=0;a<Support_Position-1;a++)
  {
   Torque_Object_Name=("X_Torque_Section_"+tostring(a));
   Size=vector(abs(Section_Torque_X[a])/Torque_Scale_Factor,5,5);//Würfelabbmessungen
   Section_Balance_Point[a].x=Section_Balance_Point[a].x+(Section_Torque_X[a]/Torque_Scale_Factor)/2;//Würfelposi
   MAKE_CUBE(Size,Torque_Object_Name,Section_Balance_Point[a], Torque_Folder_Object);
   Torque_Object_Name=("Z_Torque_Section_"+tostring(a));
   Size=vector(5,5,abs(Section_Torque_Z[a])/Torque_Scale_Factor); //Würfelabbmessungen
   Section_Balance_Point[a].x=Section_Balance_Point[a].x-(Section_Torque_X[a]/Torque_Scale_Factor)/2;//Würfelposi
   Section_Balance_Point[a].z=Section_Balance_Point[a].z+(Section_Torque_Z[a]/Torque_Scale_Factor)/2;//Würfelposi
   MAKE_CUBE(Size,Torque_Object_Name,Section_Balance_Point[a], Torque_Folder_Object);
   Torque_Object_Name=("Res_Torque_Section_"+tostring(a));
   Size=vector(5,Res_Section_Torque[a]/Torque_Scale_Factor,5); //Würfelabbmessungen
   Section_Balance_Point[a].y=Section_Balance_Point[a].y+(Res_Section_Torque[a]/Torque_Scale_Factor)/2;//Würfelpo
   Section_Balance_Point[a].z=Section_Balance_Point[a].z-(Section_Torque_Z[a]/Torque_Scale_Factor)/2;
   MAKE_CUBE(Size,Torque_Object_Name,Section_Balance_Point[a], Result_Torque_Folder_Object);
   Section_Balance_Point[a].z=Section_Balance_Point[a].z+(Section_Torque_Z[a]/Torque_Scale_Factor)/2;
  }
 }
}

//Träger berechnen und zeichnen
BEAM_CALC(Beam_Number_Intern)
{
 println(" ");
 if(Beam_Number_Intern<=1) println("IDEALTRÄGER ZEICHEN");
 if(Beam_Number_Intern>1) println(Beam_Number_Intern,"_TRÄGER ZEICHEN");
 var c,cnt,a;
 var First_Spline,Last_Spline;
 var Ideal_Beam_Width=new(array,SplinePunkte);
 var Temp_Height;
 var current_Position_1,current_Position_2;
 var temp_factor,Factor_1=1.0;
 var Widerstandsmoment;
 //Breite an den einzelnen Netzpunken (Matrixpunkten) berechenen;
 for (a=0;a<SplinePunkte;a++)
 {
  First_Spline=BRIDGE_POINTS_ARRAY[0][a];
  Last_Spline=BRIDGE_POINTS_ARRAY[Anzahl_Splines-1][a];
  XZ_Dist_2_Points_Calc(First_Spline,Last_Spline);
  Ideal_Beam_Width[a]=Dist_current_Points;
  //println("Trägerbreite an Stelle ",a," beträgt: ",Ideal_Beam_Width[a]);
 }
 current_Position_1=System_Axis_Points[0];
 current_Position_2=System_Axis_Points[Support_Position-1];
 XZ_Dist_2_Points_Calc(current_Position_1, current_Position_2);
 Cantilever_Arm_1=abs(Dist_current_Points);
 current_Position_1=System_Axis_Points[SplinePunkte-1];
 XZ_Dist_2_Points_Calc(current_Position_1, current_Position_2);
 Cantilever_Arm_2=abs(Dist_current_Points);
```

```
if(Torque_Calc_Mode==2) //Einfeldträger
{
  current_Position_1=System_Axis_Points[0];
  current_Position_2=System_Axis_Points[SplinePunkte-1];
  XZ_Dist_2_Points_Calc(current_Position_1, current_Position_2);
  Cantilever_Arm_1=abs(Dist_current_Points);
  Cantilever_Arm_2=0.0;
}
if(Torque_Curve_Calc==True)
{
  //---------------------------------------------------------------------
  //---------------------------------------------------------------------
  //--------------Die Maximale_Höhe den Momenten angleichen--------------------
  //---------------------------------------------------------------------
  println("Torque_Curve_Calc=true");
  println("Formula_Divident_Arm_1: ",Formula_Divident_Arm_1);
  println("Formula_Divident_Arm_2: ",Formula_Divident_Arm_2);
  println("Cantilever_Arm_1: ", Cantilever_Arm_1);
  println("Cantilever_Arm_2: ", Cantilever_Arm_2);
  //Section_Torque_X,Section_Torque_Z,Res_Section_Torque;
  var pos; //korrekturwert wenn auflager an stelle null. -> keine referentmoment vorhanden
  if(Torque_Calc_Mode==1) //KRAGARME VOM AUFLAGER WEG
  {
    if(Support_Position!=1)
    {
      for (a=0;a<=Support_Position-1;a++) //Alle Punkte von 1  bis zum Auflager zählen z.B.: max 1-19
      {
        for (cnt=0;cnt<Anzahl_Splines;cnt++)
        {
          Temp_Height=(2*Cantilever_Arm_1)/Formula_Divident_Arm_2;
          if(Res_Section_Torque[a]!=0) //Wenn keine Momente vorhanden sind dann gibt es keine Lasten und au
          {
            Ideal_Beam_Heights[cnt][a]=((Temp_Height/Res_Section_Torque[Support_Position-1])*Res_Section_
          }
          else Ideal_Beam_Heights[cnt][a]=Minimal_Construction_Height;//oder Null
        }
      }
      for(a=SplinePunkte-1;a>=Support_Position;a--) //Alle Punkte von Oben nach Unten bis zum Auflager zäh
      {
        for (cnt=0;cnt<Anzahl_Splines;cnt++)
        {
          Temp_Height=(2*Cantilever_Arm_2)/Formula_Divident_Arm_1;
          if(Res_Section_Torque[a]!=0) //Wenn keine Momente vorhanden sind dann gibt es keine Lasten und au
          {
            Ideal_Beam_Heights[cnt][a]=((Temp_Height/Res_Section_Torque[Support_Position])*Res_Section_T
          }
          else Ideal_Beam_Heights[cnt][a]=Minimal_Construction_Height;//oder Null
        }
      }
    } //if(Support_Position!=1) ENDE

    if(Support_Position==1) //Wenn Auflagerposition = 1 ist dann normaler Rechenmodus
    {
      for (a=SplinePunkte-1;a>=Support_Position-1;a--) //Alle Punkte von 1  bis zum Auflager zählen z.B.: max
      {
        for (cnt=0;cnt<Anzahl_Splines;cnt++)
        {
          Temp_Height=(2*Cantilever_Arm_2)/Formula_Divident_Arm_2;
          if(Res_Section_Torque[a]!=0) //Wenn keine Momente vorhanden sind dann gibt es keine Lasten und au
          {
            Ideal_Beam_Heights[cnt][a]=((Temp_Height/Res_Section_Torque[0])*Res_Section_Torque[a])+Minim
          }
          else Ideal_Beam_Heights[cnt][a]=Minimal_Construction_Height;//oder Null
        }
      }
    }
  }//if(Torque_Calc_Mode==1) ENDE

  if(Torque_Calc_Mode==0)  //2 Kragarme von der Uferzone weg
  {
    for(a=Support_Position-1;a>=0;a--) //Alle Punkte von Oben nach Unten zählen z.B.: max 19-1
    {
      for (cnt=0;cnt<Anzahl_Splines;cnt++)
      {
        Temp_Height=(2*Cantilever_Arm_1)/Formula_Divident_Arm_2;
        if(Res_Section_Torque[a]!=0) //Wenn keine Momente vorhanden sind dann gibt es keine Lasten und auc
        {
          Ideal_Beam_Heights[cnt][a]=((Temp_Height/Res_Section_Torque[0])*Res_Section_Torque[a])+Minima
        }
        else Ideal_Beam_Heights[cnt][a]=Minimal_Construction_Height;//oder Null
      }
    } //1. Arm Ende
    for (a=Support_Position;a<=SplinePunkte-1;a++) //invertiert -> Alle Punkte von Unten nach Oben zählen z.
    {
      for (cnt=0;cnt<Anzahl_Splines;cnt++)
      {
        Temp_Height=(2*Cantilever_Arm_2)/Formula_Divident_Arm_2;
        if(Res_Section_Torque[a]!=0) //Wenn keine Momente vorhanden sind dann gibt es keine Lasten und auch keine Dicke (Höhe)
        {
          Ideal_Beam_Heights[cnt][a]=((Temp_Height/Res_Section_Torque[SplinePunkte-1])*Res_Section_Torque[a])+Minimal_Construc
        }
        else Ideal_Beam_Heights[cnt][a]=Minimal_Construction_Height;//oder Null
      }
    } //2.Arm Ende
  }//if(Torque_Calc_Mode==0) ENDE

  if(Torque_Calc_Mode==2)  //2 Kragarme von der Uferzone weg
  {
    //Welches Moment ist am größten
    var Res_Section_Max_Torque=0.0;
    for(a=0;a<=SplinePunkte-1;a++) //Alle Punkte von Oben nach Unten zählen z.B.: max 19-1
    {
      if(Res_Section_Torque[a]>=Res_Section_Max_Torque) Res_Section_Max_Torque=Res_Section_Torque[a];
    }
    for(a=SplinePunkte-1;a>=0;a--) //Alle Punkte von Oben nach Unten zählen z.B.: max 19-1
    {
      for (cnt=0;cnt<Anzahl_Splines;cnt++)
      {
        Temp_Height=(Cantilever_Arm_1)/(Formula_Divident_Arm_1/0.6);
        if(Res_Section_Max_Torque!=0) //Wenn keine Momente vorhanden sind dann gibt es keine Lasten und auch keine Dicke (Höhe)
        {
          Ideal_Beam_Heights[cnt][a]=((Temp_Height/Res_Section_Max_Torque)*Res_Section_Torque[a])+Minimal_Construction_Height
        }
        else Ideal_Beam_Heights[cnt][a]=Minimal_Construction_Height;//oder Null
      }
    } //1. Arm Ende
  }//    if(Torque_Calc_Mode==2) ENDE

  if(Faktor_1_Enable==True)
  {
    //---------------------------------------------------------------------
    //----FAKTOR 1 für Höhen/Breiten Verhältnis einrechnen; Ideal=H/B=2.5----
    //---------------------------------------------------------------------
    var temp_faktor;
    for (a=0;a<SplinePunkte;a++)
    {
      for (cnt=0;cnt<Anzahl_Splines;cnt++)
      {
        temp_faktor=1/Beam_Proportion_Faktor; //z.b= 0.4; Da Beam_Proportion_Faktor=2.5;
        Widerstandsmoment=((Ideal_Beam_Heights[cnt][a]*temp_faktor)*Ideal_Beam_Heights[cnt][a]*Ideal_Beam_Heights[cnt][a])/6; //f
Breite angenommen (0.4*Höhe -> Idealträger) und ein ideales Widerstandsmoment berechnet
        temp_factor=sqrt((6*Widerstandsmoment)/Ideal_Beam_Width[a]); //Nun wird die Formel für das Widerstandmoment (Rechteckq
Höhe zu errechnen. Die Sollhöhe wird jetzt errechnet indem die Gesamtbreite eingesetzt wird.
        Ideal_Beam_Heights[cnt][a]=temp_factor+Minimal_Construction_Height; //Sollhöhe ist schließlich
      }
      //println(a," temp_factor= ",temp_factor);
      //println(a," Factor_1: ",Factor_1);
    }
  }//if(Beam_Proportion_Faktor!=0) ENDE

  if(Faktor_2_Enable==True)
  {
    //---------------------------------------------------------------------
    //----FAKTOR 2 für Höhen/Breiten Verhältnis PLUS Formfaktor Höhe 1 zu Breite 0.4 = 100%
    //---------------------------------------------------------------------
    var Averidge_High;
    var Faktor_2;
    for (a=0;a<SplinePunkte;a++)
    {
      Averidge_High=0.0;
      for (cnt=0;cnt<Anzahl_Splines;cnt++)
      {
        Averidge_High=Averidge_High+Ideal_Beam_Heights[cnt][a]; //Sollhöhe ist schließlich
      }
      Averidge_High=Averidge_High/Anzahl_Splines;
      Faktor_2=(Ideal_Beam_Width[a]/Averidge_High); //der sollte 0.4 oder kleiner sein.
      Faktor_2=Faktor_2_Proportion_Value/Faktor_2;
      for (cnt=0;cnt<Anzahl_Splines;cnt++)
      {
        //Wenn Träger höher dann günstiger
        Ideal_Beam_Heights[cnt][a]=Faktor_2*Ideal_Beam_Heights[cnt][a];
        if(Ideal_Beam_Heights[cnt][a]<=Minimal_Construction_Height) Ideal_Beam_Heights[cnt][a]=Minimal_Construction_Height;
      }
    }
  } // if(Faktor_2_Enable==True) ENDE
  if(Section_Shape_Mode!=0)
  {
    MAKE_PARABLE_KONTUR(Ideal_Beam_Heights); //Man bekommt am ende die Veränderten "Ideal_Beam_Heights[cnt][a]" zurück
```

```
  }
  //Volumshöhenpunkte zuweisen
  var Profil_Punkte=2*Anzahl_Splines;
  for (a=0;a<SplinePunkte;a++)
  {
    for (cnt=0;cnt<Anzahl_Splines;cnt++)
    {
      BRIDGE_VOLUME_POINTS_ARRAY[cnt][a]=BRIDGE_POINTS_ARRAY[cnt][a];
      BRIDGE_VOLUME_POINTS_ARRAY[(Profil_Punkte-1)-cnt][a]=BRIDGE_POINTS_ARRAY[cnt][a];
      BRIDGE_VOLUME_POINTS_ARRAY[(Profil_Punkte-1)-cnt][a].y=BRIDGE_VOLUME_POINTS_ARRAY[(Pro
      //println("HÖHEN->Ideal_Beam_Heights[",cnt,"][",a,"]:",Ideal_Beam_Heights[cnt][a]);
    }
  }
} //if(Torque_Curve_Calc==True) ENDE
//-------------------------------------------------------------------------------------------
//-------------------------------------------------------------------------------------------------------------------
//TRÄGER ZEICHNEN
var Structur_Folder_Name=("STRUCTURE_CALC"),STRUCTURE_NullObject;
var Beam_Folder_Name=(tostring(Beam_Number_Intern)+"_BEAM");
var Beam_Folder_Object, Beam_Folder_Object_Name=(tostring(Beam_Number_Intern)+"BEAM_Folder_SPA
//Haupt-Ordner erstellen //z.B.: 3_BEAM ->Beam_Folder_Name
if(!doc->FindObject(Beam_Folder_Name))
{
  STRUCTURE_NullObject=doc->FindObject(Structur_Folder_Name);
  Beam_Folder_Object=new(NullObject);
  Beam_Folder_Object->SetName(Beam_Folder_Name);
  doc->InsertObject(Beam_Folder_Object,STRUCTURE_NullObject,Null);
}
//Positiv_Volumen-Ordner erstellen //z.B.: 3_POSITIV_VOLUME
var Positiv_Volume_Object;
var Positiv_Volume_Object_Name=(tostring(Beam_Number_Intern)+"_POSITIV_VOLUME");
if(!doc->FindObject(Positiv_Volume_Object_Name))
{
  Beam_Folder_Object=doc->FindObject(Beam_Folder_Name);
  Positiv_Volume_Object=new(LoftObject);
  Positiv_Volume_Object->SetName(Positiv_Volume_Object_Name);
  doc->InsertObject(Positiv_Volume_Object,Beam_Folder_Object,Null);
}

//AUßENHÜLLE ZEICHEN
var Beam_Kontur_Name;
Positiv_Volume_Object=doc->FindObject(Positiv_Volume_Object_Name);
for (a=0;a<SplinePunkte;a++)
{
  Beam_Kontur_Name=(tostring(Beam_Number_Intern)+"BEAMKONTUR"+tostring(a));
  MAKE_3RD_BEAM_PROFILE(a,Ideal_Beam_Heights,Beam_Kontur_Name,Positiv_Volume_Object);
}
//NegativVolumen-Ordner erstellen
var Negativ_Volume_Object,Negativ_Volume_Object_Name=(tostring(Beam_Number_Intern)+"NEGATIV_VOL
if(!doc->FindObject(Negativ_Volume_Object_Name))
{
  Beam_Folder_Object=doc->FindObject(Beam_Folder_Name);
  Negativ_Volume_Object=new(NullObject);
  Negativ_Volume_Object->SetName(Negativ_Volume_Object_Name);
  doc->InsertObject(Negativ_Volume_Object,Beam_Folder_Object,Null);
}
//Profiltyp definieren
var TYP_Front,TYP_Back;
var S0P0,S0P0_H,S1P0_H,S1P0; //4 Eckpunkte
var X_DEG,Y_DEG,Z_DEG;
var Dist_S0P0S1P0,Dist_S1P0S1P0_H,Dist_S1P0_HS0P0_H,Dist_S0P0_HS0P0;
var Temp_Point_1=vector(0,0,0),Temp_Point_2=vector(0,0,0),Temp_Point_3=vector(0,0,0),Temp_Point_4=v
var Left_M_Point,Right_M_Point,Profil_Name;
var Temp_Distance,Temp_Point=vector(0,0,0);
for (cnt=0;cnt<Anzahl_Splines-1;cnt++)
{
  //LOFTOBJEKTORDNER ERSTELLEN;
  var Negativ_Tube_Object_1,Negativ_Tube_Object_1_Name=("Beam_"+tostring(Beam_Number_Intern)+"_"+t
  if(!doc->FindObject(Negativ_Tube_Object_1_Name))
  {
    Negativ_Volume_Object=doc->FindObject(Negativ_Volume_Object_Name);
    Negativ_Tube_Object_1=new(LoftObject);
    Negativ_Tube_Object_1->SetName(Negativ_Tube_Object_1_Name);
    doc->InsertObject(Negativ_Tube_Object_1,Negativ_Volume_Object,Null);
  }
  var Negativ_Tube_Object_2,Negativ_Tube_Object_2_Name=("Beam_"+tostring(Beam_Number_Intern)+"_"+
  if(!doc->FindObject(Negativ_Tube_Object_2_Name))
  {
    Negativ_Volume_Object=doc->FindObject(Negativ_Volume_Object_Name);
    Negativ_Tube_Object_2=new(LoftObject);
    Negativ_Tube_Object_2->SetName(Negativ_Tube_Object_2_Name);
    doc->InsertObject(Negativ_Tube_Object_2,Negativ_Volume_Object,Null);
  }
  Negativ_Volume_Object=doc->FindObject(Negativ_Volume_Object_Name);
  for (a=0;a<SplinePunkte;a++)
  {
    TYP_Front=PROFILETYPE[cnt][a];
    //println("[",cnt,"][",a,"] TYP_Front: ",TYP_Front);
    if(TYP_Front==1) //Vollprofil
    {
      OG_Height=0.49999;
      UG_Height=0.49999;
      Left_Width=0.49999;
      Right_Width=0.49999;
    }
    if(TYP_Front==2) //C-Profil
    {
      OG_Height=OG_Height_Percent;
      UG_Height=UG_Height_Percent;
      Left_Width=Left_Width_Percent;
      Right_Width=0.0;
     }
    if(TYP_Front==3) //Nur Ober und Untergurt-Profil
    {
      OG_Height=OG_Height_Percent;
      UG_Height=UG_Height_Percent;
      Left_Width=0.0;
      Right_Width=0.0;
    }
    if(TYP_Front==4) //verkehrtes C-Profil
    {
      OG_Height=OG_Height_Percent;
      UG_Height=UG_Height_Percent;
      Left_Width=0.0;
      Right_Width=Right_Width_Percent;
    }
    if(TYP_Front==5) //Rechteckiges Hohlprofil
    {
       OG_Height=OG_Height_Percent;
       UG_Height=UG_Height_Percent;
       Left_Width=Left_Width_Percent/1.5;
       Right_Width=Right_Width_Percent/1.5;
    }
    S0P0=BRIDGE_VOLUME_POINTS_ARRAY[cnt][a];
    S1P0=BRIDGE_VOLUME_POINTS_ARRAY[cnt+1][a];
    S0P0_H=BRIDGE_VOLUME_POINTS_ARRAY[(Anzahl_Splines*2-1)-cnt][a];
    S1P0_H=BRIDGE_VOLUME_POINTS_ARRAY[(Anzahl_Splines*2-1)-cnt-1][a];
    XYZ_Dist_2_Points_Calc(S0P0,S1P0);
    Dist_S0P0S1P0=Dist_current_Points;
    XYZ_Dist_2_Points_Calc(S1P0,S1P0_H);
    Dist_S1P0S1P0_H=Dist_current_Points;
    XYZ_Dist_2_Points_Calc(S1P0_H,S0P0_H);
    Dist_S1P0_HS0P0_H=Dist_current_Points;
    XYZ_Dist_2_Points_Calc(S0P0_H,S0P0);
    Dist_S0P0_HS0P0=Dist_current_Points;
    //PunktLinksOben
    X_DEG=S0P0.x-S1P0.x;
    Y_DEG=S0P0.y-S1P0.y;
    Z_DEG=S0P0.z-S1P0.z;
    Temp_Point_1=S0P0;
    Temp_Point_1.x=S0P0.x-((X_DEG/Dist_S0P0S1P0)*(Dist_S0P0S1P0*Left_Width));
    Temp_Point_1.y=S0P0.y-((Y_DEG/Dist_S0P0S1P0)*(Dist_S0P0S1P0*Left_Width));
    Temp_Point_1.z=S0P0.z-((Z_DEG/Dist_S0P0S1P0)*(Dist_S0P0S1P0*Left_Width));
    Temp_Point=Temp_Point_1;
    X_DEG=S0P0.x-S0P0_H.x;
    Y_DEG=S0P0.y-S0P0_H.y;
    Z_DEG=S0P0.z-S0P0_H.z;
    Temp_Point_1.x=Temp_Point_1.x-((X_DEG/Dist_S0P0_HS0P0)*(Dist_S0P0_HS0P0*OG_Height));
    Temp_Point_1.y=Temp_Point_1.y-((Y_DEG/Dist_S0P0_HS0P0)*(Dist_S0P0_HS0P0*OG_Height));
    Temp_Point_1.z=Temp_Point_1.z-((Z_DEG/Dist_S0P0_HS0P0)*(Dist_S0P0_HS0P0*OG_Height));
    XYZ_Dist_2_Points_Calc(Temp_Point,Temp_Point_1);
    Temp_Distance=abs(Dist_current_Points);
    if(Temp_Distance<=Obergurt_Mindesthoehe)
    {
      Temp_Point_1.y=Temp_Point.y-Obergurt_Mindesthoehe;
    }
    //PunktRechtsOben
    X_DEG=S1P0.x-S0P0.x;
    Y_DEG=S1P0.y-S0P0.y;
    Z_DEG=S1P0.z-S0P0.z;
    Temp_Point_2=S1P0;
    Temp_Point_2.x=S1P0.x-((X_DEG/Dist_S0P0S1P0)*(Dist_S0P0S1P0*Right_Width));
    Temp_Point_2.y=S1P0.y-((Y_DEG/Dist_S0P0S1P0)*(Dist_S0P0S1P0*Right_Width));
    Temp_Point_2.z=S1P0.z-((Z_DEG/Dist_S0P0S1P0)*(Dist_S0P0S1P0*Right_Width));

    Temp_Point=Temp_Point_2;
    X_DEG=S1P0.x-S1P0_H.x;
    Y_DEG=S1P0.y-S1P0_H.y;
```

```
Z_DEG=S1P0.z-S1P0_H.z;
Temp_Point_2.x=Temp_Point_2.x-((X_DEG/Dist_S1P0S1P0_H)*(Dist_S1P0S1P0_H*OG_Height));
Temp_Point_2.y=Temp_Point_2.y-((Y_DEG/Dist_S1P0S1P0_H)*(Dist_S1P0S1P0_H*OG_Height));
Temp_Point_2.z=Temp_Point_2.z-((Z_DEG/Dist_S1P0S1P0_H)*(Dist_S1P0S1P0_H*OG_Height));
XYZ_Dist_2_Points_Calc(Temp_Point,Temp_Point_2);
Temp_Distance=abs(Dist_current_Points);
if(Temp_Distance<=Obergurt_Mindesthoehe)
{
  Temp_Point_2.y=Temp_Point.y-Obergurt_Mindesthoehe;
}
//PunktRechtsUnten
X_DEG=S1P0_H.x-S0P0_H.x;
Y_DEG=S1P0_H.y-S0P0_H.y;
Z_DEG=S1P0_H.z-S0P0_H.z;
Temp_Point_3=S1P0_H;
Temp_Point_3.x=S1P0_H.x-((X_DEG/Dist_S1P0_HS0P0_H)*(Dist_S1P0_HS0P0_H*Right_Width));
Temp_Point_3.y=S1P0_H.y-((Y_DEG/Dist_S1P0_HS0P0_H)*(Dist_S1P0_HS0P0_H*Right_Width));
Temp_Point_3.z=S1P0_H.z-((Z_DEG/Dist_S1P0_HS0P0_H)*(Dist_S1P0_HS0P0_H*Right_Width));
Temp_Point=Temp_Point_3;

X_DEG=S1P0_H.x-S1P0.x;
Y_DEG=S1P0_H.y-S1P0.y;
Z_DEG=S1P0_H.z-S1P0.z;
Temp_Point_3.x=Temp_Point_3.x-((X_DEG/Dist_S1P0S1P0_H)*(Dist_S1P0S1P0_H*UG_Height));
Temp_Point_3.y=Temp_Point_3.y-((Y_DEG/Dist_S1P0S1P0_H)*(Dist_S1P0S1P0_H*UG_Height));
Temp_Point_3.z=Temp_Point_3.z-((Z_DEG/Dist_S1P0S1P0_H)*(Dist_S1P0S1P0_H*UG_Height));
XYZ_Dist_2_Points_Calc(Temp_Point,Temp_Point_3);
Temp_Distance=abs(Dist_current_Points);
if(Temp_Distance<=Untergurt_Mindesthoehe)
{
  Temp_Point_3.y=Temp_Point.y+Untergurt_Mindesthoehe;
}
//PunktLinksUnten
X_DEG=S0P0_H.x-S1P0_H.x;
Y_DEG=S0P0_H.y-S1P0_H.y;
Z_DEG=S0P0_H.z-S1P0_H.z;
Temp_Point_4=S0P0_H;
Temp_Point_4.x=S0P0_H.x-((X_DEG/Dist_S1P0_HS0P0_H)*(Dist_S1P0_HS0P0_H*Left_Width));
Temp_Point_4.y=S0P0_H.y-((Y_DEG/Dist_S1P0_HS0P0_H)*(Dist_S1P0_HS0P0_H*Left_Width));
Temp_Point_4.z=S0P0_H.z-((Z_DEG/Dist_S1P0_HS0P0_H)*(Dist_S1P0_HS0P0_H*Left_Width));
Temp_Point=Temp_Point_4;
X_DEG=S0P0_H.x-S0P0.x;
Y_DEG=S0P0_H.y-S0P0.y;
Z_DEG=S0P0_H.z-S0P0.z;
Temp_Point_4.x=Temp_Point_4.x-((X_DEG/Dist_S0P0_HS0P0)*(Dist_S0P0_HS0P0*UG_Height));
Temp_Point_4.y=Temp_Point_4.y-((Y_DEG/Dist_S0P0_HS0P0)*(Dist_S0P0_HS0P0*UG_Height));
Temp_Point_4.z=Temp_Point_4.z-((Z_DEG/Dist_S0P0_HS0P0)*(Dist_S0P0_HS0P0*UG_Height));
XYZ_Dist_2_Points_Calc(Temp_Point,Temp_Point_4);
Temp_Distance=abs(Dist_current_Points);
if(Temp_Distance<=Untergurt_Mindesthoehe)
{
  Temp_Point_4.y=Temp_Point.y+Untergurt_Mindesthoehe;
}
//Punkte zuweisen
S0P0=Temp_Point_1;
S1P0=Temp_Point_2;
S1P0_H=Temp_Point_3;
S0P0_H=Temp_Point_4;
if(abs(Temp_Point_1.y-Temp_Point_4.y)<=(Minimal_Construction_Height))
{
  S0P0=(Temp_Point_1+Temp_Point_4)/2;
  S0P0_H=(Temp_Point_1+Temp_Point_4)/2;
  S0P0_H.y=S0P0_H.y-0.0001;
}
if(abs(Temp_Point_2.y-Temp_Point_3.y)<=(Minimal_Construction_Height))
{
  S1P0=(Temp_Point_2+Temp_Point_3)/2;
  S1P0_H=(Temp_Point_2+Temp_Point_3)/2;
  S1P0_H.y=S1P0_H.y-0.0001;
}
//2 RECHTECKE für Innenvolumen zeichnen
//Oberes Rechteck
Left_M_Point=(S0P0+S0P0_H)/2;
Right_M_Point=(S1P0+S1P0_H)/2;
/*
println("|",cnt,"|",a,"|S0P0:",S0P0);
println("|",cnt,"|",a,"|S1P0:",S1P0);
println("|",cnt,"|",a,"|Right_M_Poin:",Right_M_Point);
println("|",cnt,"|",a,"|Left_M_Point:",Left_M_Point);
println("|",cnt,"|",a,"|S1P0_H:",S1P0_H);
println("|",cnt,"|",a,"|S0P0_H:",S0P0_H);
*/
if(TYP_Front==1)
{
  S0P0=Left_M_Point;
  S0P0.y=S0P0.y-0.0001;
  S1P0=Left_M_Point;
  S1P0.y=S1P0.y+0.0001;
  S1P0_H=Left_M_Point;
  S1P0_H.y=S1P0_H.y-0.0001;
  S0P0_H=Left_M_Point;
  S0P0_H.y=S0P0_H.y+0.0001;
 }

 Profil_Name=("Beam_"+tostring(Beam_Number_Intern)+"_"+"|"+(tostring(cnt))+"|"+(tostring(a))+"| UPPER_NEGATIV_RECT");
 Negativ_Tube_Object_1=doc->FindObject(Negativ_Tube_Object_1_Name);
 MAKE_4_POINT_RECTANGLE(S0P0,S1P0,Right_M_Point,Left_M_Point,Profil_Name,Negativ_Tube_Object_1);
 //Unteres Rechteckzeichnen
 Profil_Name=("Beam_"+tostring(Beam_Number_Intern)+"_"+"|"+(tostring(cnt))+"|"+(tostring(a))+"| LOWER_NEGATIV_RECT");
 Negativ_Tube_Object_2=doc->FindObject(Negativ_Tube_Object_2_Name);
 MAKE_4_POINT_RECTANGLE(Left_M_Point,Right_M_Point,S1P0_H,S0P0_H,Profil_Name,Negativ_Tube_Object_2);
 }
}
}

//Dreiecksfläche berechnen
Triangle_Area(A_Point,B_Point,C_Point)
{
 /*
 println(" ");
 println("Dreiecksfläche berechnen");
 println(" ");
 */
 var AC_Diagonale_Lenght,AB_Lenght,BC_Lenght;
 var Half_Outline;
 var Plane_Triangle1;
 XYZ_Dist_2_Points_Calc(A_Point,C_Point);
 AC_Diagonale_Lenght=Dist_current_Points;
 XYZ_Dist_2_Points_Calc(A_Point,B_Point);
 AB_Lenght=Dist_current_Points;
 XYZ_Dist_2_Points_Calc(B_Point,C_Point);
 BC_Lenght=Dist_current_Points;
 /*
 println("AC_Diagonale_Lenght: ",AC_Diagonale_Lenght);
 println("AB_Diagonale_Lenght: ",AB_Lenght);
 println("BC_Diagonale_Lenght: ",BC_Lenght);
 println("CD_Diagonale_Lenght: ",CD_Lenght);
 println("DA_Diagonale_Lenght: ",DA_Lenght);
 */
 //Dreieck 1
 Half_Outline=(AB_Lenght+BC_Lenght+AC_Diagonale_Lenght)/2;
 Plane_Triangle1=sqrt(Half_Outline*(Half_Outline-AB_Lenght)*(Half_Outline-BC_Lenght)*(Half_Outline-AC_Diagonale_Lenght));
 Triangle_Area_Value=Plane_Triangle1;
 Triangle_Area_Balance_Point=vector(0,0,0);
 Triangle_Area_Balance_Point=(A_Point+B_Point+C_Point)/3;
 return Triangle_Area_Value,Triangle_Area_Balance_Point;
}

//Rechtecksfläche berechnen
Rectangle_Area(A_Point,B_Point,C_Point,D_Point)
{
 /*
 println(" ");
 println("Rechtecksfläche berechnen");
 println(" ");
 */
 var AC_Diagonale_Lenght,AB_Lenght,BC_Lenght,CD_Lenght,DA_Lenght;
 var Half_Outline;
 var Plane_Triangle1,Plane_Triangle2;
 XYZ_Dist_2_Points_Calc(A_Point,B_Point);
 AB_Lenght=Dist_current_Points;
 XYZ_Dist_2_Points_Calc(B_Point,C_Point);
 BC_Lenght=Dist_current_Points;
 XYZ_Dist_2_Points_Calc(C_Point,D_Point);
 CD_Lenght=Dist_current_Points;
 XYZ_Dist_2_Points_Calc(D_Point,A_Point);
 DA_Lenght=Dist_current_Points;
 XYZ_Dist_2_Points_Calc(A_Point,C_Point);
 AC_Diagonale_Lenght=Dist_current_Points;
 /*
 println("AC_Diagonale_Lenght: ",AC_Diagonale_Lenght);
 println("AB_Diagonale_Lenght: ",AB_Lenght);
 println("BC_Diagonale_Lenght: ",BC_Lenght);
 println("CD_Diagonale_Lenght: ",CD_Lenght);
 println("DA_Diagonale_Lenght: ",DA_Lenght);
 */
 //Dreieck 1
 Half_Outline=(AB_Lenght+BC_Lenght+AC_Diagonale_Lenght)/2;
```

```
  Plane_Triangle1=sqrt(Half_Outline*(Half_Outline-AB_Lenght)*(Half_Outline-BC_Lenght)*(Half_Outline-AC_Di
  //Dreieck 2
  Half_Outline=(AC_Diagonale_Lenght+CD_Lenght+DA_Lenght)/2;
  Plane_Triangle2=sqrt(Half_Outline*(Half_Outline-AC_Diagonale_Lenght)*(Half_Outline-CD_Lenght)*(Half_Ou
  Pyramind_Base_Area=abs(Plane_Triangle1)+abs(Plane_Triangle2);//Eigentlich kann das Ergebnis nur Positiv
  Rectangle_Area_Value=Pyramind_Base_Area;
  Rectangle_Area_Balance_Point=vector(0,0,0);
  Rectangle_Area_Balance_Point=(A_Point+B_Point+C_Point+D_Point)/4;
  return Pyramind_Base_Area,Rectangle_Area_Value,Rectangle_Area_Balance_Point;
}

//Berechnung der Einzel_Volumen und des Gesamten für
CUBE_VOLUMES_BALCANCE_POINTS_CALC(P1,P2,P3,P4,P5,P6,P7,P8,cnt,a)
{
  var i;
  var Part_1_Volume,Part_2_Volume,Spat_Volume,Cube_Volume;
  var Vector_SA,Vector_SB,Vector_SC,Vector_SD;
  var Cube_Volume_Balance_Point_Groundplane=vector(0,0,0);
  var Point_A,Point_B,Point_C,Point_D;
  var Temp_Balance_Point,Temp_Height, Balance_Point_1=vector(0,0,0),Balance_Point_2=vector(0,0,0),Pyram
  var X_Degree,Y_Degree,Z_Degree;
  var Sum_Pyramid_Balance_Point=new(array,6);
  var Pyramid_Volume=new(array,6);
  Cube_Volume_Balance_Point=(P1+P2+P3+P4+P5+P6+P7+P8)/8;
  //6 Pyramidenvolumen berechnen
  //P1,P2,P3,P4,P5,P6,P7,P8 = S0P0,S1P0,S0P1,S1P1,S0P0_H,S1P0_H,S0P1_H,S1P1_H
  //1. Pyramide berechnen (P1,P2,P6,P5,Cube_Volume_Balance_Point)
  //2. Pyramide berechnen (P3,P7,P8,P4,Cube_Volume_Balance_Point)
  //3. Pyramide berechnen (P5,P6,P8,P7,Cube_Volume_Balance_Point)
  //4. Pyramide berechnen (P1,P3,P4,P2,Cube_Volume_Balance_Point)
  //5. Pyramide berechnen (P1,P5,P7,P3,Cube_Volume_Balance_Point)
  //6. Pyramide berechnen (P2,P4,P8,P6,Cube_Volume_Balance_Point)
  //-----------------------------------------------------------------
  Single_Cube_Volume=0;
  for(i=1;i<=6;i++)
  {
    //println(" ");
    //println(i," Pyramide I",cnt,"I",a,"I ");
    //println("BERECHNE PYRAMIDE ",i);
    if(i==1)Point_A=P1,Point_B=P2,Point_C=P6,Point_D=P5;//Vorderseite als Pyramidengrundfläche //Funktioni
vorhanden...(Rechterwinkel von Schwerpunkt auf Line AB und AD.)
    if(i==2)Point_A=P4,Point_B=P3,Point_C=P7,Point_D=P8;//Rückseiteseite als Pyramidengrundfläche
    if(i==3)Point_A=P7,Point_B=P5,Point_C=P6,Point_D=P8;//Unterseite als Pyramidengrundfläche//Funktionie
weil mit rechtwinkligem anstatt eines schiefwinkelingen dreicks gerchnent wird.
    if(i==4)Point_A=P1,Point_B=P3,Point_C=P4,Point_D=P2;//Oberseite als Pyramidengrundfläche//Funktionier
    if(i==5)Point_A=P3,Point_B=P1,Point_C=P5,Point_D=P7;//Linkeseite als Pyramidengrundfläche//
    if(i==6)Point_A=P2,Point_B=P6,Point_C=P8,Point_D=P4;//Rechteseite als Pyramidengrundfläche//
    /*
    println("Point_A: ",Point_A);
    println("Point_B: ",Point_B);
    println("Point_C: ",Point_C);
    println("Point_D: ",Point_D);
    println("Spitze: ",Cube_Volume_Balance_Point);
    */
    //Diese Pyramide wird weitier in 2 Pyramiden Unterteilt
    //1. Teilpyramide
    //Es werden die Vektoren berechnet die Pyramide Beschreiben. -> Richtung und Betrag -> Alle 3 Vektoren n
    Vector_SA=Point_A-Cube_Volume_Balance_Point;
    Vector_SB=Point_B-Cube_Volume_Balance_Point;
    Vector_SC=Point_C-Cube_Volume_Balance_Point;
    //println("Vector_SA: ", Vector_SA);
    //println("Vector_SB: ", Vector_SB);
    //println("Vector_SC: ", Vector_SC);
    //Spat_Volume= Kreuzprodukt aus 2 Verkoren * dem 3. Vektor und das ganze durch 6
    //ALT
//Spat_Volume=Vector_SC.x*Vector_SA.y*Vector_SB.z-Vector_SC.x*Vector_SA.z*Vector_SB.y+Vector_SC.y*
or_SB.z+Vector_SC.z*Vector_SA.x*Vector_SB.y-Vector_SC.z*Vector_SA.y*Vector_SB.x;
    var KreuzproduktSASB=vector(0,0,0);
    var Kreuzprodukt_X,Kreuzprodukt_Y,Kreuzprodukt_Z;
    Kreuzprodukt_X=Vector_SA.y*Vector_SB.z-Vector_SA.z*Vector_SB.y;
    Kreuzprodukt_Y=Vector_SA.z*Vector_SB.x-Vector_SA.x*Vector_SB.z;
    Kreuzprodukt_Z=Vector_SA.x*Vector_SB.y-Vector_SA.y*Vector_SB.x;
    KreuzproduktSASB.x=Kreuzprodukt_X;
    KreuzproduktSASB.y=Kreuzprodukt_Y;
    KreuzproduktSASB.z=Kreuzprodukt_Z;
    //Scalarprodukt mit 3.Punkt; durch 6
    Spat_Volume=abs((KreuzproduktSASB.x*Vector_SC.x+KreuzproduktSASB.y*Vector_SC.y+KreuzproduktSAS
    Part_1_Volume=Spat_Volume;
    //ALT Part_1_Volume=abs(Spat_Volume/6);
    //println(" 1. Teilvolumen: ", Part_1_Volume);
    //1. Teilschwerpunkt berechnen.
    Temp_Balance_Point=(Point_A+Point_B+Point_C)/3;  //Summe der Grundflächenpunkte/deren Anzahl
    XYZ_Dist_2_Points_Calc(Temp_Balance_Point,Cube_Volume_Balance_Point);
    Temp_Height=Dist_current_Points;
    X_Degree=(Temp_Balance_Point.x-Cube_Volume_Balance_Point.x)/Temp_Height;
    Y_Degree=(Temp_Balance_Point.y-Cube_Volume_Balance_Point.y)/Temp_Height;
    Z_Degree=(Temp_Balance_Point.z-Cube_Volume_Balance_Point.z)/Temp_Height;
    Balance_Point_1.x=Temp_Balance_Point.x+X_Degree*(Temp_Height/3);
    Balance_Point_1.y=Temp_Balance_Point.y+Y_Degree*(Temp_Height/3);
    Balance_Point_1.z=Temp_Balance_Point.z+Z_Degree*(Temp_Height/3);
    //2. Teilpyramide
    Vector_SA=Point_A-Cube_Volume_Balance_Point;
    Vector_SC=Point_C-Cube_Volume_Balance_Point;
    Vector_SD=Point_D-Cube_Volume_Balance_Point;
    //println("Vector_SA: ", Vector_SA);
    //println("Vector_SC: ", Vector_SC);
    //println("Vector_SD: ", Vector_SD);
    //ALTSpat_Volume=Vector_SC.x*Vector_SA.y*Vector_SB.z-Vector_SC.x*Vector_SA.z*Vector_SB.y+Vector_SC.y*Vec
    //*Vector_SA.x*Vector_SB.z+Vector_SC.z*Vector_SA.x*Vector_SB.y-Vector_SC.z*Vector_SA.y*Vector_SB.x;
    //ALTPart_2_Volume=abs(Spat_Volume/6);
    //println(" 2. Teilvolumen: ", Part_2_Volume);
    var KreuzproduktSASC=vector(0,0,0);
    var Kreuzprodukt_X,Kreuzprodukt_Y;Kreuzprodukt_Z;
    Kreuzprodukt_X=Vector_SA.y*Vector_SC.z-Vector_SA.z*Vector_SC.y;
    Kreuzprodukt_Y=Vector_SA.z*Vector_SC.x-Vector_SA.x*Vector_SC.z;
    Kreuzprodukt_Z=Vector_SA.x*Vector_SC.y-Vector_SA.y*Vector_SC.x;
    KreuzproduktSASC.x=Kreuzprodukt_X;
    KreuzproduktSASC.y=Kreuzprodukt_Y;
    KreuzproduktSASC.z=Kreuzprodukt_Z;
    //Scalarprodukt mit 3.Punkt; durch 6
    Spat_Volume=abs((KreuzproduktSASC.x*Vector_SD.x+KreuzproduktSASC.y*Vector_SD.y+KreuzproduktSASC.z*Vecto
    Part_2_Volume=Spat_Volume;
    //2. Teilschwerpunkt berechnen.
    Temp_Balance_Point=(Point_A+Point_C+Point_D)/3;  //Summe der Grundflächenpunkte/deren Anzahl
    XYZ_Dist_2_Points_Calc(Temp_Balance_Point,Cube_Volume_Balance_Point);
    Temp_Height=Dist_current_Points;
    X_Degree=(Temp_Balance_Point.x-Cube_Volume_Balance_Point.x)/Temp_Height;
    Y_Degree=(Temp_Balance_Point.y-Cube_Volume_Balance_Point.y)/Temp_Height;
    Z_Degree=(Temp_Balance_Point.z-Cube_Volume_Balance_Point.z)/Temp_Height;
    Balance_Point_2.x=Temp_Balance_Point.x+X_Degree*(Temp_Height/3);
    Balance_Point_2.y=Temp_Balance_Point.y+Y_Degree*(Temp_Height/3);
    Balance_Point_2.z=Temp_Balance_Point.z+Z_Degree*(Temp_Height/3);
    Pyramid_Volume[i-1]=Part_1_Volume+Part_2_Volume;
    //println(i," Pyramide_Volumen: ", Pyramid_Volume[i-1]);
    if(Pyramid_Volume[i-1]!=0)
    {
      Pyramid_Balance_Point=(Part_1_Volume*Balance_Point_1+Part_2_Volume*Balance_Point_2)/Pyramid_Volume[i-1];
    }
    else
    {
      Pyramid_Balance_Point==(P1+P2+P3+P4+P5+P6+P7+P8)/8;
    }
    Sum_Pyramid_Balance_Point[i-1]=Pyramid_Balance_Point;
    Single_Cube_Volume= Single_Cube_Volume+Pyramid_Volume[i-1];
  }
  //println("in unterprogramm Single_Cubes_Volumen: ",  Single_Cube_Volume);
  if(Single_Cube_Volume!=0)
  {
    Cube_Volume_Balance_Point=((Sum_Pyramid_Balance_Point[0]*Pyramid_Volume[0])+(Sum_Pyramid_Balance_Point
Sum_Pyramid_Balance_Point[2]
    *Pyramid_Volume[2])+(Sum_Pyramid_Balance_Point[3]*Pyramid_Volume[3])+(Sum_Pyramid_Balance_Point[4]*Pyrar
[5]
    *Pyramid_Volume[5]))/Single_Cube_Volume;
    //println("Cube_Volume_Balance_Point= ",Cube_Volume_Balance_Point);
  }
  else Cube_Volume_Balance_Point=(P1+P2+P3+P4+P5+P6+P7+P8)/8;
  return Single_Cube_Volume,Cube_Volume_Balance_Point;
}

//Berechnung der Einzel_Volumen und des Gesamten für
BRIDGE_VOLUME_CALC()
{
  println(" ");
  println("BRÜCKEN VOLUMSBERECHNUNG");

  var c,cnt,a;
  var Positiv_Cubes_Volume, Positiv_Entire_Volume,Positiv_Cubes_Balance_Point;
  var Negativ_Cubes_Volume, Negativ_Entire_Volume,Negativ_Cubes_Balance_Point;
  var Positiv_Cubes_Balance_Point,Negativ_Cubes_Balance_Point;
  var S0P0,S0P0_H,S1P0_H,S1P0,S0P1,S0P1_H,S1P1_H,S1P1; //8 Volumspunkte
  var TYP_Front,TYP_Back;
  var Temp_Point_1,Temp_Point_2,Temp_Point_3,Temp_Point_4;
  var X_DEG,Y_DEG,Z_DEG;
  var Temp_Point,Temp_Distance;
  var Profil_Point_Num=2*Anzahl_Splines;
  var DownPointCNT;
  Positiv_Cubes_Volume=new(array,Anzahl_Splines-1,SplinePunkte-1);
  Negativ_Cubes_Volume=new(array,Anzahl_Splines-1,SplinePunkte-1);
```

```
Positiv_Cubes_Balance_Point=new(array,Anzahl_Splines-1,SplinePunkte-1);
Negativ_Cubes_Balance_Point=new(array,Anzahl_Splines-1,SplinePunkte-1);
Sum_Cubes_Balance_Points=new(array,Anzahl_Splines-1,SplinePunkte-1);
Sum_of_Pyramides_Volumes=new(array,Anzahl_Splines-1,SplinePunkte-1);
//Es sollten die einzelschwerpunkte und der Gesamtschwerpunkt (Eigengewicht) berechet werden
//so wie auch Einzel- und Gesamtvolumen
Positiv_Entire_Volume=0.0;
Negativ_Entire_Volume=0.0;
Entire_Structure_Volume=0.0;
//Alle Cubus bildenden Punkte und Profile abtasten
for (a=0;a<SplinePunkte-1;a++)
{
  for (cnt=0;cnt<Anzahl_Splines-1;cnt++)
  {
    //Außenvolumen (Positivvolumen) eines Elements berechnen
    //Volume Pointsabragen...dazu muss man den aktuellenstand abfragen.
    S0P0= BRIDGE_VOLUME_POINTS_ARRAY[cnt][a];
    S1P0=BRIDGE_VOLUME_POINTS_ARRAY[cnt+1][a];
    S0P1=BRIDGE_VOLUME_POINTS_ARRAY[cnt][a+1];
    S1P1= BRIDGE_VOLUME_POINTS_ARRAY[cnt+1][a+1];
    DownPointCNT=(Profil_Point_Num-1)-cnt;
    S0P0_H=BRIDGE_VOLUME_POINTS_ARRAY[DownPointCNT][a];
    S1P0_H=BRIDGE_VOLUME_POINTS_ARRAY[DownPointCNT-1][a];
    S0P1_H=BRIDGE_VOLUME_POINTS_ARRAY[DownPointCNT][a+1];
    S1P1_H=BRIDGE_VOLUME_POINTS_ARRAY[DownPointCNT-1][a+1];
    CUBE_VOLUMES_BALCANCE_POINTS_CALC(S0P0,S1P0,S0P1,S1P1,S0P0_H,S1P0_H,S0P1_H,S1P1_H,c
    Positiv_Cubes_Balance_Point[cnt][a]=Cube_Volume_Balance_Point;
    Positiv_Cubes_Volume[cnt][a]=Single_Cube_Volume;
    //println("Hauptprogramm: Single_Cube_Volumen: ",Single_Cube_Volume);
    Positiv_Entire_Volume=Positiv_Entire_Volume+ Positiv_Cubes_Volume[cnt][a];

    //Innenvolumen (Negativvolumesn) eines Elements berechnen
    //Berechnungstyp übergeben (Typ_1=0;Typ_1=1;Typ_2=2;Typ_3=3;)

    TYP_Front=PROFILETYPE[cnt][a];
    TYP_Back =PROFILETYPE[cnt][a+1];
    //println("[",cnt,"][",a,"] TYP_Front: ",TYP_Front);
    //println("[",cnt,"][",a+1,"] TYP_Back: ",TYP_Back);
    if(TYP_Front==1) //Voll-Profil
    {
      OG_Height=0.5;
      UG_Height=0.5;
      Left_Width=0.5;
      Right_Width=0.5;
    }
    if(TYP_Front==2) //C-Profil
    {
      OG_Height=OG_Height_Percent;
      UG_Height=UG_Height_Percent;
      Left_Width=Left_Width_Percent;
      Right_Width=0.0;
    }
    if(TYP_Front==3) //Nur Ober und Untergurt-Profil
    {
      OG_Height=OG_Height_Percent;
      UG_Height=UG_Height_Percent;
      Left_Width=0.0;
      Right_Width=0.0;
    }
    if(TYP_Front==4) //verkehrtes C-Profil
    {
      OG_Height=OG_Height_Percent;
      UG_Height=UG_Height_Percent;
      Left_Width=0.0;
      Right_Width=Right_Width_Percent;
    }
    if(TYP_Front==5) //Rechteckiges Hohlprofil
    {
      OG_Height=OG_Height_Percent;
      UG_Height=UG_Height_Percent;
      Left_Width=Left_Width_Percent/1.5;
      Right_Width=Right_Width_Percent/1.5;
    }
    var Dist_S0P0S1P0,Dist_S1P0S1P0_H,Dist_S1P0_HS0P0_H,Dist_S0P0_HS0P0;
    S0P0=BRIDGE_VOLUME_POINTS_ARRAY[cnt][a];
    S1P0=BRIDGE_VOLUME_POINTS_ARRAY[cnt+1][a];
    S0P0_H=BRIDGE_VOLUME_POINTS_ARRAY[(Profil_Point_Num-1)-cnt][a];
    S1P0_H=BRIDGE_VOLUME_POINTS_ARRAY[(Profil_Point_Num-1)-cnt-1][a];
    XYZ_Dist_2_Points_Calc(S0P0,S1P0);
    Dist_S0P0S1P0=Dist_current_Points;
    XYZ_Dist_2_Points_Calc(S1P0,S1P0_H);
    Dist_S1P0S1P0_H=Dist_current_Points;
    XYZ_Dist_2_Points_Calc(S1P0_H,S0P0_H);
    Dist_S1P0_HS0P0_H=Dist_current_Points;
    XYZ_Dist_2_Points_Calc(S0P0_H,S0P0);
    Dist_S0P0_HS0P0=Dist_current_Points;
    //PunktLinksOben
    X_DEG=S0P0.x-S1P0.x;
    Y_DEG=S0P0.y-S1P0.y;
    Z_DEG=S0P0.z-S1P0.z;
    Temp_Point_1=S0P0;
    Temp_Point_1.x=S0P0.x-((X_DEG/Dist_S0P0S1P0)*(Dist_S0P0S1P0*Left_Width));
    Temp_Point_1.y=S0P0.y-((Y_DEG/Dist_S0P0S1P0)*(Dist_S0P0S1P0*Left_Width));
    Temp_Point_1.z=S0P0.z-((Z_DEG/Dist_S0P0S1P0)*(Dist_S0P0S1P0*Left_Width));
    Temp_Point=Temp_Point_1;
    X_DEG=S0P0.x-S0P0_H.x;
    Y_DEG=S0P0.y-S0P0_H.y;
    Z_DEG=S0P0.z-S0P0_H.z;

    Temp_Point_1.x=Temp_Point_1.x-((X_DEG/Dist_S0P0_HS0P0)*(Dist_S0P0_HS0P0*OG_Height));
    Temp_Point_1.y=Temp_Point_1.y-((Y_DEG/Dist_S0P0_HS0P0)*(Dist_S0P0_HS0P0*OG_Height));
    Temp_Point_1.z=Temp_Point_1.z-((Z_DEG/Dist_S0P0_HS0P0)*(Dist_S0P0_HS0P0*OG_Height));
    XYZ_Dist_2_Points_Calc(Temp_Point,Temp_Point_1);
    Temp_Distance=abs(Dist_current_Points);
    if(Temp_Distance<=Obergurt_Mindesthoehe)
    {
      Temp_Point_1.y=Temp_Point.y-Obergurt_Mindesthoehe;
    }
    //PunktRechtsOben
    X_DEG=S1P0.x-S0P0.x;
    Y_DEG=S1P0.y-S0P0.y;
    Z_DEG=S1P0.z-S0P0.z;
    Temp_Point_2=S1P0;
    Temp_Point_2.x=S1P0.x-((X_DEG/Dist_S0P0S1P0)*(Dist_S0P0S1P0*Right_Width));
    Temp_Point_2.y=S1P0.y-((Y_DEG/Dist_S0P0S1P0)*(Dist_S0P0S1P0*Right_Width));
    Temp_Point_2.z=S1P0.z-((Z_DEG/Dist_S0P0S1P0)*(Dist_S0P0S1P0*Right_Width));
    Temp_Point=Temp_Point_2;
    X_DEG=S1P0.x-S1P0_H.x;
    Y_DEG=S1P0.y-S1P0_H.y;
    Z_DEG=S1P0.z-S1P0_H.z;

    Temp_Point_2.x=Temp_Point_2.x-((X_DEG/Dist_S1P0S1P0_H)*(Dist_S1P0S1P0_H*OG_Height));
    Temp_Point_2.y=Temp_Point_2.y-((Y_DEG/Dist_S1P0S1P0_H)*(Dist_S1P0S1P0_H*OG_Height));
    Temp_Point_2.z=Temp_Point_2.z-((Z_DEG/Dist_S1P0S1P0_H)*(Dist_S1P0S1P0_H*OG_Height));
    XYZ_Dist_2_Points_Calc(Temp_Point,Temp_Point_2);
    Temp_Distance=abs(Dist_current_Points);
    if(Temp_Distance<=Obergurt_Mindesthoehe)
    {
      Temp_Point_2.y=Temp_Point.y-Obergurt_Mindesthoehe;
    }
    //PunktRechtsUnten
    X_DEG=S1P0_H.x-S0P0_H.x;
    Y_DEG=S1P0_H.y-S0P0_H.y;
    Z_DEG=S1P0_H.z-S0P0_H.z;

    Temp_Point_3=S1P0_H;
    Temp_Point_3.x=S1P0_H.x-((X_DEG/Dist_S1P0_HS0P0_H)*(Dist_S1P0_HS0P0_H*Right_Width));
    Temp_Point_3.y=S1P0_H.y-((Y_DEG/Dist_S1P0_HS0P0_H)*(Dist_S1P0_HS0P0_H*Right_Width));
    Temp_Point_3.z=S1P0_H.z-((Z_DEG/Dist_S1P0_HS0P0_H)*(Dist_S1P0_HS0P0_H*Right_Width));
    Temp_Point=Temp_Point_3;
    X_DEG=S1P0_H.x-S1P0.x;
    Y_DEG=S1P0_H.y-S1P0.y;
    Z_DEG=S1P0_H.z-S1P0.z;
    Temp_Point_3.x=Temp_Point_3.x-((X_DEG/Dist_S1P0S1P0_H)*(Dist_S1P0S1P0_H*UG_Height));
    Temp_Point_3.y=Temp_Point_3.y-((Y_DEG/Dist_S1P0S1P0_H)*(Dist_S1P0S1P0_H*UG_Height));
    Temp_Point_3.z=Temp_Point_3.z-((Z_DEG/Dist_S1P0S1P0_H)*(Dist_S1P0S1P0_H*UG_Height));
    XYZ_Dist_2_Points_Calc(Temp_Point,Temp_Point_3);
    Temp_Distance=abs(Dist_current_Points);
    if(Temp_Distance<=Untergurt_Mindesthoehe)
    {
      Temp_Point_3.y=Temp_Point.y+Untergurt_Mindesthoehe;
    }
    //PunktLinksUnten
    X_DEG=S0P0_H.x-S1P0_H.x;
    Y_DEG=S0P0_H.y-S1P0_H.y;
    Z_DEG=S0P0_H.z-S1P0_H.z;

    Temp_Point_4=S0P0_H;
    Temp_Point_4.x=S0P0_H.x-((X_DEG/Dist_S1P0_HS0P0_H)*(Dist_S1P0_HS0P0_H*Left_Width));
    Temp_Point_4.y=S0P0_H.y-((Y_DEG/Dist_S1P0_HS0P0_H)*(Dist_S1P0_HS0P0_H*Left_Width));
    Temp_Point_4.z=S0P0_H.z-((Z_DEG/Dist_S1P0_HS0P0_H)*(Dist_S1P0_HS0P0_H*Left_Width));
    Temp_Point=Temp_Point_4;
    X_DEG=S0P0_H.x-S0P0.x;
    Y_DEG=S0P0_H.y-S0P0.y;
    Z_DEG=S0P0_H.z-S0P0.z;
    Temp_Point_4.x=Temp_Point_4.x-((X_DEG/Dist_S0P0_HS0P0)*(Dist_S0P0_HS0P0*UG_Height));
    Temp_Point_4.y=Temp_Point_4.y-((Y_DEG/Dist_S0P0_HS0P0)*(Dist_S0P0_HS0P0*UG_Height));
    Temp_Point_4.z=Temp_Point_4.z-((Z_DEG/Dist_S0P0_HS0P0)*(Dist_S0P0_HS0P0*UG_Height));
```

```
XYZ_Dist_2_Points_Calc(Temp_Point,Temp_Point_4);
Temp_Distance=abs(Dist_current_Points);
if(Temp_Distance<=Untergurt_Mindesthoehe)
{
  Temp_Point_4.y=Temp_Point.y+Untergurt_Mindesthoehe;
}
//Punkte zuweisen
S0P0=Temp_Point_1;
S1P0=Temp_Point_2;
S1P0_H=Temp_Point_3;
S0P0_H=Temp_Point_4;
if(abs(Temp_Point_1.y-Temp_Point_4.y)<=(Minimal_Construction_Height))
{
  S0P0=(Temp_Point_1+Temp_Point_4)/2;
  S0P0_H=(Temp_Point_1+Temp_Point_4)/2;
}
if(abs(Temp_Point_2.y-Temp_Point_3.y)<=(Minimal_Construction_Height))
{
  S1P0=(Temp_Point_2+Temp_Point_3)/2;
  S1P0_H=(Temp_Point_2+Temp_Point_3)/2;
}
if(TYP_Back==1) //Vollprofil
{
  OG_Height=0.5;
  UG_Height=0.5;
  Left_Width=0.5;
  Right_Width=0.5;
}
if(TYP_Back==2) //C-Profil
{
  OG_Height=OG_Height_Percent;
  UG_Height=UG_Height_Percent;
  Left_Width=Left_Width_Percent;
  Right_Width=0.0;
}
if(TYP_Back==3) //Nur Ober und Untergurt-Profil
{
  OG_Height=OG_Height_Percent;
  UG_Height=UG_Height_Percent;
  Left_Width=0.0;
  Right_Width=0.0;
}
if(TYP_Back==4) //verkehrtes C-Profil
{
  OG_Height=OG_Height_Percent;
  UG_Height=UG_Height_Percent;
  Left_Width=0.0;
  Right_Width=Right_Width_Percent;
}
if(TYP_Back==5) //Rechteckiges Hohlprofil
{
  OG_Height=OG_Height_Percent;
  UG_Height=UG_Height_Percent;
  Left_Width=Left_Width_Percent/1.5;
  Right_Width=Right_Width_Percent/1.5;
}
S0P1=BRIDGE_VOLUME_POINTS_ARRAY[cnt][a+1];
S1P1=BRIDGE_VOLUME_POINTS_ARRAY[cnt+1][a+1];
S0P1_H=BRIDGE_VOLUME_POINTS_ARRAY[(Profil_Point_Num-1)-cnt][a+1];
S1P1_H=BRIDGE_VOLUME_POINTS_ARRAY[(Profil_Point_Num-1)-cnt-1][a+1];
XYZ_Dist_2_Points_Calc(S0P1,S1P1);
Dist_S0P0S1P0=Dist_current_Points;
XYZ_Dist_2_Points_Calc(S1P1,S1P1_H);
Dist_S1P0S1P0_H=Dist_current_Points;
XYZ_Dist_2_Points_Calc(S1P1_H,S0P1_H);
Dist_S1P0_HS0P0_H=Dist_current_Points;
XYZ_Dist_2_Points_Calc(S0P1_H,S0P1);
Dist_S0P0_HS0P0=Dist_current_Points;

//PunktLinksOben - back
X_DEG=S0P1.x-S1P1.x;
Y_DEG=S0P1.y-S1P1.y;
Z_DEG=S0P1.z-S1P1.z;
/*
println("-----------------------------");
println("Spline: ",cnt," Punkt: ",a);
println("S0P0: ",S0P0);
println("S1P0: ",S1P0);
println("Dist_S0P0S1P0: ",Dist_S0P0S1P0);
println("Dist_S1P0S1P0_H: ",Dist_S1P0S1P0_H);
println("Dist_S1P0_HS0P0_H: ",Dist_S1P0_HS0P0_H);
println("Dist_S0P0_HS0P0: ",Dist_S0P0_HS0P0);
*/
Temp_Point_1=S0P1;
Temp_Point_1.x=S0P1.x-((X_DEG/Dist_S0P0S1P0)*(Dist_S0P0S1P0*Left_Width));
Temp_Point_1.y=S0P1.y-((Y_DEG/Dist_S0P0S1P0)*(Dist_S0P0S1P0*Left_Width));
Temp_Point_1.z=S0P1.z-((Z_DEG/Dist_S0P0S1P0)*(Dist_S0P0S1P0*Left_Width));

Temp_Point=Temp_Point_1;
X_DEG=S0P1.x-S0P1_H.x;
Y_DEG=S0P1.y-S0P1_H.y;
Z_DEG=S0P1.z-S0P1_H.z;

Temp_Point_1.x=Temp_Point_1.x-((X_DEG/Dist_S0P0_HS0P0)*(Dist_S0P0_HS0P0*OG_Height));
Temp_Point_1.y=Temp_Point_1.y-((Y_DEG/Dist_S0P0_HS0P0)*(Dist_S0P0_HS0P0*OG_Height));
Temp_Point_1.z=Temp_Point_1.z-((Z_DEG/Dist_S0P0_HS0P0)*(Dist_S0P0_HS0P0*OG_Height));
XYZ_Dist_2_Points_Calc(Temp_Point,Temp_Point_1);
Temp_Distance=abs(Dist_current_Points);
if(Temp_Distance<=Obergurt_Mindesthoehe)
{
  Temp_Point_1.y=Temp_Point.y-Obergurt_Mindesthoehe;
}
//PunktRechtsOben - back
X_DEG=S1P1.x-S0P1.x;
Y_DEG=S1P1.y-S0P1.y;
Z_DEG=S1P1.z-S0P1.z;
Temp_Point_2=S1P1;
Temp_Point_2.x=S1P1.x-((X_DEG/Dist_S0P0S1P0)*(Dist_S0P0S1P0*Right_Width));
Temp_Point_2.y=S1P1.y-((Y_DEG/Dist_S0P0S1P0)*(Dist_S0P0S1P0*Right_Width));
Temp_Point_2.z=S1P1.z-((Z_DEG/Dist_S0P0S1P0)*(Dist_S0P0S1P0*Right_Width));
Temp_Point=Temp_Point_2;
X_DEG=S1P1.x-S1P1_H.x;
Y_DEG=S1P1.y-S1P1_H.y;
Z_DEG=S1P1.z-S1P1_H.z;
Temp_Point_2.x=Temp_Point_2.x-((X_DEG/Dist_S1P0S1P0_H)*(Dist_S1P0S1P0_H*OG_Height));
Temp_Point_2.y=Temp_Point_2.y-((Y_DEG/Dist_S1P0S1P0_H)*(Dist_S1P0S1P0_H*OG_Height));
Temp_Point_2.z=Temp_Point_2.z-((Z_DEG/Dist_S1P0S1P0_H)*(Dist_S1P0S1P0_H*OG_Height));
XYZ_Dist_2_Points_Calc(Temp_Point,Temp_Point_2);
Temp_Distance=abs(Dist_current_Points);
if(Temp_Distance<=Obergurt_Mindesthoehe)
{
  Temp_Point_2.y=Temp_Point.y-Obergurt_Mindesthoehe;
}
//PunktRechtsUnten - back
X_DEG=S1P1_H.x-S0P1_H.x;
Y_DEG=S1P1_H.y-S0P1_H.y;
Z_DEG=S1P1_H.z-S0P1_H.z;
Temp_Point_3=S1P1_H;
Temp_Point_3.x=S1P1_H.x-((X_DEG/Dist_S1P0_HS0P0_H)*(Dist_S1P0_HS0P0_H*Right_Width));
Temp_Point_3.y=S1P1_H.y-((Y_DEG/Dist_S1P0_HS0P0_H)*(Dist_S1P0_HS0P0_H*Right_Width));
Temp_Point_3.z=S1P1_H.z-((Z_DEG/Dist_S1P0_HS0P0_H)*(Dist_S1P0_HS0P0_H*Right_Width));
Temp_Point=Temp_Point_3;
X_DEG=S1P1_H.x-S1P1.x;
Y_DEG=S1P1_H.y-S1P1.y;
Z_DEG=S1P1_H.z-S1P1.z;
Temp_Point_3.x=Temp_Point_3.x-((X_DEG/Dist_S1P0S1P0_H)*(Dist_S1P0S1P0_H*UG_Height));
Temp_Point_3.y=Temp_Point_3.y-((Y_DEG/Dist_S1P0S1P0_H)*(Dist_S1P0S1P0_H*UG_Height));
Temp_Point_3.z=Temp_Point_3.z-((Z_DEG/Dist_S1P0S1P0_H)*(Dist_S1P0S1P0_H*UG_Height));
XYZ_Dist_2_Points_Calc(Temp_Point,Temp_Point_3);
Temp_Distance=abs(Dist_current_Points);
if(Temp_Distance<=Untergurt_Mindesthoehe)
{
  Temp_Point_3.y=Temp_Point.y+Untergurt_Mindesthoehe;
}
//PunktLinksUnten - back
X_DEG=S0P1_H.x-S1P1_H.x;
Y_DEG=S0P1_H.y-S1P1_H.y;
Z_DEG=S0P1_H.z-S1P1_H.z;
Temp_Point_4=S0P1_H;
Temp_Point_4.x=S0P1_H.x-((X_DEG/Dist_S1P0_HS0P0_H)*(Dist_S1P0_HS0P0_H*Left_Width));
Temp_Point_4.y=S0P1_H.y-((Y_DEG/Dist_S1P0_HS0P0_H)*(Dist_S1P0_HS0P0_H*Left_Width));
Temp_Point_4.z=S0P1_H.z-((Z_DEG/Dist_S1P0_HS0P0_H)*(Dist_S1P0_HS0P0_H*Left_Width));

Temp_Point=Temp_Point_4;
X_DEG=S0P1_H.x-S0P1.x;
Y_DEG=S0P1_H.y-S0P1.y;
Z_DEG=S0P1_H.z-S0P1.z;
Temp_Point_4.x=Temp_Point_4.x-((X_DEG/Dist_S0P0_HS0P0)*(Dist_S0P0_HS0P0*UG_Height));
Temp_Point_4.y=Temp_Point_4.y-((Y_DEG/Dist_S0P0_HS0P0)*(Dist_S0P0_HS0P0*UG_Height));
Temp_Point_4.z=Temp_Point_4.z-((Z_DEG/Dist_S0P0_HS0P0)*(Dist_S0P0_HS0P0*UG_Height));
XYZ_Dist_2_Points_Calc(Temp_Point,Temp_Point_4);
Temp_Distance=abs(Dist_current_Points);
if(Temp_Distance<=Untergurt_Mindesthoehe)
{
  Temp_Point_4.y=Temp_Point.y+Untergurt_Mindesthoehe;
}
//Punkte zuweisen
S0P1=Temp_Point_1;
```

```
      S1P1=Temp_Point_2;
      S1P1_H=Temp_Point_3;
      S0P1_H=Temp_Point_4;

      if(abs(Temp_Point_1.y-Temp_Point_4.y)<=(Minimal_Construction_Height))
      {
        S0P1=(Temp_Point_1+Temp_Point_4)/2;
        S0P1_H=(Temp_Point_1+Temp_Point_4)/2;
      }
      if(abs(Temp_Point_2.y-Temp_Point_3.y)<=(Minimal_Construction_Height))
      {
        S1P1=(Temp_Point_2+Temp_Point_3)/2;
        S1P1_H=(Temp_Point_2+Temp_Point_3)/2;
      }

      CUBE_VOLUMES_BALCANCE_POINTS_CALC(S0P0,S1P0,S0P1,S1P1,S0P0_H,S1P0_H,S0P1_H,S1P1_H,cr
      Negativ_Cubes_Balance_Point[cnt][a]=Cube_Volume_Balance_Point;
      Negativ_Cubes_Volume[cnt][a]=Single_Cube_Volume;
      Negativ_Entire_Volume=Negativ_Entire_Volume+ Negativ_Cubes_Volume[cnt][a];

      //Summe
      Sum_of_Pyramides_Volumes[cnt][a]=Positiv_Cubes_Volume[cnt][a]-Negativ_Cubes_Volume[cnt][a];
      Sum_Cubes_Balance_Points[cnt][a]=Positiv_Cubes_Balance_Point[cnt][a]; //DIES STIMMT NOCH NICHT -
      //println("Volumsbrechenung: Positiv_Cubes_Balance_Point[cnt][a]: ",Positiv_Cubes_Balance_Point[cnt][a])
      //println("Volumsbrechenung: Sum_Cubes_Balance_Points[cnt][a]: ",Sum_Cubes_Balance_Points[cnt][a]);
      Entire_Structure_Volume=Entire_Structure_Volume+Sum_of_Pyramides_Volumes[cnt][a];
    }
  }
  //println("Das Positive_Gesamtvolumen beträgt: ",Positiv_Entire_Volume,"[cm³]");
  println("Das Positive_Gesamtvolumen beträgt: ",Positiv_Entire_Volume/1000000,"[m³]");
  //println("Das Negative_Gesamtvolumen beträgt: ",Negativ_Entire_Volume,"[cm³]");
  println("Das Negative_Gesamtvolumen beträgt: ",Negativ_Entire_Volume/1000000,"[m³]");
  //println("Das Structur_Volumen beträgt: ",Entire_Structure_Volume,"[cm³]");
  println("Das Structur_Volumen beträgt: ",Entire_Structure_Volume/1000000,"[m³]");
}


//Profiltypen für den Brückenquerschnitt definieren //5 Typen je nach Brückenbreite und Höhe
DEFINE_PROFILETYPES()
{
  println(" ");
  println("SCHNITTPROFILE DEFINIEREN");
  var a,cnt,i;
  var P1,P2;
  var LastSpline=Anzahl_Splines-1,FirstSpline=0;
  var Bridge_Width,Brigde_Height,current_Element_Width;
  var Num_Profiles,StepSize,Real_StepSize,Real_StepSize_Width;
  var Max_Element_Width,Min_Element_Width,Vollprofil_Mindest_Breite;
  var Middle_Profil_Height,Spline_Steps, ProfilAxis=False;
  var Mittlere_Spline_1,Mittlere_Spline_2,Abweichung,Temp_Height;
  var ProfilAxis_Spline;
  var Num_Part_Profiles; //Wenn ein Steg sehr dick ist, dann muss er aus mehreren Profilen zusammengesetz
  var ProfilPunkte=Anzahl_Splines*2;
  var DownSpline;
  PROFILETYPE=new(array,Anzahl_Splines,SplinePunkte);

  //Kammerbreiten vergeben
  if(Material_Typ==0)  //STAHL
  {
    Max_Element_Width=200;//Es müssen mindestens alles 200 meter stege sein.
    Min_Element_Width=20; //kleinere Kammern lassen sich schwerer herstellen und machen höchstwahrschei
  }
  else                    //BETON
  {
      Max_Element_Width=200;//Es müssen mindestens alles 200 meter stege sein.
      Min_Element_Width=50; //höchstens alle 80cm ein Stegkleinere Kammern lassen sich schwerer betonier
  }
  //PROFILTYPEN
  //1_Vollprofil
  //2_C-Profil
  //3_nur Ober und Untergurt
  //4_verkehrtes C-Profil
  //5_Hohlprofil
  //-----------------------------------------------------------------
  if(Beam_Number_Intern<=3) //Dem Idealträger, dem 2. Träger und dem 3. Träger werden Vollquerschnitt zug
  {
    if(Beam_Number_Intern==1)
    {
      println("Für die Idealträgerberechnung werden alles VOLLPROFILE vergeben");
    }
    else println("Es werden nochmals VOLLPROFILE vergeben");
    for (a=0;a<SplinePunkte;a++)
    {
      for (cnt=0;cnt<Anzahl_Splines;cnt++)
      {
        PROFILETYPE[cnt][a]=1;//Vollprofile
      }
    }
  }
  else
  {
    for (a=0;a<SplinePunkte;a++)
    {
      //Durchschnittshöhe berechnen
      Middle_Profil_Height=0;
      //println("Schnitt: ",a);
      for (cnt=0;cnt<Anzahl_Splines;cnt++)
      {
        P1=BRIDGE_VOLUME_POINTS_ARRAY[cnt][a];
        DownSpline=(ProfilPunkte-1)-cnt;
        P2=BRIDGE_VOLUME_POINTS_ARRAY[DownSpline][a];
        XYZ_Dist_2_Points_Calc(P1,P2);
        Temp_Height=Dist_current_Points;
        Middle_Profil_Height=Middle_Profil_Height+Temp_Height;
        //println(cnt," Temp_Height: ",Temp_Height);
      }
      Middle_Profil_Height=Middle_Profil_Height/Anzahl_Splines;

      //println("Profil ",a," Minimal_Construction_Height: ",Minimal_Construction_Height);
      //println("Profil ",a," Middle_Profil_Height: ",Middle_Profil_Height);
      //1. TEIL: Breiten, und Stegabstände, für aktuellen Schnitt ermitteln
      //Schnitt_Breite berechnen
      P1=BRIDGE_POINTS_ARRAY[FirstSpline][a];
      P2=BRIDGE_POINTS_ARRAY[LastSpline][a];
      XZ_Dist_2_Points_Calc(P1,P2);
      Bridge_Width=abs(Dist_current_Points);
      //println("BRÜCKENBREITE: ",Bridge_Width);
      //MITTLERE SPLINES/-s Ermitteln----------------------------------------------------------------
       Abweichung=Anzahl_Splines/2;
       if(Abweichung!=int(Abweichung)) //Wenn die Abweichung z.B.: Abweichung 9.5 ist, dann ist sie ungleich int(Abweichung)=9. -> Erg
       {
         //ungerade Splineanzahl
         Mittlere_Spline_1=int(Anzahl_Splines/2)+1; //wenn immer abgerundet wird...dann stimmt dies //Ansonsten...
         Mittlere_Spline_2=int(Anzahl_Splines/2)+1; //= Spline 1
       }
       else
       {
         //gerade Splineanzahl=2 Mittelsplines
         Mittlere_Spline_1=int(Anzahl_Splines/2)-1;
         Mittlere_Spline_2=int(Anzahl_Splines/2);
       }
       //println("Mittlere_Spline_1=",Mittlere_Spline_1," Mittlere_Spline_2=",Mittlere_Spline_2);
       //StegAbstand-Schrittweite  Ideales Stegabstands_Verhältnis = 1/0.4=2.25=>H/B=2.5->Beam_Proportion_Faktor
       if(Beam_Proportion_Faktor!=0) //ACHTUNG -> Dies ob die folgenden 5 Zeilen Sinn machen ist erst zu überprüfen
       {
         Real_StepSize_Width=Middle_Profil_Height/Beam_Proportion_Faktor; //Idealer Stegabstand in cm.
       }
       else Real_StepSize_Width=Middle_Profil_Height; //
       if(Real_StepSize_Width>=Max_Element_Width) //Wenn der Wegabstand größer als der Maximal erlaubte Stegabstand ist dann setze
maximalen Stegabstand
       {
         Real_StepSize_Width=Max_Element_Width;
         //println("[",cnt,"][",a,"] Real_StepSize_Width=Max_Element_Width");
       }
       if(Real_StepSize_Width<=Min_Element_Width) //Wenn der Wegabstand kleiner als der minimal erlaubte Stegabstand ist dann setze
minimalen Stegabstand
       {
         Real_StepSize_Width=Min_Element_Width;
         //println("[",cnt,"][",a,"] Real_StepSize_Width=Min_Element_Width");
       }

       //MAXIMALE STEGANZAHL ERMITTELN
       Real_StepSize=Bridge_Width/Real_StepSize_Width; //Brückenbreite /Elementbreite= Anzahl der Elemente ->
       StepSize=int(Real_StepSize); //Da es keine Halben Stege gibt, wird abgerundet.

       //Wieviele Profileprofile werden benötigt?
       Num_Profiles=StepSize+1; //Anzahl der geforderten Stege //alte version ohne +1
       //Welche Splines werden als ProfilAchsen verwendet?
       //jede Spline_Steps Spline
       Spline_Steps=1; //Würde bedeuten jede Spline
       if(Num_Profiles-1!=0) Spline_Steps=Anzahl_Splines/(Num_Profiles); //z.B.: 20 Splines/4 Profile = 5 Steps...-> alle 5 Splines ist ein
Num_Profiles-1
       //println("-----------------------------------------------------------------------------------------------------------");
       for (cnt=0;cnt<=Mittlere_Spline_1;cnt++)
       {
         // Anfangs wird jedem Profil der 3_Typ zugewießen
         PROFILETYPE[cnt][a]=3;   //3_nur Ober und Untergurt
         if(Middle_Profil_Height>(Minimal_Construction_Height)) //Wenn die Mittlere Profilhöhe größer ist als die Minimale Konstruktionshi
```

```
5157        {
5158          //println("1. Hälfte: PROFILETYPE[",cnt,"][",a,"]. - Es wird vorerst einmal ein Obergurt-Untergurtprofil v
5159          //Teilquerschnittbreite berechnen - kleine Elementbreite
5160          P1=BRIDGE_POINTS_ARRAY[cnt][a];
5161          P2=BRIDGE_POINTS_ARRAY[cnt+1][a];
5162          XZ_Dist_2_Points_Calc(P1,P2);
5163          current_Element_Width=abs(Dist_current_Points);
5164
5165          //2. TEIL UNTERSUCHEN OB AKTURELLE SPLINE EINE ACHSENSPLINE IST
5166          ProfilAxis=False;
5167          //Von Rand zur Mittelspline hin untersuchen
5168          //1.HÄLFTE
5169          ProfilAxis_Spline=0;
5170          for (i=0;i<=Mittlere_Spline_1;i++)
5171          {
5172            //println(i," ProfilAxis_Spline=",ProfilAxis_Spline);
5173            if(cnt==ProfilAxis_Spline) //Noch falsch
5174            {
5175              //println("Spline ",cnt," ist ein Achsprofil");
5176              ProfilAxis=True;
5177            }
5178            //println(i," Spline_Steps=",Spline_Steps);
5179            ProfilAxis_Spline=ProfilAxis_Spline+Spline_Steps; //Hier werden die Splines vom Rand zur Mitte hin
5180          }
5181          P1=BRIDGE_VOLUME_POINTS_ARRAY[cnt][a];
5182          DownSpline=(ProfilPunkte-1)-cnt;
5183          P2=BRIDGE_VOLUME_POINTS_ARRAY[DownSpline][a];
5184          XYZ_Dist_2_Points_Calc(P1,P2);
5185          Temp_Height=Dist_current_Points;
5186          //println("Height",Temp_Height);
5187          //println("P1[",cnt,"|",a,"]= ",P1," P2[",DownSpline,"|",a,"]= ",P2,"  1nd Half_Height: ",Temp_Height);
5188          //WENN AKTUELLE SPLINE EINE PROFIL ACHSE IST, DANN, BERECHNE HÖHE AN DER STELLE
5189          if(ProfilAxis==True)
5190          {
5191            //Wieviel sollte die Stegbreite (Vollprofilbreite) sein?
5192            Vollprofil_Mindest_Breite=Temp_Height*Stegbreiten_Faktor; //Voll-Profil z.B.: 10% der Breite
5193            //Wieviele Vollprofile werden benötigt?
5194            Num_Part_Profiles=abs(Vollprofil_Mindest_Breite/current_Element_Width);
5195            //if(Num_Part_Profiles<Anzahl_Splines) Num_Part_Profiles=Anzahl_Splines-1; //Maximal mögliche F
5196            //println("Num_Part_Profiles: ",Num_Part_Profiles," Vollprofil_Mindest_Breite: ",Vollprofil_Mindest_B
5197            var AchsVerschiebung=int(Num_Part_Profiles/2); //Weil die aktuelle Spline die Achsenmitte ist!!!
5198            var Vollprofil_Abweichung;
5199            var temp_Nr;
5200            Vollprofil_Abweichung=Num_Part_Profiles-int(Num_Part_Profiles); //Die Restbreite die Übrig bleibt -
5201
5202            //println("Benötigte Stegprofile: ",Num_Part_Profiles," | AchsVerschiebung von Spline: ",AchsVerschie
",Vollprofil_Abweichung);
5203            if((cnt-AchsVerschiebung)<=0) AchsVerschiebung=0;//-> wichtig für Rand
5204            i=0;
5205            for(i=0;i<int(Num_Part_Profiles);i++)
5206            {
5207              //println("Vollprofil_Abweichung:",Vollprofil_Abweichung," AchsVerschiebung: ",AchsVerschiebung,"
5208              PROFILETYPE[cnt+i-AchsVerschiebung][a]=1;//Vollprofil
5209              temp_Nr=cnt+i-AchsVerschiebung;
5210              //println("VOLLPROFIL wird für PROFILNUMMER[",temp_Nr,"][",a,"] vergeben!");
5211            }
5212            //i=i+1;
5213            //Was passiert wenn sich kein Vollprofil ausgeht oder eine komma Anzahl z.B.: 2.5 Vollprofile?
5214            if(Vollprofil_Abweichung!=0.0)
5215            {
5216              //println("int(Num_Part_Profiles):",i,"| Aktuelle Spline cnt:",cnt," | AchsVerschiebung: ",AchsVerschie
5217              //Eigentlich sollte die Wertigkeit aufgrund der Tragfähigkeit sein: Vollprofil, C-Profile, Hohlprofil,UGO
5218              //Bei gringer der Wert der Abweichung, kann ein niederwertiges Profil eingesetzt werden.
5219              //println("a:",a,"Vollprofil_Abweichung:",Vollprofil_Abweichung," AchsVerschiebung: ",AchsVerschiebt
5220              //Bei Trophy wurde überall Profil 4 eingesetzt...
5221            if(Material_Typ==0) //STAHL
5222            {
5223              if(Vollprofil_Abweichung>0.9)
5224              {
5225                PROFILETYPE[cnt+i-AchsVerschiebung][a]=1;//Vollprofil
5226                //println("Es wird ein Vollprofil vergeben für cnt: ",cnt+i-AchsVerschiebung," vergeben");
5227              }
5228              if(0.5<Vollprofil_Abweichung && Vollprofil_Abweichung<=0.9)
5229              {
5230                PROFILETYPE[cnt+i-AchsVerschiebung][a]=2;//C-Profil
5231                //println("Es wird ein C-Profil vergeben für cnt: ",cnt+i-AchsVerschiebung," vergeben");
5232              }
5233              if(0.4<Vollprofil_Abweichung && Vollprofil_Abweichung<=0.5)
5234              {
5235                PROFILETYPE[cnt+i-AchsVerschiebung][a]=5;//Hohlprofil --> Dieses kommt oft vor
5236                //println("Es wird ein Hohl-Profil vergeben für cnt: ",cnt+i-AchsVerschiebung," vergeben");
5237              }
5238              if(Vollprofil_Abweichung<=0.4)
5239              {
5240                PROFILETYPE[cnt+i-AchsVerschiebung][a]=3;//3_nur Ober und Untergurt
5241                //println("Es wird ein Ober- und Untergurt vergeben für cnt: ",cnt+i-AchsVerschiebung," verg
5242              }
5243            }
5244            if(Material_Typ==1) //Beton
5245            {
5246              if(Vollprofil_Abweichung>0.9) PROFILETYPE[cnt+i-AchsVerschiebung][a]=1;//Vollprofil
5247              if(0.5<Vollprofil_Abweichung<=0.9) PROFILETYPE[cnt+i-AchsVerschiebung][a]=2;//C-Profil
5248              if(0.4<Vollprofil_Abweichung<=0.5) PROFILETYPE[cnt+i-AchsVerschiebung][a]=3;//Hohlprofil
5249              if(Vollprofil_Abweichung<=0.4) PROFILETYPE[cnt+i-AchsVerschiebung][a]=3;//UGOG-Profil=r
5250            }
5251            //println("VollprofilAbweichung beträgt: ",Vollprofil_Abweichung," , dehalb wurde noch ein ",PR(
5252          }//if(Vollprofil_Abweichung!=0.0) ENDE
5253        }/// if(ProfilAxis==True)
5254        //else PROFILETYPE[cnt][a]=3;   //3_nur Ober und Untergurt
5255        if(Temp_Height<=Minimal_Construction_Height)
5256        { //println("Temp_Height<=Minimal_Construction_Height -> Es wird ein Vollprofil für PROFILETYPE
5257          PROFILETYPE[cnt][a]=1;
5258        }
5259      }//Middle_Profil_Height>(Minimal_Construction_Height)
5260      else
5261      {
5262        PROFILETYPE[cnt][a]=1;//Vollprofil //Wenn die Konstruktion sehr niedrig ist dann werden alles Voll
5263        //println("ELSE: Es wird ein Vollprofil für PROFILETYPE[",cnt-1,"][",a,"] vergeben! -  //Wenn die Kor
5264      }
5265      //println("Gewählter PROFILETYPE[",cnt,"][",a,"]: ",PROFILETYPE[cnt][a]);
5266      //println("----------------------------------------------------------------");
5267    } //if(Middle_Profil_Height>(Minimal_Construction_Height)) ENDE
5268
5269    //2. Hälfte----------------------------------------------------------------
5270    for (cnt=Anzahl_Splines-1;cnt>=Mittlere_Spline_2;cnt--)
5271    {
5272      //println("2. Hälfte: PROFILETYPE[",cnt,"][",a,"].- Es wird vorerst einmal ein Obergurt-Untergurtprofi
5273      // Anfangs wird jedem Profil der 3_Typ zugewießen
5274      PROFILETYPE[cnt-1][a]=3;//3_nur Ober und Untergurt
5275      if(Middle_Profil_Height>(Minimal_Construction_Height)) //Wenn die Mittlere Profilhöhe größer ist al
5276      {
5277        //Teilquerschnittbreite berechnen - kleine Elementbreite
5278        P1=BRIDGE_POINTS_ARRAY[cnt-1][a];
5279        P2=BRIDGE_POINTS_ARRAY[cnt][a];
5280        XZ_Dist_2_Points_Calc(P1,P2);
5281        Current_Element_Width=abs(Dist_current_Points);
5282
5283        //2. TEIL UNTERSUCHEN OB AKTURELLE SPLINE EINE ACHSENSPLINE IST
5284        ProfilAxis=False;
5285        //Von Rand zur Mittelspline hin untersuchen
5286        //1.HÄLFTE
5287        ProfilAxis_Spline=0;
5288
5289        //2.HÄLFTE
5290        ProfilAxis_Spline=Anzahl_Splines-1;
5291        for (i=Anzahl_Splines-1;i>=Mittlere_Spline_2;i--)
5292        {
5293          if(cnt==ProfilAxis_Spline) //Noch falsch
5294          {
5295            //println("Spline ",cnt," ist ein Achsprofil");
5296            ProfilAxis=True;
5297          }
5298          //println("Spline_Steps=",Spline_Steps);
5299          ProfilAxis_Spline=ProfilAxis_Spline-Spline_Steps;//Hier werden die Splines vom Rand zur Mitte
5300        }
5301        P1=BRIDGE_VOLUME_POINTS_ARRAY[cnt][a];
5302        DownSpline=(ProfilPunkte-1)-cnt;
5303        P2=BRIDGE_VOLUME_POINTS_ARRAY[DownSpline][a];
5304        XYZ_Dist_2_Points_Calc(P1,P2);
5305        Temp_Height=Dist_current_Points;
5306        //println("Height",Temp_Height);
5307        //println("P1[",cnt,"|",a,"]= ",P1," | P2[",DownSpline,"|",a,"]= ",P2," | 2nd Half_Height: ",Temp_Height
5308        //WENN AKTUELLE SPLINE EINE PROFIL ACHSE IST, DANN, BERECHNE HÖHE AN DER STELLE
5309        if(ProfilAxis==True)
5310        {
5311          //Wieviel sollte die Stegbreite (Vollprofilbreite) sein?
5312          Vollprofil_Mindest_Breite=Temp_Height*Stegbreiten_Faktor; //Voll-Profil z.B.: 10% der Breite
5313          //Wieviele Vollprofile werden benötigt?
5314          Num_Part_Profiles=abs(Vollprofil_Mindest_Breite/current_Element_Width);
5315          var AchsVerschiebung=int(Num_Part_Profiles/2); //Weil die aktuelle Spline die Achsenmitte ist!!!
5316          var Vollprofil_Abweichung;
5317          var temp_Nr;
5318          Vollprofil_Abweichung=Num_Part_Profiles-int(Num_Part_Profiles); //Die Restbreite die Übrig ble
5319
5320          //println("Benötigte Stegprofile: ",Num_Part_Profiles," | AchsVerschiebung: ",AchsVerschiebung,"
5321          if((cnt+AchsVerschiebung)>=Anzahl_Splines-1) AchsVerschiebung=0;//-> wichtig für Rand
5322          i=0;
5323          for(i=0;i<int(Num_Part_Profiles);i++)
```

```
        {
          PROFILETYPE[cnt-1-i+AchsVerschiebung][a]=1;//Vollprofil
          temp_Nr=cnt-1-i+AchsVerschiebung;
          //println("VOLLPROFIL wird für PROFILETYPE[",temp_Nr,"][",a,"] vergeben!");
        }

        //println("Vollprofil_Abweichung:",Vollprofil_Abweichung," AchsVerschiebung: ",AchsVerschiebung," cr
        //Was passiert wenn sich kein Vollprofil ausgeht oder eine komma Anzahl z.B.: 2.5 Vollprofile?
        if(Vollprofil_Abweichung!=0.0)
        {
          //if(i>=Anzahl_Splines-1-cnt) i=Anzahl_Splines-1-cnt;
          //Eigentlich sollte die Wertigkeit aufgrund der Tragfähigkeit sein: Vollprofil, C-Profile, Hohlprofil,UGOG
          //Bei gringer der Wert der Abweichung, kann ein niederwertiges Profil eingesetzt werden.
          //Bei Trophy wurde überall Profil 2 eingesetzt...
          if(Material_Typ==0) //STAHL
          {
            if(Vollprofil_Abweichung>0.9)
            {
              PROFILETYPE[cnt-1-i+AchsVerschiebung][a]=1;//Vollprofil
              //println("Es wird ein Vollprofil für cnt: ",cnt-1-i+AchsVerschiebung," vergeben");
            }
            if(0.5<Vollprofil_Abweichung && Vollprofil_Abweichung<=0.9)
            {
              PROFILETYPE[cnt-1-i+AchsVerschiebung][a]=4;//umgekehrtes C-Profil
              //println("Es wird ein C-Profil für cnt: ",cnt-1-i+AchsVerschiebung," vergeben");
            }
            if(0.4<Vollprofil_Abweichung && Vollprofil_Abweichung<=0.5)
            {
              PROFILETYPE[cnt-1-i+AchsVerschiebung][a]=5;//Hohlprofil --> Dieses kommt oft vor
              //println("Es wird ein Hohl-Profil für cnt: ",cnt-1-i+AchsVerschiebung," vergeben");
            }
            if(Vollprofil_Abweichung<=0.4)
            {
              PROFILETYPE[cnt-1-i+AchsVerschiebung][a]=3;//3_nur Ober und Untergurt
              //println("Es wird ein Ober- und Untergurt für cnt: ",cnt-i+AchsVerschiebung," vergeben");
            }
          }
          if(Material_Typ==1) //STAHL
          {
            if(Vollprofil_Abweichung>0.9) PROFILETYPE[cnt-1-i+AchsVerschiebung][a]=1;//Vollprofil
            if(0.5<Vollprofil_Abweichung<=0.9) PROFILETYPE[cnt-1-i+AchsVerschiebung][a]=4;//umgekehrtes (
            if(0.4<Vollprofil_Abweichung<=0.5) PROFILETYPE[cnt-1-i+AchsVerschiebung][a]=3;//Hohlprofil -->
            if(Vollprofil_Abweichung<=0.4) PROFILETYPE[cnt-1-i+AchsVerschiebung][a]=3;//UGOG-Profil=norm
          }
          //println("VollprofilAbweichung beträgt: ",Vollprofil_Abweichung," , dehalb wurde noch ein ",PROFILE
        } // if(Vollprofil_Abweichung!=0.0) ENDE
      }//if(ProfilAxis==True)
      //else PROFILETYPE[cnt][a]=3;   //3_nur Ober und Untergurt
      if (Temp_Height<=Minimal_Construction_Height) PROFILETYPE[cnt-1][a]=1;
    }//if(Middle_Profil_Height)<(Minimal_Construction_Height)) ENDE
    else
    {
      PROFILETYPE[cnt-1][a]=1;//Vollprofil //Wenn die Konstruktion sehr niedrig ist dann werden alles Vollpro
      //println("ELSE: Es wird ein Vollprofil für PROFILETYPE[",cnt-1,"][",a,"] vergeben! -  //Wenn die Konstrul
    }
    //println("Gewählter PROFILETYPE[",cnt,"][",a,"]: ",PROFILETYPE[cnt][a]);
    //println("-----------------------------------------------------------------");
  } //Splinezählschleife Ende
 }//Schnitteschleife Ende
} // else ->if(Beam_Number_Intern<=2) ENDE
}

//STABILITY_CHECK
MOMENTS_OF_INERTIA_CALC()
{
 println(" ");
 println("FLÄCHENTRÄGHEITSMOMENTE BERECHNEN");
 //Je nach Ausgang dieser Berechnung werden,
 //A. die Pfade verschoben
 //B. die Dicken geändert
 //C. die Brücke mehr ausgehöhlt
 //je nach ausgang müssen das knet und path clearance modul angesteuert werden.
 //SEQUENCE
 //1. Schnittflächen und Profilschwerpunkte berechnen
 //2. Flächenträgheitsmomente um X un Y berechnen,
 //3. Sigma Zulässig Ober, Untergurt, /Sicherheitsbeiwerte berücksichtigen
 //4. Maßnahmen setzten - z.B.: 1. Über die Ausgänge - Bridgekonstruktion Fine = False|True; A:Pfad Punkte v
Aushöhlen), C: Verzerrung der Splinepunkte (außer den Punkten im Wegbereich);
 //5. ACHTUNG: Es muss ein Zähler programmiert werden, damit nie unendlich lange optimiert werden kann.

 var cnt,a;
 var TYP_Front;
 var S0P0,S0P0_H,S1P0_H,S1P0,S0P1,S0P1_H,S1P1_H,S1P1; //8 Volumspunkte
 var X_DEG,Y_DEG,Z_DEG,Temp_Point_1,Temp_Point,Temp_Distance,Temp_Point_2,Temp_Point_3,Temp_P
 var Temp_Section_Area_Balance_Point,Temp_Section_Moments_of_Inertia_Y,Temp_Section_Moments_of_Inertia_XZ,Temp_Area_S
 Section_Moments_of_Inertia_Y=new(array,SplinePunkte);
 Section_Moments_of_Inertia_XZ=new(array,SplinePunkte);
 Section_Area_Balance_Point=new(array,SplinePunkte);
 Section_Area=new(array,SplinePunkte);
 //2 verzogene Rechtecke berechnen
 var Rect_1_Area,Rect_1_Balance_Point,Rect_2_Area,Rect_2_Balance_Point;
 var Rect_Area,Rect_Balance_Point;
 var OUTER_Rect_Moments_of_Inertia_Y,OUTER_Rect_Moments_of_Inertia_Z;
 var INNER_Rect_Moments_of_Inertia_Y,INNER_Rect_Moments_of_Inertia_Z;
 var Part_Profil_Moments_of_Inertia_Y,Part_Profil_Moments_of_Inertia_Z,Part_Profil_Area;
 var Rect_Moments_of_Inertia_Y,Rect_Moments_of_Inertia_Z;
 var Rect_1_Moments_of_Inertia_Y,Rect_2_Moments_of_Inertia_Y;
 var Rect_1_Moments_of_Inertia_Z,Rect_2_Moments_of_Inertia_Z;
 var OUTER_Profil_Rect_Area,OUTER_Rect_Balance_Point=vector(0,0,0);
 var INNER_Profil_Rect_Area,INNER_Rect_Balance_Point=vector(0,0,0);
 var Triangle_1_Moments_of_Inertia_Z,Triangle_2_Moments_of_Inertia_Z;
 var Triangle_1_Moments_of_Inertia_Y,Triangle_2_Moments_of_Inertia_Y;
 var Temp_Dist_1,Temp_Dist_2,Temp_Dist_3,Temp_Point;
 var Triangle_1_Balcance_Point,Triangle_2_Balcance_Point;
 var Triangle_1_Area,Triangle_2_Area;
 var Temp_Section_Area;
 var Part_Profil_Sum_Balance_Point=vector(0,0,0);
 Rect_1_Balance_Point=vector(0,0,0);
 Rect_2_Balance_Point=vector(0,0,0);
 Rect_Balance_Point=vector(0,0,0);
 Triangle_1_Balcance_Point=vector(0,0,0);
 Triangle_2_Balcance_Point=vector(0,0,0);
 Temp_Point=vector(0,0,0);

 //Schleife - Alle Profilschitte und Kammern durchgehen
 for (a=0;a<SplinePunkte;a++)
 {
  Temp_Section_Moments_of_Inertia_Y=0;
  Temp_Section_Moments_of_Inertia_XZ=0;
  Temp_Section_Area_Balance_Point=vector(0,0,0);
  Temp_Section_Area=0.0;
  for (cnt=0;cnt<Anzahl_Splines-1;cnt++)
  {
   //1.Schnittflächen und Profilschwerpunkte  //MAKE_4_POINT_RECTANGLE(S0P0,S1P0,Right_M_Point,Left_M_Point);
   //Äußeres_Rechteck:
   var A_Point,B_Point,C_Point,D_Point;
   var Beam_Kontur_Name=(tostring(Beam_Number_Intern)+"BEAMKONTUR"+tostring(a));
   var Beam_Kontur_Object=doc->FindObject(Beam_Kontur_Name);

   A_Point=Beam_Kontur_Object->GetPoint(cnt);
   B_Point=Beam_Kontur_Object->GetPoint(cnt+1);
   C_Point=Beam_Kontur_Object->GetPoint(Anzahl_Splines*2-cnt-2);
   D_Point=Beam_Kontur_Object->GetPoint(Anzahl_Splines*2-cnt-1);

   //Inneres_Rechteck: S0P0,S1P0,S1P0_H,S0P0_H;
   //  B_Point  o---__
   //              |    ---__
   //              |  o-S1P0  ----o A_Point
   //              |  |  --- ___ |  |
   //              |  |            o S0P0
   //              |  |            |  |
   //              |  |S0P0_H o  |
   //      S1P0_H o--      __--o D_Point
   //              |  ___---
   //    Point C o--
   //
   //Flächenträgheits_Moment um Y= (h/6)*();
   //----------------------1. Rechteck: Außenpunkte---------
   //Teilflächenträgheitmomente berechnen
   //1. um Moment um Y // A.) 1. Dreieck B.) 2.Dreieck C.) Gesamtträgheitmoment
   //A.Dreieck
   XZ_Dist_2_Points_Calc(A_Point,B_Point);
   Temp_Dist_1=abs(Dist_current_Points);  //Breite
   Temp_Dist_2=abs(A_Point.y-B_Point.y);  //Höhe
   Triangle_1_Moments_of_Inertia_Y=(Temp_Dist_1*(Temp_Dist_2*Temp_Dist_2*Temp_Dist_2))/36;
   Triangle_1_Moments_of_Inertia_Z=(Temp_Dist_2*(Temp_Dist_1*Temp_Dist_1*Temp_Dist_1))/36;
   /*
   println(a,"|",cnt,"|----------------------------------------------------------------------------------------");
   println("Außen:");
   println("1.Dreiecks_Breite:",Temp_Dist_1);
   println("1.Dreiecks_Höhe:",Temp_Dist_2);
   println("1.Dreieck Moment_Y:",Triangle_1_Moments_of_Inertia_Y);
   println("1.Dreieck Moment_XZ:", Triangle_1_Moments_of_Inertia_Z);
   */
   //Kantenlängen für Flächenberechnung und Balance_Punkt ermitteln
   var Temp1_Point_A,Temp1_Point_B, Temp1_Point_C,Temp1_Point_D;
   if(A_Point.y<=B_Point.y)
   {
```

```
  Temp_Point=A_Point;
  Temp_Point.y=Temp_Point.y+Temp_Dist_2;
  Temp1_Point_A=Temp_Point;
  Temp1_Point_B=B_Point;
 }
 else
 {
  Temp_Point=B_Point;
  Temp_Point.y=Temp_Point.y+Temp_Dist_2;
  Temp1_Point_A=A_Point;
  Temp1_Point_B=Temp_Point;
 }

 Triangle_Area(A_Point,B_Point,Temp_Point);
 Triangle_1_Area=Triangle_Area_Value;
 Triangle_1_Balcance_Point=Triangle_Area_Balance_Point;
 //println("1. Dreiecksfläche: ",Triangle_1_Area);

 //B.Dreieck
 XZ_Dist_2_Points_Calc(D_Point,C_Point);
 Temp_Dist_1=abs(Dist_current_Points);  //Breite
 Temp_Dist_2=abs(D_Point.y-C_Point.y);  //Höhe
 Triangle_2_Moments_of_Inertia_Y=(Temp_Dist_1*(Temp_Dist_2*Temp_Dist_2*Temp_Dist_2))/36;
 Triangle_2_Moments_of_Inertia_Z=(Temp_Dist_2*(Temp_Dist_1*Temp_Dist_1*Temp_Dist_1))/36;
 /*
 println("2.Dreiecks_Breite:",Temp_Dist_1);
 println("2.Dreiecks_Höhe:",Temp_Dist_2);
 println("2.Dreieck Moment_Y:",Triangle_2_Moments_of_Inertia_Y);
 println("2.Dreieck Moment_XZ:", Triangle_2_Moments_of_Inertia_Z);
 */
 //Kantenlängen für Flächenberechnung und Balance_Punkt ermitteln
 if(D_Point.y<=C_Point.y)
 {
  Temp_Point=C_Point;
  Temp_Point.y=Temp_Point.y-Temp_Dist_2;
  Temp1_Point_C=Temp_Point;
  Temp1_Point_D=D_Point;
 }
 else
 {
  Temp_Point=D_Point;
  Temp_Point.y=Temp_Point.y-Temp_Dist_2;
  Temp1_Point_C=C_Point;
  Temp1_Point_D=Temp_Point;
 }
 Triangle_Area(D_Point,C_Point,Temp_Point);
 Triangle_2_Area=Triangle_Area_Value;
 Triangle_2_Balcance_Point=Triangle_Area_Balance_Point;
 //println("2. Dreiecksfläche: ",Triangle_2_Area);

 //C.) Trägheitmoment_Rechteck
 XZ_Dist_2_Points_Calc(A_Point,B_Point);
 Temp_Dist_1=abs(Dist_current_Points);  //Breite
 //Es müssen wirklich die äußersten Punkte gewählt werden -> ACHTUNG:Fehlerquelle
 //Die größre Länge ist die Höhe des umschließenden Rechtecks
 Temp_Dist_2=abs(Temp1_Point_A.y-Temp1_Point_D.y);  //Höhe
 Temp_Dist_3=abs(Temp1_Point_B.y-Temp1_Point_C.y);  //Höhe
 if(Temp_Dist_3>=Temp_Dist_2) Temp_Dist_2=Temp_Dist_3;
 Rect_Moments_of_Inertia_Y=(Temp_Dist_1*(Temp_Dist_2*Temp_Dist_2*Temp_Dist_2))/12;
 Rect_Moments_of_Inertia_Z=(Temp_Dist_2*(Temp_Dist_1*Temp_Dist_1*Temp_Dist_1))/12;
 /*
 println("Rechteck BREITE: ",Temp_Dist_1," HÖHE: ",Temp_Dist_2);
 println("Rechteck Moment_Y:",Rect_Moments_of_Inertia_Y);
 println("Rechteck Moment_XZ:", Rect_Moments_of_Inertia_Z);
 */
 var OUTER_Rect_Area=Temp_Dist_1*Temp_Dist_2;
 //println("Rechtecksfläche: ",OUTER_Rect_Area);
 //Rechtecksschwerpunkt = die 4 Ecken /4
 var OUTER_Rect_Balcance_Point=(Temp1_Point_A+Temp1_Point_B+Temp1_Point_C+Temp1_Point_D)/4;
 //Gesamt Flächenträgheitsmoment von dem Außenrechteck berechnen
 var OUTER_Profil_Balance_Point;

 Rectangle_Area(A_Point,B_Point,C_Point,D_Point); //Außen
 OUTER_Profil_Rect_Area=Rectangle_Area_Value;
 OUTER_Profil_Balance_Point=Rectangle_Area_Balance_Point;
 var Dist_Triangle_1_AchseY,Dist_Triangle_2_AchseY,Dist_Rec_Y;
 Dist_Triangle_1_AchseY=abs(OUTER_Profil_Balance_Point.y-Triangle_1_Balcance_Point.y);
 Dist_Triangle_2_AchseY=abs(OUTER_Profil_Balance_Point.y-Triangle_2_Balcance_Point.y);
 Dist_Rec_Y=abs(OUTER_Profil_Balance_Point.y-OUTER_Rect_Balcance_Point.y);
 /*
 println("1.Dreieck Y_Distanz: ",Dist_Triangle_1_AchseY);
 println("2.Dreieck Y_Distanz: ",Dist_Triangle_2_AchseY);
 println("Rechteck  Y_Distanz: ",Dist_Rec_Y);
 */
```

```
  OUTER_Rect_Moments_of_Inertia_Y=OUTER_Rect_Area*Dist_Rec_Y*Dist_Rec_Y+Rect_Moments_of_Inertia_Y-(Tria
Dist_Triangle_1_AchseY+Triangle_1_Moments_of_Inertia_Y+Triangle_2_Area*Dist_Triangle_2_AchseY*Dist_Triangle_2_
 //Satz von Steiner
  //println(a,"|",cnt," ÄUßERES Rechteck_PROFIL Trägheitsmoment_Y: ",OUTER_Rect_Moments_of_Inertia_Y);
  var Dist_Triangle_1_AchseXZ,Dist_Triangle_2_AchseXZ,Dist_Rec_XZ;
  XZ_Dist_2_Points_Calc(OUTER_Profil_Balance_Point,Triangle_1_Balcance_Point);
  Dist_Triangle_1_AchseXZ=abs(Dist_current_Points);
  XZ_Dist_2_Points_Calc(OUTER_Profil_Balance_Point,Triangle_2_Balcance_Point);
  Dist_Triangle_2_AchseXZ=abs(Dist_current_Points);
  XZ_Dist_2_Points_Calc(OUTER_Profil_Balance_Point,OUTER_Rect_Balcance_Point);
  Dist_Rec_XZ=abs(Dist_current_Points);
  OUTER_Rect_Moments_of_Inertia_Z=OUTER_Rect_Area*Dist_Rec_XZ*Dist_Rec_XZ+Rect_Moments_of_Inertia_Z-(
Dist_Triangle_1_AchseXZ+Triangle_1_Moments_of_Inertia_Z+Triangle_2_Area*Dist_Triangle_2_AchseXZ*Dist_Triangle_
Triangle_2_Moments_of_Inertia_Z);//Satz von Steiner
  /*
  println("1.Dreieck XZ_Distanz: ",Dist_Triangle_1_AchseXZ);
  println("2.Dreieck XZ_Distanz: ",Dist_Triangle_2_AchseXZ);
  println("Rechteck  XZ_Distanz: ",Dist_Rec_XZ);
  println(a,"|",cnt," ÄUßERES Rechteck_PROFIL Trägheitsmoment_XZ: ",OUTER_Rect_Moments_of_Inertia_Z);
  */
  //----------------------2. Rechteck: Innenpunkte---------
  //       S1P0  o------___ o S0P0
  //               |            |
  //               |            |
  //               |            |
  //               |    ___---o S0P0_H
  //    S1P0_H o---
  //
  var UPPER_Neagitv_Beam_Kontur_Name=("Beam_"+tostring(Beam_Number_Intern)+"_|"+tostring(cnt)+"|"+tostring(a
  var UPPER_Neagitv_Beam_Kontur_Object=doc->FindObject(UPPER_Neagitv_Beam_Kontur_Name);
  //z.B: Beam_1_|18|19| UPPER_NEGATIV_RECT //ACHTUNG bedeutet: Erste Brücke 18. oberer Negativhohlraum_Loft
  S0P0=UPPER_Negativ_Beam_Kontur_Object->GetPoint(0);
  S1P0=UPPER_Negativ_Beam_Kontur_Object->GetPoint(1);
  var LOWER_Neagitv_Beam_Kontur_Name=("Beam_"+tostring(Beam_Number_Intern)+"_|"+tostring(cnt)+"|"+tostring(a
  var LOWER_Negativ_Beam_Kontur_Object=doc->FindObject(LOWER_Neagitv_Beam_Kontur_Name);
  //z.B:Beam_1_|18|19| LOWER_NEGATIV_RECT
  S1P0_H=LOWER_Negativ_Beam_Kontur_Object->GetPoint(2);
  S0P0_H=LOWER_Negativ_Beam_Kontur_Object->GetPoint(3);

  //A.Dreieck
  XZ_Dist_2_Points_Calc(S0P0,S1P0);
  Temp_Dist_1=abs(Dist_current_Points);  //Breite
  Temp_Dist_2=abs(S0P0.y-S1P0.y);  //Höhe
  Triangle_1_Moments_of_Inertia_Y=(Temp_Dist_1*(Temp_Dist_2*Temp_Dist_2*Temp_Dist_2))/36;
  Triangle_1_Moments_of_Inertia_Z=(Temp_Dist_2*(Temp_Dist_1*Temp_Dist_1*Temp_Dist_1))/36;
  /*
  println("INNEN:");
  println("1.Dreieck Moment_Y:",Triangle_1_Moments_of_Inertia_Y);
  println("1.Dreieck Moment_XZ:", Triangle_1_Moments_of_Inertia_Z);
  */
  if(S0P0.y<=S1P0.y)
  {
   Temp_Point=S0P0;
   Temp_Point.y=Temp_Point.y+Temp_Dist_2;
   Temp1_Point_A=Temp_Point;
   Temp1_Point_B=S1P0;
  }
  else
  {
   Temp_Point=S1P0;
   Temp_Point.y=Temp_Point.y+Temp_Dist_2;
   Temp1_Point_A=S0P0;
   Temp1_Point_B=Temp_Point;
  }
  Triangle_Area(S0P0,S1P0,Temp_Point);
  Triangle_1_Area=Triangle_Area_Value;
  Triangle_1_Balcance_Point=Triangle_Area_Balance_Point;
  //println("1. Dreiecksfläche: ",Triangle_1_Area);
  //B.Dreieck
  XZ_Dist_2_Points_Calc(S0P0_H,S1P0_H);
  Temp_Dist_1=abs(Dist_current_Points);  //Breite
  Temp_Dist_2=abs(S0P0_H.y-S1P0_H.y);  //Höhe
  Triangle_2_Moments_of_Inertia_Y=(Temp_Dist_1*(Temp_Dist_2*Temp_Dist_2*Temp_Dist_2))/36;
  Triangle_2_Moments_of_Inertia_Z=(Temp_Dist_2*(Temp_Dist_1*Temp_Dist_1*Temp_Dist_1))/36;

  //println("2.Dreieck Moment_Y:",Triangle_2_Moments_of_Inertia_Y);
  //println("2.Dreieck Moment_XZ:", Triangle_2_Moments_of_Inertia_Z);
  if(S0P0_H.y<=S1P0_H.y)
  {
   Temp_Point=S1P0_H;
   Temp_Point.y=Temp_Point.y-Temp_Dist_2;
   Temp1_Point_C=Temp_Point;
   Temp1_Point_D=S0P0_H;
  }
```

```
    else
    {
      Temp_Point=S0P0_H;
      Temp_Point.y=Temp_Point.y-Temp_Dist_2;
      Temp1_Point_C=S1P0_H;
      Temp1_Point_D=Temp_Point;
    }
    Triangle_Area(S1P0_H,S0P0_H,Temp_Point);
    Triangle_2_Area=Triangle_Area_Value;
    Triangle_2_Balcance_Point=Triangle_Area_Balance_Point;
    //println("2. Dreiecksfläche: ",Triangle_2_Area);

    //C.) Trägheitmoment_Rechteck
    XZ_Dist_2_Points_Calc(S0P0,S1P0);
    Temp_Dist_1=abs(Dist_current_Points);  //Breite
    //Die größre Länge ist die Höhe des umschließenden Rechtecks
    Temp_Dist_2=abs(Temp1_Point_A.y-Temp1_Point_D.y);  //Höhe
    Temp_Dist_3=abs(Temp1_Point_B.y-Temp1_Point_C.y);  //Höhe
    if(Temp_Dist_3)>=Temp_Dist_2) Temp_Dist_2=Temp_Dist_3;
    Rect_Moments_of_Inertia_Y=(Temp_Dist_1*(Temp_Dist_2*Temp_Dist_2*Temp_Dist_2))/12;
    Rect_Moments_of_Inertia_Z=(Temp_Dist_2*(Temp_Dist_1*Temp_Dist_1*Temp_Dist_1))/12;
    /*
    println("Rechteck BREITE: ",Temp_Dist_1," HÖHE: ",Temp_Dist_2);
    println("Rechteck Moment_Y:",Rect_Moments_of_Inertia_Y);
    println("Rechteck Moment_XZ:", Rect_Moments_of_Inertia_Z);
    */
    var INNER_Rect_Area=Temp_Dist_1*Temp_Dist_2;
    //println("Rechtecksfläche: ",OUTER_Rect_Area);
    //Rechtecksschwerpunkt = die 4 Ecken /4
    var INNER_Rect_Balcance_Point=(Temp1_Point_A+Temp1_Point_B+Temp1_Point_C+Temp1_Point_D)/4;
    //Gesamt Flächenträgheitsmoment von dem Innenrechteck berechnen
    var INNER_Profil_Balance_Point;
    Rectangle_Area(S0P0,S1P0,S1P0_H,S0P0_H); //INNEN
    INNER_Profil_Rect_Area=Rectangle_Area_Value;
    INNER_Profil_Balance_Point=Rectangle_Area_Balance_Point;

    var Dist_Triangle_1_AchseY,Dist_Triangle_2_AchseY,Dist_Rec_Y;
    Dist_Triangle_1_AchseY=abs(INNER_Profil_Balance_Point.y-Triangle_1_Balcance_Point.y);
    Dist_Triangle_2_AchseY=abs(INNER_Profil_Balance_Point.y-Triangle_2_Balcance_Point.y);
    Dist_Rec_Y=abs(INNER_Profil_Balance_Point.y- INNER_Rect_Balcance_Point.y);
    /*
    println("1.Dreieck Y_Distanz: ",Dist_Triangle_1_AchseY);
    println("2.Dreieck Y_Distanz: ",Dist_Triangle_2_AchseY);
    println("Rechteck  Y_Distanz: ",Dist_Rec_Y);
    */
    INNER_Rect_Moments_of_Inertia_Y=INNER_Rect_Area*Dist_Rec_Y*Dist_Rec_Y+Rect_Moments_of_Inert
Dist_Triangle_1_AchseY+Triangle_1_Moments_of_Inertia_Y+Triangle_2_Area*Dist_Triangle_2_AchseY*Dist_T
//Satz von Steiner
    //println(a,"|",cnt," INNERES Rechteck_PROFIL Trägheitsmoment_Y: ",INNER_Rect_Moments_of_Inertia_Y
    var Dist_Triangle_1_AchseXZ,Dist_Triangle_2_AchseXZ,Dist_Rec_XZ;
    XZ_Dist_2_Points_Calc(INNER_Profil_Balance_Point,Triangle_1_Balcance_Point);
    Dist_Triangle_1_AchseXZ=abs(Dist_current_Points);
    XZ_Dist_2_Points_Calc(INNER_Profil_Balance_Point,Triangle_2_Balcance_Point);
    Dist_Triangle_2_AchseXZ=abs(Dist_current_Points);
    XZ_Dist_2_Points_Calc(INNER_Profil_Balance_Point,INNER_Rect_Balcance_Point);
    Dist_Rec_XZ=abs(Dist_current_Points);
    INNER_Rect_Moments_of_Inertia_Z=INNER_Rect_Area*Dist_Rec_XZ*Dist_Rec_XZ+Rect_Moments_of_In
Dist_Triangle_1_AchseXZ+Triangle_1_Moments_of_Inertia_Z+Triangle_2_Area*Dist_Triangle_2_AchseXZ*Dis
Triangle_2_Moments_of_Inertia_Z);//Satz von Steiner
    /*
    println("1.Dreieck XZ_Distanz: ",Dist_Triangle_1_AchseXZ);
    println("2.Dreieck XZ_Distanz: ",Dist_Triangle_2_AchseXZ);
    println("Rechteck  XZ_Distanz: ",Dist_Rec_XZ);
    println(a,"|",cnt," INNERES Rechteck_PROFIL Trägheitsmoment_XZ: ",INNER_Rect_Moments_of_Inertia_Z)
    */
    //AUßEN und INNEN FLÄCHENTRÄGHEITSMOMENTE VEREINEN
    var OUTER_Dist_Axis_Y,OUTER_Dist_Axis_XZ,INNER_Dist_Axis_Y,INNER_Dist_Axis_XZ;
    Part_Profil_Sum_Balance_Point=((OUTER_Profil_Rect_Area*OUTER_Profil_Balance_Point)+(INNER_Prof
OUTER_Profil_Rect_Area+INNER_Profil_Rect_Area);//Teil_Profil_Schwerpunkt berechnen
    OUTER_Dist_Axis_Y=abs(Part_Profil_Sum_Balance_Point.y-OUTER_Profil_Balance_Point.y);
    INNER_Dist_Axis_Y=abs(Part_Profil_Sum_Balance_Point.y-INNER_Profil_Balance_Point.y);
    //Teilprofil Y-Flächenträgheitsmoment
    Part_Profil_Moments_of_Inertia_Y=(OUTER_Profil_Rect_Area*OUTER_Dist_Axis_Y*OUTER_Dist_Axis_Y+
INNER_Profil_Rect_Area*INNER_Dist_Axis_Y*INNER_Dist_Axis_Y+INNER_Rect_Moments_of_Inertia_Y); //Sa
    XZ_Dist_2_Points_Calc(Part_Profil_Sum_Balance_Point,OUTER_Profil_Balance_Point);
    OUTER_Dist_Axis_XZ=abs(Dist_current_Points);
    XZ_Dist_2_Points_Calc(Part_Profil_Sum_Balance_Point,INNER_Profil_Balance_Point);
    INNER_Dist_Axis_XZ=abs(Dist_current_Points);
    //Teilprofil XZ-Flächenträgheitsmoment
    Part_Profil_Moments_of_Inertia_Z=(OUTER_Profil_Rect_Area*OUTER_Dist_Axis_XZ*OUTER_Dist_Axis_
INNER_Profil_Rect_Area*INNER_Dist_Axis_XZ*INNER_Dist_Axis_XZ+INNER_Rect_Moments_of_Inertia_Z); //
    Part_Profil_Area=OUTER_Profil_Rect_Area-INNER_Profil_Rect_Area;
    //println("TEILPROFILS_FLÄCHENTRÄGHEITSMOMENT_Y: ",Part_Profil_Moments_of_Inertia_Y);
    //println("TEILPROFILS_FLÄCHENTRÄGHEITSMOMENT_XZ: ", Part_Profil_Moments_of_Inertia_Z);
    //SCHNITT AKTUALISIEREN -> temporärer SchnittbalancePunkt und temp. Flächenträgheitsmoment
    var Dist1_Axis_Y,Dist1_Axis_XZ,Dist2_Axis_Y,Dist2_Axis_XZ,Old_Balcance_Point;
    Old_Balcance_Point=Temp_Section_Area_Balance_Point;
    //TEMPORÄRER SCHNITT MITTELPUNKT
    Temp_Section_Area_Balance_Point=(Temp_Section_Area_Balance_Point*Temp_Section_Area+Part_Profil_Sum_Balance_Point*P
Temp_Section_Area+Part_Profil_Area);
    Dist1_Axis_Y=abs(Temp_Section_Area_Balance_Point.y-Old_Balcance_Point.y); //Abstand aktueller Querschnittsbalancepunkt zum
    Dist2_Axis_Y=abs(Temp_Section_Area_Balance_Point.y-Part_Profil_Sum_Balance_Point.y);//Abstand aktueller Querschnittsbalanc
Teilquerschnittbalancepunkt
    //                                VORHERGEHENDER QUERSCHNITT                +                AKTUELLEM TEILQU
    Temp_Section_Moments_of_Inertia_Y=(Temp_Section_Area*Dist1_Axis_Y*Dist1_Axis_Y+Temp_Section_Moments_of_Inertia_Y)+(
*Dist2_Axis_Y+Part_Profil_Moments_of_Inertia_Y);//Satz von Steiner.
    XZ_Dist_2_Points_Calc(Temp_Section_Area_Balance_Point,Old_Balcance_Point);
    Dist1_Axis_XZ=abs(Dist_current_Points);
    XZ_Dist_2_Points_Calc(Temp_Section_Area_Balance_Point,Part_Profil_Sum_Balance_Point);
    Dist2_Axis_XZ=abs(Dist_current_Points);
    Temp_Section_Moments_of_Inertia_XZ=(Temp_Section_Area*Dist1_Axis_XZ*Dist1_Axis_XZ+Temp_Section_Moments_of_Inertia_
Dist2_Axis_XZ*Dist2_Axis_XZ+Part_Profil_Moments_of_Inertia_Z);//Satz von Steiner.
    Temp_Section_Area=Temp_Section_Area+Part_Profil_Area;

    //Überprüfung ob die richtigen Eckpunkte für die Rechtecke gewählt wurden
    var Scale=0.5; //Skallierung hat Auswirkung auf die SchwerpunktKugelgrößen
    var Eckpunkt_Name;
    Eckpunkt_Name=("|"+tostring(cnt)+"|"+tostring(a)+"|_Profil_Part_Balance_Point");
    //MAKE_SPHERE(Scale,Eckpunkt_Name,Temp_Section_Area_Balance_Point,Null);
    /*
    Eckpunkt_Name=("|"+tostring(cnt)+"|"+tostring(a)+"|_TEMP_SECTION_BALCANCEPOINT");
    MAKE_SPHERE(Scale,Eckpunkt_Name,Part_Profil_Sum_Balance_Point,Null);
    Eckpunkt_Name=("|"+tostring(cnt)+"|"+tostring(a)+"|_A_Point");
    MAKE_SPHERE(Scale,Eckpunkt_Name,A_Point,Null);
    Eckpunkt_Name=("|"+tostring(cnt)+"|"+tostring(a)+"|_B_Point");
    MAKE_SPHERE(Scale,Eckpunkt_Name,B_Point,Null);
    Eckpunkt_Name=("|"+tostring(cnt)+"|"+tostring(a)+"|_C_Point");
    MAKE_SPHERE(Scale,Eckpunkt_Name,C_Point,Null);
    Eckpunkt_Name=("|"+tostring(cnt)+"|"+tostring(a)+"|_D_Point");
    MAKE_SPHERE(Scale,Eckpunkt_Name,D_Point,Null);
    Eckpunkt_Name=("|"+tostring(cnt)+"|"+tostring(a)+"|_S0P0");
    MAKE_SPHERE(Scale,Eckpunkt_Name,S0P0,Null);
    Eckpunkt_Name=("|"+tostring(cnt)+"|"+tostring(a)+"|_S1P0");
    MAKE_SPHERE(Scale,Eckpunkt_Name,S1P0,Null);
    Eckpunkt_Name=("|"+tostring(cnt)+"|"+tostring(a)+"|_S1P0_H");
    MAKE_SPHERE(Scale,Eckpunkt_Name,S1P0_H,Null);
    Eckpunkt_Name=("|"+tostring(cnt)+"|"+tostring(a)+"|_S0P0_H");
    MAKE_SPHERE(Scale,Eckpunkt_Name,S0P0_H,Null);
    Scale=1;
    Eckpunkt_Name=("|"+tostring(cnt)+"|"+tostring(a)+"|_Temp1_Point_A");
    MAKE_SPHERE(Scale,Eckpunkt_Name,Temp1_Point_A,Null);
    Eckpunkt_Name=("|"+tostring(cnt)+"|"+tostring(a)+"|_Temp1_Point_B");
    MAKE_SPHERE(Scale,Eckpunkt_Name,Temp1_Point_B,Null);
    Eckpunkt_Name=("|"+tostring(cnt)+"|"+tostring(a)+"|_Temp1_Point_C");
    MAKE_SPHERE(Scale,Eckpunkt_Name,Temp1_Point_C,Null);
    Eckpunkt_Name=("|"+tostring(cnt)+"|"+tostring(a)+"|_Temp1_Point_D");
    MAKE_SPHERE(Scale,Eckpunkt_Name,Temp1_Point_D,Null);
    Scale=2;
    Eckpunkt_Name=("|"+tostring(cnt)+"|"+tostring(a)+"|_Outer_REC_BalancePoint");
    MAKE_SPHERE(Scale,Eckpunkt_Name,OUTER_Rect_Balcance_Point,Null);
    Eckpunkt_Name=("|"+tostring(cnt)+"|"+tostring(a)+"|_Profil_Outer_REC_BalancePoint");
    MAKE_SPHERE(Scale,Eckpunkt_Name,OUTER_Profil_Balance_Point,Null);
    Eckpunkt_Name=("|"+tostring(cnt)+"|"+tostring(a)+"|_Inner_REC_BalancePoint");
    MAKE_SPHERE(Scale,Eckpunkt_Name,INNER_Rect_Balcance_Point,Null);
    Eckpunkt_Name=("|"+tostring(cnt)+"|"+tostring(a)+"|_Profil_Inner_REC_BalancePoin");
    MAKE_SPHERE(Scale,Eckpunkt_Name,INNER_Profil_Balance_Point,Null);
    Scale=0.1;
    Eckpunkt_Name=("|"+tostring(cnt)+"|"+tostring(a)+"|_Inner_Tri_1_BalancePoin");
    MAKE_SPHERE(Scale,Eckpunkt_Name,Triangle_1_Balcance_Point,Null);
    Eckpunkt_Name=("|"+tostring(cnt)+"|"+tostring(a)+"|_Inner_Tri_2_BalancePoin");
    MAKE_SPHERE(Scale,Eckpunkt_Name,Triangle_2_Balcance_Point,Null);
    */
  }
  //println(a,".Kontroll Schnitt_Gesamtfläche:",Kontrollflaeche/10000);
  //Schnitte den Gesamtwert zuweisen:
  Section_Moments_of_Inertia_Y[a]=Temp_Section_Moments_of_Inertia_Y;
  Section_Moments_of_Inertia_XZ[a]=Temp_Section_Moments_of_Inertia_XZ;
  Section_Area_Balance_Point[a]=Temp_Section_Area_Balance_Point;
  Section_Area[a]=Temp_Section_Area;
  //println(a,"| SCHNITTFLÄCHENTRÄGHEITSMOMENT_Y: ", Section_Moments_of_Inertia_Y[a]);
  //println(a,"| SCHNITTFLÄCHENTRÄGHEITSMOMENT_XZ: ", Section_Moments_of_Inertia_XZ[a]);
  /*
  var Scale=2; //Skallierung hat Auswirkung auf die SchwerpunktKugelgrößen
  var Eckpunkt_Name,Punkt;
  Punkt=  Section_Area_Balance_Point[a];
  Eckpunkt_Name=("|"+tostring(cnt)+"|"+tostring(a)+"|_Profil_BALANCE_Point");
  MAKE_SPHERE(Scale,Eckpunkt_Name,Punkt,Null);
  */
```

```
  } //Schnittberechnen Ende
}

//BENDING_CALC
BENDING_CALC()
{
  println(" ");
  println("DURCHBIEGUNG DES TRÄGERS SCHÄTZEN");
  var a,i;
  var E_Modul,Average_Section_Moments_of_Inertia_Y;
  var temp_Moments_of_Inertia_Y=0.0;
  var Max_Bending_of_Arm_1,Max_Bending_of_Arm_2;
  //println("Hebelsarm 1: ",Cantilever_Arm_1);
  //println("Hebelsarm 2: ",Cantilever_Arm_2);
  Max_Bending_of_Arm_1=Cantilever_Arm_1/150;
  Max_Bending_of_Arm_2=Cantilever_Arm_2/150;
  if(Torque_Calc_Mode==2) Max_Bending_of_Arm_1=Cantilever_Arm_1/300;
  if(Torque_Calc_Mode==2) Max_Bending_of_Arm_2=Cantilever_Arm_1/300;

  //Section_Moments_of_Inertia_Y[a]);
  if(Material_Typ==0)    //Stahl: E= Elastizitätsmodul 21000 kN/cm2 210kN/mm²
  {
    E_Modul=21000;
  }
  else   E_Modul=2500;  //Beton: E= Elastizitätsmodul 2500 kN/cm2 25 kN/mm²

  //Achsenlängen für die Linienlast q berechnen.
  var Temp_Point_1,Temp_Point_2;
  Bridge_Segment_Lenght=new(array,SplinePunkte-1);
  Bridge_Lenght=0.0;
  for (a=0;a<SplinePunkte-1;a++)
  {
    Temp_Point_1=System_Axis_Points[a];
    Temp_Point_2=System_Axis_Points[a+1];
    XYZ_Dist_2_Points_Calc(Temp_Point_1,Temp_Point_2);
    Bridge_Segment_Lenght[a]=Dist_current_Points;
    Bridge_Lenght=Bridge_Lenght+Bridge_Segment_Lenght[a];
    //println("Brückensegment ",a," ist ",Bridge_Segment_Lenght[a],"m lang.");
  }
  println("Die Brückensystem_Achse ist ",Bridge_Lenght,"m lang.");

  var Arm_1_Load=0.0;
  var Arm_2_Load=0.0;
  var Bending_Arm_1_Lenght=0.0;
  var Bending_Arm_2_Lenght=0.0;

  //WINKELGEWICHTE IN JEDEM SCHNITT JE NACH AUFLAGERPOSITION BERECHNEN;
  Angle_Weight=new(array,SplinePunkte);
  println("Winkelgewichte berechnen:");
  //1. Randwert
  Angle_Weight[0]=(1/(E_Modul*Section_Moments_of_Inertia_Y[0]))*(Bridge_Segment_Lenght[0]/6)*(2*Sectic
  for(a=1;a<SplinePunkte-1;a++)
  {
    Angle_Weight[a]=(1/(E_Modul*Section_Moments_of_Inertia_Y[a]))*(Bridge_Segment_Lenght[a]/6)*(Section
Section_Torque_X[a+1]);
    //println("1/(E_Modul*Section_Moments_of_Inertia_Y[a]))*(Bridge_Segment_Lenght[a]/6)*(Section_Torque

//println("1/",E_Modul,"*",Section_Moments_of_Inertia_Y[a],"))*(",Bridge_Segment_Lenght[a],"/6)*(",Section_T
ue_X[a+1]);
    //println("Angle_Weight[",a,"]: ",Angle_Weight[a]);
  }
  //2.Randwert
  Angle_Weight[SplinePunkte-1]=(1/(E_Modul*Section_Moments_of_Inertia_Y[SplinePunkte-1]))*(Bridge_Segm
SplinePunkte-2]+2*Section_Torque_X[SplinePunkte-1]);
  if(Support_Position!=1 && Support_Position!=SplinePunkte)
  {
    Angle_Weight[Support_Position-1]=(1/(E_Modul*Section_Moments_of_Inertia_Y[Support_Position-1]))*(Brid
Section_Torque_X[Support_Position-2]+4*Section_Torque_X[Support_Position-1]+Section_Torque_X[Support
  }
  //Kragarnsberechnung mit Vertikal gespiegeltem System und den Angle_Weights als Einzellasten.
  Bending_Section=new(array,SplinePunkte);
  var temp_Bending;
  var temp_Distanz_Section;
  var Section_Position;
  var temp_Angle_Weight_Point_Position;
  var c;
  //Angle_Weight[a]--> Entspricht der Kraft an den Schnitten --> System_Axis_Points[a]
  for(a=0;a<SplinePunkte;a++)
  {
    Bending_Section[a]=0.0;
  }
  //---------------------------------verschiedene Momentenberechnungsmodi--------------------
  //Torque_Calc_Mode==0 muss umgekehrt zu Momentenberechnung verwendet werden...da wo in der Mome
  if(Torque_Calc_Mode==0) //KRAGARME VOM AUFLAGER WEG in Momentenberechnung Torque_Calc_Mode
{
  if(Support_Position!=1)
  {
    println("Biegung berechenen Arm 1 ursprünglich2 Kragarme von Uferzonen -> umgekehrtes System = 2 Kragarme vo
    for (c=1;c<=Support_Position-1;c++) //Alle Punkte von 1  bis zum Auflager zählen z.B.: max 1-19
    {
      //Die Summe aller Resultierenden die im Schnitt betrachtet werden multipliziert mit deren Abstand zum Schnitt.
      temp_Bending=0.0;
      Section_Position=System_Axis_Points[c];
      for (a=c;a>=0;a--) //Vom aktuellen Schnitt bis zum Schnitt 1 herunter zählen (z.B.: Schnitt 1 x Load 0)
      {
        temp_Angle_Weight_Point_Position=System_Axis_Points[a];
        XZ_Dist_2_Points_Calc(Section_Position,temp_Angle_Weight_Point_Position);
        temp_Distanz_Section=Dist_current_Points;
        temp_Bending=temp_Bending+temp_Distanz_Section*Angle_Weight[a];
      }
      Bending_Section[c]=temp_Bending;
      Bending_Section[0]=0;
    }
    for(c=SplinePunkte-2;c>=Support_Position;c--) //Alle Punkte von Oben nach Unten bis zum Auflager zählen z.B.: max
    {
      //println("SCHNITT ",c," Schleife B");
      temp_Bending=0.0;
      Section_Position=System_Axis_Points[c];
      //Die Summe aller Resultierenden die im Schnitt betrachtet werden multipliziert mit deren Abstand zum Schnitt.
      for (a=c-1;a<=SplinePunkte-1;a++) //Vom aktuellen Schnitt bis zum vorletzen Schnitt rechnen
      {
        temp_Angle_Weight_Point_Position=System_Axis_Points[a];
        XZ_Dist_2_Points_Calc(Section_Position,temp_Angle_Weight_Point_Position);
        temp_Distanz_Section=Dist_current_Points;
        temp_Bending=temp_Bending+temp_Distanz_Section*Angle_Weight[a];
      }
      Bending_Section[c]=temp_Bending;
    }
  } //if(Support_Position!=1) ENDE
  if(Support_Position==1) //Wenn Auflagerposition = 1 ist dann normaler Rechenmodus
  {
    //println("NORMALER BERECHNUNGSMODUS");
    for (c=SplinePunkte-2;c>=Support_Position-1;c--) //Alle Punkte von 1  bis zum Auflager zählen z.B.: max 0-19
    {
      temp_Bending=0.0;
      Section_Position=System_Axis_Points[c];
      //Die Summe aller Resultierenden die im Schnitt betrachtet werden multipliziert mit deren Abstand zum Schnitt.
      for (a=c;a<=SplinePunkte-1;a++) //Vom aktuellen Schnitt bis zum Schnitt 1 mal Load 0
      {
        temp_Angle_Weight_Point_Position=System_Axis_Points[a];
        XZ_Dist_2_Points_Calc(Section_Position,temp_Angle_Weight_Point_Position);
        temp_Distanz_Section=Dist_current_Points;
        temp_Bending=temp_Bending+temp_Distanz_Section*Angle_Weight[a];
      }
      Bending_Section[c]=temp_Bending;
      Bending_Section[SplinePunkte-1]=0;
    }
  }
} //if(Torque_Calc_Mode==1) ENDE

if(Torque_Calc_Mode==1)  //2 Kragarme von der Uferzone weg in Momentenberechnung Torque_Calc_Mode==0
{
  println("Biegung berechenen Arm 1 ursprünglich 2 Kragarme von Auflagerweg -> umgekehrtes System = 2 Kragarme v
  //SUPPORT IST KRAGARMSPITZE
  //1. ARM
  for(c=Support_Position-2;c>=0;c--) //Alle Punkte von Support bis nach 0 zählen z.B.: max 18-0 -Schnitte
  {
    temp_Bending=0.0;
    Section_Position=System_Axis_Points[c];
    //Die Summe aller Resultierenden die im Schnitt betrachtet werden multipliziert mit deren Abstand zum Schnitt.
    for (a=c+1;a<=Support_Position-1;a++)
    {
      temp_Angle_Weight_Point_Position=System_Axis_Points[a];
      XZ_Dist_2_Points_Calc(Section_Position,temp_Angle_Weight_Point_Position);
      temp_Distanz_Section=Dist_current_Points;
      temp_Bending=temp_Bending+temp_Distanz_Section*Angle_Weight[a];
    }
    Bending_Section[c]=temp_Bending;
    //Bending_Section[SplinePunkte-1]=0;
  } //1. Arm Ende
  //2. ARM
  for (c=Support_Position;c<=SplinePunkte-1;c++) //invertiert -> Alle Punkte von Unten nach Support zählen z.B.: max 0-
  {
    temp_Bending=0.0;
    Section_Position=System_Axis_Points[c];
    //println(" ");
    //println("Schnitt ",c);
    for (a=c-1;a>=Support_Position-1;a--)  //Alle Punkte von C nach unten zählen z.B.: (max c-0)
```

```
    {
      temp_Angle_Weight_Point_Position=System_Axis_Points[a];
      XZ_Dist_2_Points_Calc(Section_Position,temp_Angle_Weight_Point_Position);
      temp_Distanz_Section=Dist_current_Points;
      temp_Bending=temp_Bending+temp_Distanz_Section*Angle_Weight[a];
      //print(temp_Distanz_Section,"*",Angle_Weight[a]," + ");
    }
    Bending_Section[c]=temp_Bending;
    //println("Bending_Section[",c,"]: ",Bending_Section[c]);
  } //2.Arm Ende
 }//  if(Torque_Calc_Mode==0) ENDE

 if(Torque_Calc_Mode==2)  //Einfeldträger
 {
   //WINKELGEWICHTE IN JEDEM SCHNITT JE NACH AUFLAGERPOSITION BERECHNEN;
   Angle_Weight=new(array,SplinePunkte);
   Angle_Weight[0]=(1/(E_Modul*Section_Moments_of_Inertia_Y[0]))*(Bridge_Segment_Lenght[0]/6)*(2*Sect
   //println("Angle_Weight[",0,"]: ", Angle_Weight[0]);
   for(a=1;a<SplinePunkte-1;a++)
   {
     Angle_Weight[a]=(1/(E_Modul*Section_Moments_of_Inertia_Y[a]))*(Bridge_Segment_Lenght[a]/6)*(Sectic
Section_Torque_X[a+1]);
     //println("Angle_Weight[",a,"]: ", Angle_Weight[a]);
   }
   //Randwerte
   Angle_Weight[SplinePunkte-1]=(1/(E_Modul*Section_Moments_of_Inertia_Y[SplinePunkte-1]))*(Bridge_Segr
SplinePunkte-2]+2*Section_Torque_X[SplinePunkte-1]);
   //println("Angle_Weight[",SplinePunkte-1,"]: ", Angle_Weight[SplinePunkte-1]);
   //Summe Aller Angle_Weight
   var Sum_Angle_Weight=0.0;
   var temp_pos,Sum_Angle_Weight_Balance_Point;
   temp_pos=vector(0,0,0);
   for(a=0;a<=SplinePunkte-1;a++) //Alle Schnitte von Oben nach Unten zählen z.B.: max 18-0
   {
     Sum_Angle_Weight=Sum_Angle_Weight+Angle_Weight[a];
     temp_pos=temp_pos+System_Axis_Points[a]*Angle_Weight[a];
   }
   Sum_Angle_Weight_Balance_Point=temp_pos/Sum_Angle_Weight;
   //println("Sum_Angle_Weight: ", Sum_Angle_Weight);
   //println("Sum_Angle_Weight_Balance_Point: ", Sum_Angle_Weight_Balance_Point);
   //Auflagerkräfte berechnen;
   //Auflagerkraft B=Summe aller Momente um A
   //Achtung nachstehender teil muss mit folgenden 2 Variablen kürzer programmiert werden...
   var Dist_ALL,Dist_RES;
   var Support_Angle_Weight_A,Support_Angle_Weight_B;
   var P1,P2;
   P1=System_Axis_Points[SplinePunkte-1];
   P2=System_Axis_Points[0];
   XZ_Dist_2_Points_Calc(P1,P2);
   Dist_ALL=Dist_current_Points;
   P1=System_Axis_Points[SplinePunkte-1];
   P2=Sum_Angle_Weight_Balance_Point;
   XZ_Dist_2_Points_Calc(P1,P2);
   Dist_RES=Dist_current_Points;
   Support_Angle_Weight_B=(Sum_Angle_Weight*Dist_RES)/Dist_ALL;
   Support_Angle_Weight_A=Sum_Angle_Weight-Support_Angle_Weight_B;
   //println("Support_Angle_Weight_A: ",Support_Angle_Weight_A);
   //println("Support_Angle_Weight_B: ",Support_Angle_Weight_B);
   var Armlenght=0.0;
   var Section_Angle_Weight_Pos=vector(0,0,0);
   var Section_Angle_Weight=new(array,SplinePunkte);
   var X_Dist,Y_Dist,Z_Dist,Res_Dist;
   var Res_Dist,Dist_Sup;
   var temp_Bending;
   //MOMENTENVERLAUFBERECHNEN unter Berücksichtigung des entgegenwirkenden Auflagermomentes be
   for(a=SplinePunkte-2;a>=0;a--) //Alle Schnitte von Oben nach Unten zählen z.B.: max 19-0
   { //Summe aller Winkelgewichte und Duschnittsposition
     temp_Bending=0.0;
     //Summe aller Kräfte mal Abstand //Winkelgewicht x Abstand zum Schnitt
     for(c=a+1;c<=SplinePunkte-1;c++) //Alle Schnitte von Oben nach Unten zählen z.B.: max 18-0
     {
       P1=System_Axis_Points[a];
       P2=System_Axis_Points[c];
       XZ_Dist_2_Points_Calc(P1,P2);
       Res_Dist=Dist_current_Points;
       temp_Bending=temp_Bending+Res_Dist*Angle_Weight[c]; //Kraft mal Kraftarm
     }
     P1=System_Axis_Points[a];
     P2=System_Axis_Points[SplinePunkte-1];
     XZ_Dist_2_Points_Calc(P1,P2);
     Dist_Sup=Dist_current_Points;
     //println("temp_Bending: ",temp_Bending," - Support_Angle_Weight_B*Dist_Sup: ",Support_Angle_Weight
     Bending_Section[a]=abs(temp_Bending)-abs(Support_Angle_Weight_B*Dist_Sup);
   }
   Bending_Section[SplinePunkte-1]=0.0;
   Bending_Section[0]=0.0;
   //Bending Werte umdrehen -> da gepiegeltes System;
   var temp_Bending=new(array,SplinePunkte);
   for(a=0;a<=SplinePunkte-1;a++)
   {
     temp_Bending[a]=Bending_Section[a];
   }
   for(a=0;a<=SplinePunkte-1;a++)
   {
     Bending_Section[a]=abs(temp_Bending[SplinePunkte-1-a]);
     //println("Bending_Section[",a,"]",Bending_Section[a]);
   }
 } // if(Torque_Calc_Mode==2)  ENDE

 //-------------------------------------------------------------------------------------------------------
 /*
 for(a=0;a<SplinePunkte;a++)
 {
   println("-------------------------------------------------------------------------------------------");
   println("Das Schnittmoment am Schnitt[",a,"] beträgt",Section_Torque_X[a]);
   println("Flächenträgheitmoment_Y[",a,"]= ",Section_Moments_of_Inertia_Y[a]," cm^4.");
   //println("Das Winkelgewicht an Schnitt[",a,"] beträgt",Angle_Weight[a]);
   //if(a<SplinePunkte-1) println("Segmentlänge Schnitt[",a,"]:", Bridge_Segment_Lenght[a]);
   println("Durchbiegung Schnitt[",a,"] : ",Bending_Section[a],"cm.");
 }
 println(" ");
 */
 //Je nach Auflagerung jeweilige Durchbiegung zuweißen.
 var Current_Bending_Arm_1;
 var Current_Bending_Arm_2;

 if(Torque_Calc_Mode==0)  //2 Kragarme vom Ufer aus -> Größte Durchbiegung ist am einestelleten Auflager (Spitze)
 {
   if(Support_Position!=1 && Support_Position!=SplinePunkte)
   {
     Current_Bending_Arm_1=Bending_Section[Support_Position-1];
     Current_Bending_Arm_2=Bending_Section[Support_Position];
   }
   if(Support_Position==1)
   {
     Current_Bending_Arm_1=0;
     Current_Bending_Arm_2=Bending_Section[Support_Position-1];
   }
   if(Support_Position==SplinePunkte)
   {
     Current_Bending_Arm_1=Bending_Section[Support_Position-1];
     Current_Bending_Arm_2=0;
   }
 }

 if(Torque_Calc_Mode==1)  //2 Kragarme vom Auflager -> Größte Durchbiegung an den Ufern (Spitzen)
 {
   if(Support_Position!=1 && Support_Position!=SplinePunkte)
   {
     Current_Bending_Arm_1=Bending_Section[0];
     Current_Bending_Arm_2=Bending_Section[SplinePunkte-1];
   }
   if(Support_Position==1)
   {
     Current_Bending_Arm_1=0;
     Current_Bending_Arm_2=Bending_Section[SplinePunkte-1];
   }
   if(Support_Position==SplinePunkte)
   {
     Current_Bending_Arm_1=Bending_Section[0];
     Current_Bending_Arm_2=0;
   }
 }

 if(Torque_Calc_Mode==2)  //Einfeldträger
 {
   var Max_Bending=0.0;
   for(a=0;a<=SplinePunkte-1;a++)
   {
     if(Bending_Section[a]>=abs(Max_Bending)) Max_Bending=Bending_Section[a];
   }
   Current_Bending_Arm_1=Max_Bending;
   Current_Bending_Arm_2=Max_Bending;
 }

if(Current_Bending_Arm_1!=0)
 {
   Bending_of_Arm_1_Efficiency=1/(Max_Bending_of_Arm_1/Current_Bending_Arm_1);
```

```
  } else Bending_of_Arm_1_Efficiency=0;

  if(Current_Bending_Arm_2!=0)
  {
    Bending_of_Arm_2_Efficiency=1/(Max_Bending_of_Arm_2/Current_Bending_Arm_2);
  } else Bending_of_Arm_2_Efficiency=0;
  var MaxBending,temp_Bending;

  if(Bending_of_Arm_1_Efficiency>Bending_of_Arm_2_Efficiency)
  {
    MaxBending=Bending_of_Arm_1_Efficiency;
  }
  else MaxBending=Bending_of_Arm_2_Efficiency;

  var Current_Max_Bending=0;
  Section_Bending_Efficiency=new(array,SplinePunkte);
  for(a=0;a<SplinePunkte;a++) //Alle Schnitte von Oben nach Unten zählen z.B.: max 19-0
  {
    if(Bending_Section[a]>=Current_Max_Bending) Current_Max_Bending=Bending_Section[a];
  }
  for(a=0;a<SplinePunkte;a++) //Alle Schnitte von Oben nach Unten zählen z.B.: max 19-0
  {
    Section_Bending_Efficiency[a]=(Bending_Section[a]/Current_Max_Bending);
    //println("Section_Bending_Efficiency[",a,"]",Section_Bending_Efficiency[a]);
    //println("Bending_Section[",a,"]",Bending_Section[a]);
  }

  println("Kragarm 1 Länge (Brückenlänge)= ",Cantilever_Arm_1," cm.");
  println("Kragarm 2 Länge (Brückenlänge)= ",Cantilever_Arm_2," cm.");
  println("Träger 1 darf sich max. um ",Max_Bending_of_Arm_1," cm durchbiegen!");
  println("Träger 2 darf sich max. um ",Max_Bending_of_Arm_2," cm durchbiegen!");
  println("Träger 1 biegt sich ",Current_Bending_Arm_1,"cm durch!");
  println("Träger 2 biegt sich ",Current_Bending_Arm_2,"cm durch!");
  println("Das E-Modul beträgt: ",E_Modul,"kN/cm². ");
  println("Das durchschnittliche Flächenträgheitsmoment beträgt ",Average_Section_Moments_of_Inertia_Y,"cm
  println("Bending_Arm_1_Efficiency: ",Bending_of_Arm_1_Efficiency*100,"%");
  println("Bending_Arm_2_Efficiency: ",Bending_of_Arm_2_Efficiency*100,"%");
}


//STABILITY CHECK
STABILITY_CHECK()
{
  println(" ");
  println("AUF ZULÄSSIGE SPANNUNGEN ÜBERPRÜFEN");
  var a,cnt;
  var Bal_Point,Check_Point;
  var Check_Distance_Y,Check_Distance_XZ,Smallest_Distance_Y,Smallest_Distance_XZ,Biggest_Check_Dista
  var Check_Distance_X, Smallest_Distance_X,Biggest_Check_Distance_X;
  var Check_Distance_Z, Smallest_Distance_Z,Biggest_Check_Distance_Z;
  var OG_ARM,UG_ARM,Max_Left_ARM,Max_Right_ARM;
  var OG_Point, UG_Point, Max_Left_Point, Max_Right_Point;
  OG_Restiance_Torque=new(array,SplinePunkte);
  UG_Restiance_Torque=new(array,SplinePunkte);
  Left_Side_Restiance_Torque=new(array,SplinePunkte);
  Right_Side_Restiance_Torque=new(array,SplinePunkte);
  var prefix_Z=1;
  var prefix_X=1;
  var Sigma_Zul_Intern=Sigma_Zul;
  OG_Efficiency=new(array,SplinePunkte);
  UG_Efficiency=new(array,SplinePunkte);
  Left_Efficiency=new(array,SplinePunkte);
  Right_Efficiency=new(array,SplinePunkte);

  //----------------------SCHNITTE FÜR HEBELARME CHECKEN----------------------------------
  for (a=0;a<SplinePunkte;a++) //Einzelne Schnitte überprüfen
  {
    //Hebelarmeberechnen vom Schwerpunkt zu OBER- und UNTERGURT (sowie auch Links und Rechts) bere
    Smallest_Distance_Y=0.0;
    Smallest_Distance_XZ=0.0;
    OG_ARM=0.0;
    UG_ARM=0.0;
    Max_Left_ARM=0.0;
    Max_Right_ARM=0.0;
    OG_Point=0;
    UG_Point=0;
    Max_Left_Point=0;
    Max_Right_Point=0;

    for (cnt=0;cnt<Anzahl_Splines*2;cnt++)
    {
      Check_Distance_Y=Section_Area_Balance_Point[a].y-BRIDGE_VOLUME_POINTS_ARRAY[cnt][a].y;
      if(Check_Distance_Y>=OG_ARM)
      {
        OG_ARM=Check_Distance_Y;
        OG_Point=cnt;
      }
      if(Check_Distance_Y<UG_ARM)
      {
        UG_ARM=Check_Distance_Y;
        UG_Point=cnt;
      }
      //Vorzeichen feststellen
      Check_Distance_Z=Section_Area_Balance_Point[a].z-BRIDGE_VOLUME_POINTS_ARRAY[cnt][a].z;
      if(Check_Distance_Z>=0) prefix_Z=1;
      if(Check_Distance_Z<0) prefix_Z=-1;
      Check_Distance_X=Section_Area_Balance_Point[a].x-BRIDGE_VOLUME_POINTS_ARRAY[cnt][a].x;
      if(Check_Distance_X>=0) prefix_X=1;
      if(Check_Distance_X<0) prefix_X=-1;
      Bal_Point=Section_Area_Balance_Point[a];
      Check_Point=BRIDGE_VOLUME_POINTS_ARRAY[cnt][a];
      XZ_Dist_2_Points_Calc(Check_Point,Bal_Point);
      Check_Distance_XZ=Dist_current_Points;
      //println("Check_Distance_XZ: ",Check_Distance_XZ);
      if(prefix_Z<0 && prefix_X<0) Check_Distance_XZ=-Check_Distance_XZ;
      if(prefix_Z<0 && prefix_X>0) Check_Distance_XZ=-Check_Distance_XZ;
      if(prefix_Z>0 && prefix_X<0) Check_Distance_XZ=Check_Distance_XZ;
      if(prefix_Z>0 && prefix_X>0) Check_Distance_XZ=Check_Distance_XZ;
      //Links
      if(Check_Distance_XZ<=0)
      {
        if(Check_Distance_XZ<=Max_Left_ARM)
        {
          Max_Left_ARM=Check_Distance_XZ;
          Max_Left_Point=cnt;
        }
      }
      //Rechts
      if(Check_Distance_XZ>0)
      {
        if(Check_Distance_XZ>Max_Right_ARM)
        {
          Max_Right_ARM=Check_Distance_XZ;
          Max_Right_Point=cnt;
        }
      }
    }
    /*
    println("SchnittNr: ", a);
    println("OG Punkt ",OG_Point," hat einen Hebelarm von ",OG_ARM,"cm.");
    println("UG Punkt ",UG_Point," hat einen Hebelarm von ",UG_ARM,"cm.");
    println("Der linkeste Punkt ",Max_Left_Point," hat einen Hebelarm von ",Max_Left_ARM,"cm.");
    println("Der rechteste Punkt ",Max_Right_Point," hat einen Hebelarm von ",Max_Right_ARM,"cm.");
    */
    //-------------------------------------------------------------------------------------------------------------------------
    if(Material_Typ==0) //STAHL
    {
      //println("Der Stabilitätsnachweis für Stahlbrücke wird berechnet");
      //1.:W=I/z;
      //2.:SIGMA_ZUL=M/W;
      //Wiederstandsmomente aus Trägheitsmomente berechnen.
      if(OG_ARM==0) OG_ARM=0.000000001;
      if(UG_ARM==0) UG_ARM=0.000000001;
      if(Max_Left_ARM==0) Max_Left_ARM=0.000000001;
      if(Max_Right_ARM==0) Max_Right_ARM=0.000000001;
      OG_Restiance_Torque[a]=Section_Moments_of_Inertia_Y[a]/OG_ARM;
      UG_Restiance_Torque[a]=Section_Moments_of_Inertia_Y[a]/UG_ARM;
      if(Section_Torque_X[a]==0) Section_Torque_X[a]=0.000000001;
      if(Section_Torque_Z[a]==0) Section_Torque_Z[a]=0.000000001;
      /*
      println("-------------------------------------------------------------------------------");
      println("SchnittNr: ", a);
      println("Flächenträgheitsmoment um Y: ",Section_Moments_of_Inertia_Y[a],"cm^4.");
      println("Flächenträgheitsmoment um XZ: ",Section_Moments_of_Inertia_XZ[a],"cm^4.");
      println("OG-Hebelarm: ",OG_ARM,"cm.");
      println("UG-Hebelarm: ",UG_ARM,"cm.");
      println("OG_Widerstandsmoment: ",OG_Restiance_Torque[a],"cm³");
      println("UG_Widerstandsmoment: ",UG_Restiance_Torque[a],"cm³");
      println("MOMENT um  X: ",Section_Torque_X[a],"kNcm");
      println("MOMENT um ZX: ",Section_Torque_Z[a],"kNcm");
      */

      //Aktuelles Sigma ausrechen!
      //2.:SIGMA_ZUL=M/W;
      Sigma_OG_Actual=(Global_Safty_Factor*Section_Torque_X[a]/OG_Restiance_Torque[a]);
      Sigma_UG_Actual=(Global_Safty_Factor*Section_Torque_X[a]/UG_Restiance_Torque[a]);
      /*
      println("Sigma_Zul: ",Sigma_Zul," kN/cm²");
```

```
    println("Sigma_OG_Actual: ",Sigma_OG_Actual,"kN/cm²");
    println("Sigma_UG_Actual: ",Sigma_UG_Actual,"kN/cm²");
    println("Sigma_Left_Actual: ",Sigma_Left_Actual,"kN/cm²");
    println("Sigma_Right_Actual: ",Sigma_Right_Actual,"kN/cm²");
    */
    //Aktuelles Sigma mit Sigma zulässig vergleichen!
    //SigmaZul wird in kN/cm² angegeben.
    //OG_Efficiency,UG_Efficiency
    //Auslastung berechnen
    //max.Sigma / zul.Sigma
    OG_Efficiency[a]=abs(Sigma_Zul_Intern/100*Sigma_OG_Actual);
    UG_Efficiency[a]=abs(Sigma_Zul_Intern/100*Sigma_UG_Actual);
    //println("[",a,"] Obergurt ist zu ",OG_Efficiency[a]*100,"% ausgelastet!");
    //println("[",a,"] Untergurt ist zu ",UG_Efficiency[a]*100,"% ausgelastet!");
    //Einzelne Profile sehr überschlägig auf Torsion bewerten.
  }
  else   //Stahl_Beton
  {
    //Durchscnittshöhenwerte der oberen Negativvolumspunkte berechnen -> für Hebelarm -> Stahlspannstan
    //Durchnschittshöhenwere der Positiven Unteren Punkte -> für Hebelarm -> Betonpressung
    //Kd-Verfahren
    var i;
    var d_CONCRETE_ARM,STEEL_ARM;
    var Width_Calc; //z.B.: 50% der Brückenbreite
    var kd_Value,ks_Value,Chart_Row;
    var kd_Chart=new(array,37);
    var ks_Chart=new(array,37);
    var As_Erf;
    var Error=False;

    if(OG_ARM==0) OG_ARM=0.000000001;
    if(UG_ARM==0) UG_ARM=0.000000001;
    if(Max_Left_ARM==0) Max_Left_ARM=0.000000001;
    if(Max_Right_ARM==0) Max_Right_ARM=0.000000001;
    if(Section_Torque_X[a]==0) Section_Torque_X[a]=0.000000001;
    if(Section_Torque_Z[a]==0) Section_Torque_Z[a]=0.000000001;
    d_CONCRETE_ARM=abs(OG_ARM+UG_ARM)*0.9; //cm Achtung: Wird später in m eingesetzt
    STEEL_ARM=abs(OG_ARM)+abs(UG_ARM)-5;
    Width_Calc=(abs(Max_Left_ARM)+abs(Max_Right_ARM))*0.5; //z.B.: 50% der Brückenbreite
    kd_Value=(d_CONCRETE_ARM/100)/(sqrt((Global_Safty_Factor*Section_Torque_X[a]/100)/Width_Calc));
    println("kd_Value= ",kd_Value);
    //kd_Tabelle für B50
    kd_Chart[0]=3.65; kd_Chart[1]=3.16; kd_Chart[2]=2.83; kd_Chart[3]=2.58;
    kd_Chart[4]=2.39; kd_Chart[5]=2.24; kd_Chart[6]=2.11; kd_Chart[7]=2.00;
    kd_Chart[8]=1.91; kd_Chart[9]=1.83; kd_Chart[10]=1.75; kd_Chart[11]=1.69;
    kd_Chart[12]=1.63; kd_Chart[13]=1.58; kd_Chart[14]=1.53; kd_Chart[15]=1.49;
    kd_Chart[16]=1.45; kd_Chart[17]=1.41; kd_Chart[18]=1.38; kd_Chart[19]=1.35;
    kd_Chart[20]=1.32; kd_Chart[21]=1.29; kd_Chart[22]=1.29; kd_Chart[23]=1.26;
    kd_Chart[24]=1.24 ; kd_Chart[25]=1.22; kd_Chart[26]=1.20; kd_Chart[27]=1.17;
    kd_Chart[28]=1.16; kd_Chart[29]=1.15; kd_Chart[30]=1.14; kd_Chart[31]=1.12;
    kd_Chart[32]=1.10; kd_Chart[33]=1.08; kd_Chart[34]=1.07; kd_Chart[35]=1.05;
    kd_Chart[36]=1.05;

    for(i=0;i<36;i++)
    {
      if(kd_Value>kd_Chart[0])
      {
        Error=True;
        Chart_Row=0;
        break;
      }
      if(kd_Chart[i]>= kd_Value >=kd_Chart[i+1])
      {
        Chart_Row=i;
      }
      if(kd_Value<kd_Chart[36])
      {
        Error=True;
        Chart_Row=36;
        break;
      }
    }

    if(Error==True)
    {
      if(Chart_Row==0)
      {
        println("System versagt!!!!");
        OG_Efficiency[a]=1.7;
        UG_Efficiency[a]=1.7;
      }
      if(Chart_Row==36)
      {
        println("System ist überdimensioniert!!!");
        OG_Efficiency[a]=0.7;
        UG_Efficiency[a]=0.7;
      }
    }
    println("Tabellen_Zeile: ",Chart_Row);
    //In ks_Tabelle Wert nachschlagen
    ks_Chart[0]=47.1; ks_Chart[1]=46.8; ks_Chart[2]=46.5; ks_Chart[3]=46.3;
    ks_Chart[4]=46.0; ks_Chart[5]=45.7; ks_Chart[6]=45.5; ks_Chart[7]=45.2;
    ks_Chart[8]=44.9; ks_Chart[9]=44.6; ks_Chart[10]=44.4; ks_Chart[11]=44.1;
    ks_Chart[12]=43.8; ks_Chart[13]=43.5; ks_Chart[14]=43.2; ks_Chart[15]=42.9;
    ks_Chart[16]=42.6; ks_Chart[17]=42.2; ks_Chart[18]=41.9; ks_Chart[19]=41.6;
    ks_Chart[20]=41.3; ks_Chart[21]=40.9; ks_Chart[22]=40.8; ks_Chart[23]=40.6;
    ks_Chart[24]=40.2; ks_Chart[25]=39.8; ks_Chart[26]=39.5; ks_Chart[27]=39.1;
    ks_Chart[28]=38.9; ks_Chart[29]=38.7; ks_Chart[30]=38.3; ks_Chart[31]=37.9;
    ks_Chart[32]=37.5; ks_Chart[33]=37.0; ks_Chart[34]=36.6; ks_Chart[35]=36.1;
    ks_Chart[36]=36.0;
    //Erforderliche Stahlfläche berechnen
    ks_Value=ks_Chart[Chart_Row];
    println("ks_Value: ",ks_Value);
    if(d_CONCRETE_ARM==0.0)
    {
      println("Beton Hebelarm ist sehr klein!");
      d_CONCRETE_ARM=0.00001;
    }
    As_Erf=(Global_Safty_Factor*Section_Torque_X[a]/100)/(ks_Value*(d_CONCRETE_ARM/100));
    if(Error==False)
    {
      println("Die erforderliche Stahlfläche beträgt: ",As_Erf);
    }
    else println("Die erforderliche Stahlfläche mit: ",As_Erf," ist falsch, da kd-Wert bereits falsch berechnet wurde!");
  }
 }
}

//Die Kontrollpunkte der Weg verzerren
MODIFY_PATHS()
{
  println(" ");
  println("Die Wegeführung wird verändert!");
  var a,cnt,i,c;
  var Section_Percent_Step,Percent_Position=0.0;
  var Path_Point_Crd,Path_1_Object,Path_2_Object,Path_3_Object;
  var Path_Width, Path_1_Full_Lenght, Path_2_Full_Lenght, Path_3_Full_Lenght;
  var Path_1_Control_Points,Path_2_Control_Points,Path_3_Control_Points;
  var current_Point_Crd,NearestSection_Num=0;
  var Path_to_Point_Distance,temp_Distance,Smallest_Distance;
  var Change_Faktor_1;
  var X_Distanz_to_Section_Balance_Point,Y_Distanz_to_Section_Balance_Point, Z_Distanz_to_Section_Balance_Point;
  var Y_Distanz_to_Water;
  var New_Path_1_Point_Crd,New_Path_2_Point_Crd,New_Path_3_Point_Crd;
  var P1,P2;
  var Path_Ideal_Lenght,Near_Distance,Point_Distance_Path1_Path2,Point_Distance_Path1_Path3,Point_Distance_Path2_Path3;
  var Path_1_is_Near=False,Path_2_is_Near=False,Path_3_is_Near=False;
  var Path_1_Near_Points_FLAG,Path_2_Near_Points_FLAG,Path_3_Near_Points_FLAG;
  var Path_1_Point_Crd=vector(0,0,0),Path_2_Point_Crd=vector(0,0,0),Path_3_Point_Crd=vector(0,0,0);
  var Deepest_Point,Deepest_Point_Num;
  var Waterlevel_Already_Checked=False; //-> Damit die Wege nur einmal nach oben verschoben werden müssen
  var Max_Torque_Z,Max_Torque_X,Temp_Torque;
  var Change_Paths=0;
  Path_to_Point_Distance=100000000;
  Smallest_Distance=100000000;
  var Zufall, Zufallszahl_1,Zufallszahl_2,Zufallszahl_3;
  Zufall=new(Random);
  Zufall->Init(time());
  var BridgeWidth=Pedestrian_Width+Biker_Width+WheelChair_Width;
   //Verhältnis zwischen Momenten berechnen.
   Max_Torque_Z=Section_Torque_Z[0];
   for (a=1;a<SplinePunkte-1;a++) //PunktNr
   {
    Temp_Torque=Section_Torque_Z[a];
    if(Temp_Torque>=Max_Torque_Z) Max_Torque_Z=Temp_Torque;
   }
   Max_Torque_X=Section_Torque_X[0];
   for (a=1;a<SplinePunkte-1;a++) //PunktNr
   {
    Temp_Torque=Section_Torque_X[a];
    if(Temp_Torque>=Max_Torque_X) Max_Torque_X=Temp_Torque;
   }
   //if(Max_Torque_Z>=Max_Torque_X/10) Change_Direction=True;
   if(Max_Torque_Z<=(Max_Torque_X/4)) Change_Paths=0; //Pfade nicht verändern
   if(Max_Torque_Z>(Max_Torque_X/4)) Change_Paths=1; //Pfade weg von der Ideallinie bewegen
   if(Max_Torque_Z>(Max_Torque_X/2)) Change_Paths=2; //Pfade zur Ideallinie bewegen
   println("Max_Torque_Z: ",Max_Torque_Z," Max_Torque_X: ",Max_Torque_X);
   //Welcher SplinePunkt ist am Nähesten zum Kontroll_Punkt -> Mit dem Resultat kann der Schnitt an der Stelle beurteilt werden.
```

```
//-------------------------------------------------------------------------------------------------------------
if(doc->FindObject("Pedestrian_Path_Spline"))
{
  Path_1_Object=doc->FindObject("Pedestrian_Path_Spline");
  Path_1_Control_Points=Path_1_Object->GetPointCount();
  println("Pedestrian_Path_Spline gefunden mit ",Path_1_Control_Points," Controlpoints gefunden!!!");
  //Welcher Schnitt ist am nähersten zum aktuellen Path_Control_Point?
  for(i=1;i<Path_1_Control_Points-1;i++)
  {
    println("Controlpoint_Num: ",i);
    Path_Point_Crd=Path_1_Object->GetPoint(i);
    for (a=1;a<SplinePunkte-1;a++) //PunktNr
    {
      current_Point_Crd=Section_Area_Balance_Point[a];
      XZ_Dist_2_Points_Calc(current_Point_Crd,Path_Point_Crd);
      Path_to_Point_Distance=abs(Dist_current_Points);
      if(Path_to_Point_Distance<=Smallest_Distance)
      {
        NearestSection_Num=a;
        Smallest_Distance=Path_to_Point_Distance;
        //println("NearestSection_Num: ",NearestSection_Num," und Smallest_Distance: ",Smallest_Distance);
      }
    }
    New_Path_1_Point_Crd=Path_Point_Crd;
    current_Point_Crd=Section_Area_Balance_Point[NearestSection_Num];
    P1=Path_1_Object->GetPoint(0);
    P2=Path_1_Object->GetPoint(Path_1_Control_Points-1);
    XYZ_Dist_2_Points_Calc(P1,P2);
    Path_Ideal_Lenght=abs(Dist_current_Points);
    Near_Distance=Path_Ideal_Lenght/Path_1_Control_Points;
    //-----------------------------------------------------------------------
    //Liegt ein oder meherer anderere Pfadkontrollpunkt in unmittelbarer Nähe?
    if(doc->FindObject("Biker_Path_Spline"))
    {
      Path_2_Object=doc->FindObject("Biker_Path_Spline");
      Path_2_Control_Points=Path_2_Object->GetPointCount();
      Path_2_Near_Points_FLAG=new(array,Path_2_Control_Points);
      //Welcher der Path_2_Punkte liegt näher als ... an New_Path_1_Point_Crd
      for(a=1;a<Path_2_Control_Points-1;a++)
      {
        Path_Point_Crd=Path_2_Object->GetPoint(a);
        XYZ_Dist_2_Points_Calc(New_Path_1_Point_Crd,Path_Point_Crd);
        Point_Distance_Path1_Path2=abs(Dist_current_Points);
        Path_2_Near_Points_FLAG[a]=False;
        if(Point_Distance_Path1_Path2<Near_Distance)
        {
          Path_2_is_Near=True;
          Path_2_Near_Points_FLAG[a]==True;
        }
      }
    }
    if(doc->FindObject("Barrier_Free_Path_Spline"))
    {
      Path_3_Object=doc->FindObject("Barrier_Free_Path_Spline");
      Path_3_Control_Points=Path_3_Object->GetPointCount();
      Path_3_Near_Points_FLAG=new(array,Path_3_Control_Points);
      for(a=1;a<Path_3_Control_Points-1;a++)
      {
        Path_Point_Crd=Path_3_Object->GetPoint(a);
        XYZ_Dist_2_Points_Calc(New_Path_1_Point_Crd,Path_Point_Crd);
        Point_Distance_Path1_Path3=abs(Dist_current_Points);
        Path_3_Near_Points_FLAG[a]=False;
        if(Point_Distance_Path1_Path3<Near_Distance)
        {
          Path_3_is_Near=True;
          Path_3_Near_Points_FLAG[a]==True;
        }
      }
    }
    //-----------------------------------------------------------------
    //Nun steht fest ob und welche anderen Pfadkontrollpunkte sich in der Nähe befinden
    //WASSERSTAND BERÜCKSICHTIGEN: Abfragen und eventuell Veränderungen vornehmen
    var current_Point;
    Deepest_Point=BRIDGE_VOLUME_POINTS_ARRAY[0][NearestSection_Num];
    Deepest_Point_Num=0;
    for(cnt=0;cnt<Anzahl_Splines*2;cnt++)
    {
      current_Point=BRIDGE_VOLUME_POINTS_ARRAY[cnt][NearestSection_Num];
      if(current_Point.y<=Deepest_Point.y)
      {
        Deepest_Point=BRIDGE_VOLUME_POINTS_ARRAY[cnt][NearestSection_Num];
        Deepest_Point_Num=Deepest_Point_Num+1;
        //println("Der Tiefste Punkt an Schnitt ",NearestSection_Num," ist Punkt[",cnt,"|",NearestSection_Num,"."
      }
    }
    println("Waterlevel_Point: ",Waterlevel_Point);
    if(Deepest_Point.y<=Waterlevel_Point.y+Minimal_Construction_Height)
    {
      //Waterlevel_Already_Checked=True;
      //An aktuellem Schnitt den tiefsten Kontruktionspunkt ermitteln (Der Wegpunkt ist vielleicht nicht im wasser...aber di
      //NearestSection_Num enthält den nähesten Schnitt
      Y_Distanz_to_Water=(Waterlevel_Point.y+Minimal_Construction_Height)-Deepest_Point.y;
      println("Der Punkt",Deepest_Point," ist im Wasser, das sind ",Y_Distanz_to_Water,"cm zuviel");
      println("Punkt ",i," der Pedestrian_Path_Spline wird verschoben");
      //Nun müssen alle Weg gleichmäßig nach oben verschoben werden
      New_Path_1_Point_Crd.y=(Waterlevel_Point.y+Minimal_Construction_Height)+Y_Distanz_to_Water;
      if(doc->FindObject("Biker_Path_Spline"))
      { //Alle nahen Path_2_Control_Punkte nach oben verschieben.
        for(c=1;c<Path_2_Control_Points-1;c++)
        {
          if(Path_2_Near_Points_FLAG[c]==True)
          {
            Path_2_Point_Crd=Path_2_Object->GetPoint(c);
            Path_2_Point_Crd.y= Path_2_Point_Crd.y+Y_Distanz_to_Water;
            Path_2_Object->SetPoint(c);
            println("Punkt ",c," der Biker Path Spline wird verschoben");
          }
        }
        Path_2_Object->Message(MSG_UPDATE);
      } //if(doc->FindObject("Biker_Path_Spline")) ENDE
      if(doc->FindObject("Barrier_Free_Path_Spline"))
      { //Alle nahen Path_3_Control_Punkte nach oben verschieben.
        for(c=1;c<Path_3_Control_Points-1;c++)
        {
          if(Path_3_Near_Points_FLAG[c]==True)
          {
            Path_3_Point_Crd=Path_3_Object->GetPoint(c);
            Path_3_Point_Crd.y= Path_3_Point_Crd.y+Y_Distanz_to_Water;
            Path_3_Object->SetPoint(c);
            println("Punkt ",c," der Barrier_Free_Path_Spline wird verschoben");
          }
        }
        Path_3_Object->Message(MSG_UPDATE);
      } //if(doc->FindObject("Biker_Path_Spline")) ENDE
    }//WASSERSTNAD BERÜCKSICHTIGEN ENDE
    else println("Der Punkt",Deepest_Point," ist nicht im Wasser!");
    Path_1_Object->SetPoint(i,New_Path_1_Point_Crd);
    Path_1_Object->Message(MSG_UPDATE);
    //BRÜCKEN ACHSLINIENFÜHRUNG VERÄNDERN -> Um Platz für Querschnittveränderungen nach oben zu schaffen
verringern.
    if(Change_Bridge_Symmetry==True)
    {
      var Ideal_Line_Start_Point,Ideal_Line_End_Point, New_Ideal_Endpoint,Ideal_Spline_Num,Degree_Ideal_Line,Normal
      var Path_Point_New_Temp,Count,Degree_Path_Point;
      var Prefix;
      //1 ZWEI ENDPUNTE DER IDEALSPLINE FESTELGEN
      Ideal_Spline_Num=int(Anzahl_Splines/2);
      Ideal_Line_Start_Point=BRIDGE_VOLUME_POINTS_ARRAY[Ideal_Spline_Num][0];
      Ideal_Line_End_Point=BRIDGE_VOLUME_POINTS_ARRAY[Ideal_Spline_Num][SplinePunkte-1];
      New_Ideal_Endpoint=Ideal_Line_End_Point;
      New_Ideal_Endpoint.x=Ideal_Line_End_Point.x-Ideal_Line_Start_Point.x;
      New_Ideal_Endpoint.z=Ideal_Line_End_Point.z-Ideal_Line_Start_Point.z;
      Degree_Ideal_Line=New_Ideal_Endpoint.x/New_Ideal_Endpoint.z;
      Normal_Degree_Ideal_Line=New_Ideal_Endpoint.z/New_Ideal_Endpoint.x;
      XZ_Dist_2_Points_Calc(Ideal_Line_Start_Point,Ideal_Line_End_Point);
      Ideal_Line_Lenght=Dist_current_Points;
      Path_Point_Crd=Path_1_Object->GetPoint(i);
      Path_Point_New_Temp=Path_Point_Crd;
      Path_Point_New_Temp.x=Path_Point_Crd.x-Ideal_Line_Start_Point.x;
      Path_Point_New_Temp.z=Path_Point_Crd.z-Ideal_Line_Start_Point.z;
      Degree_Path_Point=Path_Point_New_Temp.x/Path_Point_New_Temp.z;
      println("-----------------------------------------------------------------------");
      println("Ideal_Line_Start_Point: ", Ideal_Line_Start_Point);
      println("New_Ideal_Endpoint: ",New_Ideal_Endpoint);
      println("Ideal_Line_Lenght: ", Ideal_Line_Lenght);
      println("Degree_Ideal_Line: ",Degree_Ideal_Line);
      println("Normal_Degree_Ideal_Line: ",Normal_Degree_Ideal_Line);
      println("Path_Point_New_Temp: ", Path_Point_New_Temp);
      println("Degree_Path_Point: ",Degree_Path_Point);
      println("Degree_Ideal_Line ",Degree_Ideal_Line,"> Degree_Path_Point",Degree_Path_Point);
      if(Degree_Path_Point<0)
      {
        if(Change_Paths==1)
        {
          if(abs(Degree_Ideal_Line)>abs(Degree_Path_Point)) Prefix=1;
          if(abs(Degree_Ideal_Line)<=abs(Degree_Path_Point)) Prefix=-1;
        }
        if(Change_Paths==2)
```

```
        {
          if(abs(Degree_Ideal_Line)>abs(Degree_Path_Point)) Prefix=-1;
          if(abs(Degree_Ideal_Line)<=abs(Degree_Path_Point)) Prefix=1;
        }
      }
      else
      {
        if(Change_Paths==1)
        {
          if(abs(Degree_Ideal_Line)>abs(Degree_Path_Point)) Prefix=-1;
          if(abs(Degree_Ideal_Line)<=abs(Degree_Path_Point)) Prefix=1;
        }
        if(Change_Paths==2)
        {
          if(abs(Degree_Ideal_Line)>abs(Degree_Path_Point)) Prefix=1;
          if(abs(Degree_Ideal_Line)<=abs(Degree_Path_Point)) Prefix=-1;
        }
      }
      if(Change_Paths==0) Prefix=0;
      Zufallszahl_1=Zufall->Get01();
      Zufallszahl_1=Zufallszahl_1+1;
      New_Path_1_Point_Crd=Path_Point_Crd;
      New_Path_1_Point_Crd.x=Path_Point_Crd.x+Prefix*((Pedestrian_Width/4)*Zufallszahl_1)*Normal_Degre
      New_Path_1_Point_Crd.z=Path_Point_Crd.z+Prefix*((Pedestrian_Width/4)*Zufallszahl_1);
    }
    Path_1_Object->SetPoint(i,New_Path_1_Point_Crd);
  }
  Path_1_Object->Message(MSG_UPDATE);
}
//-------------------------------------------------------------------------------------------------------------------
if(doc->FindObject("Biker_Path_Spline"))
{
  Path_2_Object=doc->FindObject("Biker_Path_Spline");
  Path_2_Control_Points=Path_2_Object->GetPointCount();
  println("Biker_Path_Spline gefunden mit ",Path_2_Control_Points," Controlpoints gefunden!!!");
  //Welcher Schnitt ist am nähersten zum aktuellen Path_Control_Point?
  for(i=1;i<Path_2_Control_Points-1;i++)
  {
    println("Controlpoint_Num: ",i);
    Path_Point_Crd=Path_2_Object->GetPoint(i);
    for (a=1;a<SplinePunkte-1;a++) //PunktNr
    {
      current_Point_Crd=Section_Area_Balance_Point[a];
      XZ_Dist_2_Points_Calc(current_Point_Crd,Path_Point_Crd);
      Path_to_Point_Distance=abs(Dist_current_Points);
      if(Path_to_Point_Distance<=Smallest_Distance)
      {
        NearestSection_Num=a;
        Smallest_Distance=Path_to_Point_Distance;
        //println("NearestSection_Num: ",NearestSection_Num," und Smallest_Distance: ",Smallest_Distance);
      }
    }
    New_Path_2_Point_Crd=Path_Point_Crd;
    current_Point_Crd=Section_Area_Balance_Point[NearestSection_Num];
    P1=Path_2_Object->GetPoint(0);
    P2=Path_2_Object->GetPoint(Path_2_Control_Points-1);
    XYZ_Dist_2_Points_Calc(P1,P2);
    Path_Ideal_Lenght=abs(Dist_current_Points);
    Near_Distance=Path_Ideal_Lenght/Path_2_Control_Points;
    //-----------------------------------------------------------------------
    //Liegt ein oder meherer anderere Pfadkontrollpunkt in unmittelbarer Nähe?
    if(doc->FindObject("Pedestrian_Path_Spline"))
    {
      Path_1_Object=doc->FindObject("Pedestrian_Path_Spline");
      Path_1_Control_Points=Path_1_Object->GetPointCount();
      Path_1_Near_Points_FLAG=new(array,Path_1_Control_Points);
      //Welcher der Path_2_Punkte liegt näher als ... an New_Path_1_Point_Crd
      for(a=1;a<Path_1_Control_Points-1;a++)
      {
        Path_Point_Crd=Path_1_Object->GetPoint(a);
        XYZ_Dist_2_Points_Calc(New_Path_2_Point_Crd,Path_Point_Crd);
        Point_Distance_Path1_Path2=abs(Dist_current_Points);
        Path_1_Near_Points_FLAG[a]=False;
        if(Point_Distance_Path1_Path2<Near_Distance)
        {
          Path_1_is_Near=True;
          Path_1_Near_Points_FLAG[a]==True;
        }
      }
    }
    if(doc->FindObject("Barrier_Free_Path_Spline"))
    {
      Path_3_Object=doc->FindObject("Biker_Path_Spline");
      Path_3_Control_Points=Path_3_Object->GetPointCount();
      Path_3_Near_Points_FLAG=new(array,Path_3_Control_Points);
      for(a=1;a<Path_3_Control_Points-1;a++)
      {
        Path_Point_Crd=Path_3_Object->GetPoint(a);
        XYZ_Dist_2_Points_Calc(New_Path_2_Point_Crd,Path_Point_Crd);
        Point_Distance_Path1_Path3=abs(Dist_current_Points);
        Path_3_Near_Points_FLAG[a]=False;
        if(Point_Distance_Path1_Path3<Near_Distance)
        {
          Path_3_is_Near=True;
          Path_3_Near_Points_FLAG[a]==True;
        }
      }
    }
    //-----------------------------------------------------------------------
    //Nun steht fest ob und welche anderen Pfadkontrollpunkte sich in der Nähe befinden
    //WASSERSTAND BERÜCKSICHTIGEN: Abfragen und eventuell Veränderungen vornehmen
    if(Path_2_Near_Points_FLAG[i]==False) //Wenn dieser Punkt bereits vorher verschoben wurde dann keine Überprüfung
    {
      var current_Point;
      Deepest_Point=BRIDGE_VOLUME_POINTS_ARRAY[0][NearestSection_Num];
      Deepest_Point_Num=0;
      for(cnt=0;cnt<Anzahl_Splines*2;cnt++)
      {
        current_Point=BRIDGE_VOLUME_POINTS_ARRAY[cnt][NearestSection_Num];
        if(current_Point.y<=Deepest_Point.y)
        {
          Deepest_Point=BRIDGE_VOLUME_POINTS_ARRAY[cnt][NearestSection_Num];
          Deepest_Point_Num=Deepest_Point_Num+1;
          //println("Der Tiefste Punkt an Schnitt ",NearestSection_Num," ist Punkt[",cnt,"|",NearestSection_Num,"."");
        }
      }
      println("Waterlevel_Point: ",Waterlevel_Point);
      if(Deepest_Point.y<=Waterlevel_Point.y+Minimal_Construction_Height)
      {
        //Waterlevel_Already_Checked=True;
        Y_Distanz_to_Water=(Waterlevel_Point.y+Minimal_Construction_Height)-Deepest_Point.y;
        println("Der Punkt",Deepest_Point," ist im Wasser, das sind ",Y_Distanz_to_Water,"cm zuviel");
        //Nun müssen alle Weg gleichmäßig nach oben verschoben werden
        New_Path_2_Point_Crd.y=(Waterlevel_Point.y+Minimal_Construction_Height)+Y_Distanz_to_Water;
        println("Punkt ",i," der Biker_Path_Spline wird verschoben");
        if(doc->FindObject("Pedestrian_Path_Spline"))
        {
          //Alle nahen Path_2_Control_Punkte nach oben verschieben.
          for(c=1;c<Path_1_Control_Points-1;c++)
          {
            if(Path_1_Near_Points_FLAG[c]==True)
            {
              Path_1_Point_Crd=Path_1_Object->GetPoint(c);
              Path_1_Point_Crd.y= Path_1_Point_Crd.y+Y_Distanz_to_Water;
              Path_1_Object->SetPoint(c);
              println("Punkt ",c," der Pedestrian_Path_Spline wird verschoben");
            }
          }
          Path_1_Object->Message(MSG_UPDATE);
        } //if(doc->FindObject("Biker_Path_Spline")) ENDE
        if(doc->FindObject("Barrier_Free_Path_Spline"))
        {
          //Alle nahen Path_3_Control_Punkte nach oben verschieben.
          for(c=1;c<Path_3_Control_Points-1;c++)
          {
            if(Path_3_Near_Points_FLAG[c]==True)
            {
              Path_3_Point_Crd=Path_3_Object->GetPoint(c);
              Path_3_Point_Crd.y= Path_3_Point_Crd.y+Y_Distanz_to_Water;
              Path_3_Object->SetPoint(c);
              println("Punkt ",c," der Barrier_Free_Path_Spline wird verschoben");
            }
          }
          Path_3_Object->Message(MSG_UPDATE);
        } //if(doc->FindObject("Biker_Path_Spline")) ENDE
      }//WASSERSTNAD BERÜCKSICHTIGEN ENDE
      else println("Der Punkt",Deepest_Point," ist nicht im Wasser!");
    } // if(Waterlevel_Already_Checked==False) ENDE
    else println("Punkt ",i," wird nicht überprüft, da dieser bereits vorher verschoben wurde!");
    Path_2_Object->SetPoint(i,New_Path_1_Point_Crd);
    Path_2_Object->Message(MSG_UPDATE);

    //BRÜCKEN ACHSLINIENFÜHRUNG VERÄNDERN -> Um Platz für Querschnittveränderungen nach oben zu schaffen und um eventu
veringern.
    if(Change_Bridge_Symmetry==True)
    {
      var Ideal_Line_Start_Point,Ideal_Line_End_Point, New_Ideal_Endpoint,Ideal_Spline_Num,Degree_Ideal_Line,Normal_Degree_Ideal
      var Path_Point_New_Temp,Count,Degree_Path_Point;
```

```
    var Prefix;

    //1 ZWEI ENDPUNTE DER IDEALSPLINE FESTELGEN
    Ideal_Spline_Num=int(Anzahl_Splines/2);
    Ideal_Line_Start_Point=BRIDGE_VOLUME_POINTS_ARRAY[Ideal_Spline_Num][0];
    Ideal_Line_End_Point=BRIDGE_VOLUME_POINTS_ARRAY[Ideal_Spline_Num][SplinePunkte-1];
    New_Ideal_Endpoint=Ideal_Line_End_Point;
    New_Ideal_Endpoint.x=Ideal_Line_End_Point.x-Ideal_Line_Start_Point.x;
    New_Ideal_Endpoint.z=Ideal_Line_End_Point.z-Ideal_Line_Start_Point.z;
    Degree_Ideal_Line=New_Ideal_Endpoint.x/New_Ideal_Endpoint.z;
    Normal_Degree_Ideal_Line=New_Ideal_Endpoint.z/New_Ideal_Endpoint.x;
    XZ_Dist_2_Points_Calc(Ideal_Line_Start_Point,Ideal_Line_End_Point);
    Ideal_Line_Lenght=Dist_current_Points;
    Path_Point_Crd=Path_2_Object->GetPoint(i);
    Path_Point_New_Temp=Path_Point_Crd;
    Path_Point_New_Temp.x=Path_Point_Crd.x-Ideal_Line_Start_Point.x;
    Path_Point_New_Temp.z=Path_Point_Crd.z-Ideal_Line_Start_Point.z;
    Degree_Path_Point=Path_Point_New_Temp.x/Path_Point_New_Temp.z;
    if(Degree_Path_Point<0)
    {
     if(Change_Paths==1)
     {
      if(abs(Degree_Ideal_Line)>abs(Degree_Path_Point)) Prefix=1;
      if(abs(Degree_Ideal_Line)<=abs(Degree_Path_Point)) Prefix=-1;
     }
     if(Change_Paths==2)
     {
      if(abs(Degree_Ideal_Line)>abs(Degree_Path_Point)) Prefix=-1;
      if(abs(Degree_Ideal_Line)<=abs(Degree_Path_Point)) Prefix=1;
     }
    }
    else
    {
     if(Change_Paths==1)
     {
      if(abs(Degree_Ideal_Line)>abs(Degree_Path_Point)) Prefix=-1;
      if(abs(Degree_Ideal_Line)<=abs(Degree_Path_Point)) Prefix=1;
     }
     if(Change_Paths==2)
     {
      if(abs(Degree_Ideal_Line)>abs(Degree_Path_Point)) Prefix=1;
      if(abs(Degree_Ideal_Line)<=abs(Degree_Path_Point)) Prefix=-1;
     }
    }
    if(Change_Paths==0) Prefix=0;
    Zufallszahl_2=Zufall->Get01();
    Zufallszahl_2=Zufallszahl_2+1;
    New_Path_2_Point_Crd=Path_Point_Crd;
    New_Path_2_Point_Crd.x=Path_Point_Crd.x+Prefix*((Biker_Width/4)*Zufallszahl_2)*Normal_Degree_Ide
    New_Path_2_Point_Crd.z=Path_Point_Crd.z+Prefix*((Biker_Width/4)*Zufallszahl_2);
   }
   Path_2_Object->SetPoint(i,New_Path_2_Point_Crd);
  }
 Path_2_Object->Message(MSG_UPDATE);
 }
 //-------------------------------------------------------------------------------------------------------
 if(doc->FindObject("Barrier_Free_Path_Spline"))
 {
  Path_3_Object=doc->FindObject("Barrier_Free_Path_Spline");
  Path_3_Control_Points=Path_3_Object->GetPointCount();
  println("Barrier_Free_Path_Spline gefunden mit ",Path_3_Control_Points," Controlpoints gefunden!!!");
  //Welcher Schnitt ist am nähersten zum aktuellen Path_Control_Point?
  for(i=1;i<Path_3_Control_Points-1;i++)
  {
   println("Controlpoint_Num: ",i);
   Path_Point_Crd=Path_3_Object->GetPoint(i);
   for (a=1;a<SplinePunkte-1;a++) //PunktNr
   {
    current_Point_Crd=Section_Area_Balance_Point[a];
    XZ_Dist_2_Points_Calc(current_Point_Crd,Path_Point_Crd);
    Path_to_Point_Distance=abs(Dist_current_Points);
    if(Path_to_Point_Distance<=Smallest_Distance)
    {
     NearestSection_Num=a;
     Smallest_Distance=Path_to_Point_Distance;
     //println("NearestSection_Num: ",NearestSection_Num," und Smallest_Distance: ",Smallest_Distance);
    }
   }
   New_Path_3_Point_Crd=Path_Point_Crd;
   current_Point_Crd=Section_Area_Balance_Point[NearestSection_Num];
   P1=Path_3_Object->GetPoint(0);
   P2=Path_3_Object->GetPoint(Path_3_Control_Points-1);
   XYZ_Dist_2_Points_Calc(P1,P2);
   Path_Ideal_Lenght=abs(Dist_current_Points);
   Near_Distance=Path_Ideal_Lenght/Path_3_Control_Points;
   //-----------------------------------------------------------------
   //Liegt ein oder meherer anderere Pfadkontrollpunkt in unmittelbarer Nähe?
   if(doc->FindObject("Pedestrian_Path_Spline"))
   {
    Path_1_Object=doc->FindObject("Pedestrian_Path_Spline");
    Path_1_Control_Points=Path_1_Object->GetPointCount();
    Path_1_Near_Points_FLAG=new(array,Path_1_Control_Points);
    //Welcher der Path_2_Punkte liegt näher als ... an New_Path_1_Point_Crd
    for(a=1;a<Path_1_Control_Points-1;a++)
    {
     Path_Point_Crd=Path_1_Object->GetPoint(a);
     XYZ_Dist_2_Points_Calc(New_Path_3_Point_Crd,Path_Point_Crd);
     Point_Distance_Path1_Path3=abs(Dist_current_Points);
     Path_1_Near_Points_FLAG[a]=False;
     if(Point_Distance_Path1_Path3<Near_Distance)
     {
      Path_1_is_Near=True;
      Path_1_Near_Points_FLAG[a]==True;
     }
    }
   }
   if(doc->FindObject("Biker_Path_Spline"))
   {
    Path_2_Object=doc->FindObject("Biker_Path_Spline");
    Path_2_Control_Points=Path_2_Object->GetPointCount();
    Path_2_Near_Points_FLAG=new(array,Path_2_Control_Points);
    for(a=1;a<Path_2_Control_Points-1;a++)
    {
     Path_Point_Crd=Path_2_Object->GetPoint(a);
     XYZ_Dist_2_Points_Calc(New_Path_3_Point_Crd,Path_Point_Crd);
     Point_Distance_Path2_Path3=abs(Dist_current_Points);
     Path_2_Near_Points_FLAG[a]=False;
     if(Point_Distance_Path2_Path3<Near_Distance)
     {
      Path_2_is_Near=True;
      Path_2_Near_Points_FLAG[a]==True;
     }
    }
   }
   //-----------------------------------------------------------------
   //Nun steht fest ob und welche anderen Pfadkontrollpunkte sich in der Nähe befinden
   //WASSERSTAND BERÜCKSICHTIGEN: Abfragen und eventuell Veränderungen vornehmen
   if(Path_3_Near_Points_FLAG[i]==False) //Wenn dieser Punkt bereits vorher verschoben wurde dann
   {
    var current_Point;
    Deepest_Point=BRIDGE_VOLUME_POINTS_ARRAY[0][NearestSection_Num];
    Deepest_Point_Num=0;
    for(cnt=0;cnt<Anzahl_Splines*2;cnt++)
    {
     current_Point=BRIDGE_VOLUME_POINTS_ARRAY[cnt][NearestSection_Num];
     if(current_Point.y<=Deepest_Point.y)
     {
      Deepest_Point=BRIDGE_VOLUME_POINTS_ARRAY[cnt][NearestSection_Num];
      Deepest_Point_Num=Deepest_Point_Num+1;
      //println("Der Tiefste Punkt an Schnitt ",NearestSection_Num," ist Punkt[",cnt,"]",NearestSection_
     }
    }
    println("Waterlevel_Point: ",Waterlevel_Point);
    if(Deepest_Point.y<=Waterlevel_Point.y+Minimal_Construction_Height)
    {
     //Waterlevel_Already_Checked=True;
     Y_Distanz_to_Water=(Waterlevel_Point.y+Minimal_Construction_Height)-Deepest_Point.y;
     println("Der Punkt",Deepest_Point," ist im Wasser, das sind ",Y_Distanz_to_Water,"cm zuviel");
     //Nun müssen alle Weg gleichmäßig nach oben verschoben werden
     New_Path_3_Point_Crd.y=(Waterlevel_Point.y+Minimal_Construction_Height)+Y_Distanz_to_Wate
     println("Punkt ",i," der Barrier_Free_Path_Spline wird verschoben");
     if(doc->FindObject("Pedestrian_Path_Spline"))
     {
      //Alle nahen Path_2_Control_Punkte nach oben verschieben.
      for(c=1;c<Path_1_Control_Points-1;c++)
      {
       if(Path_1_Near_Points_FLAG[c]==True)
       {
        Path_1_Point_Crd=Path_1_Object->GetPoint(c);
        Path_1_Point_Crd.y= Path_1_Point_Crd.y+Y_Distanz_to_Water;
        Path_1_Object->SetPoint(c);
        println("Punkt ",c," der Pedestrian_Path_Spline wird verschoben");
       }
      }
      Path_1_Object->Message(MSG_UPDATE);
     } //if(doc->FindObject("Biker_Path_Spline")) ENDE
     if(doc->FindObject("Biker_Path_Spline"))
     { //Alle nahen Path_3_Control_Punkte nach oben verschieben.
```

```
          for(c=1;c<Path_2_Control_Points-1;c++)
          {
            if(Path_2_Near_Points_FLAG[c]==True)
            {
              Path_2_Point_Crd=Path_2_Object->GetPoint(c);
              Path_2_Point_Crd.y= Path_2_Point_Crd.y+Y_Distanz_to_Water;
              Path_2_Object->SetPoint(c);
              println("Punkt ",c," der Biker_Path_Spline wird verschoben");
            }
          }
          Path_2_Object->Message(MSG_UPDATE);
        } //if(doc->FindObject("Biker_Path_Spline")) ENDE
       }//WASSERSTNAD BERÜCKSICHTIGEN ENDE
       else println("Der Punkt",Deepest_Point," ist nicht im Wasser!");
      } //if(Waterlevel_Already_Checked==False) ENDE
      else println("Punkt ",i," wird nicht überprüft, da dieser bereits vorher verschoben wurde!");
      Path_3_Object->SetPoint(i,New_Path_3_Point_Crd);
      Path_3_Object->Message(MSG_UPDATE);
      if(Change_Bridge_Symmetry==True)
      {
        var Ideal_Line_Start_Point,Ideal_Line_End_Point, New_Ideal_Endpoint,Ideal_Spline_Num,Degree_Ideal_L
        var Path_Point_New_Temp,Count,Degree_Path_Point;
        var Prefix;
        //1 ZWEI ENDPUNTE DER IDEALSPLINE FESTELGEN
        Ideal_Spline_Num=int(Anzahl_Splines/2);
        Ideal_Line_Start_Point=BRIDGE_VOLUME_POINTS_ARRAY[Ideal_Spline_Num][0];
        Ideal_Line_End_Point=BRIDGE_VOLUME_POINTS_ARRAY[Ideal_Spline_Num][SplinePunkte-1];
        New_Ideal_Endpoint=Ideal_Line_End_Point;
        New_Ideal_Endpoint.x=Ideal_Line_End_Point.x-Ideal_Line_Start_Point.x;
        New_Ideal_Endpoint.z=Ideal_Line_End_Point.z-Ideal_Line_Start_Point.z;
        Degree_Ideal_Line=New_Ideal_Endpoint.x/New_Ideal_Endpoint.z;
        Normal_Degree_Ideal_Line=New_Ideal_Endpoint.z/New_Ideal_Endpoint.x;
        XZ_Dist_2_Points_Calc(Ideal_Line_Start_Point,Ideal_Line_End_Point);
        Ideal_Line_Lenght=Dist_current_Points;
        Path_Point_Crd=Path_3_Object->GetPoint(i);
        Path_Point_New_Temp=Path_Point_Crd;
        Path_Point_New_Temp.x=Path_Point_Crd.x-Ideal_Line_Start_Point.x;
        Path_Point_New_Temp.z=Path_Point_Crd.z-Ideal_Line_Start_Point.z;
        Degree_Path_Point=Path_Point_New_Temp.x/Path_Point_New_Temp.z;
        if(Degree_Path_Point<0)
        {
          if(Change_Paths==1)
          {
            if(abs(Degree_Ideal_Line)>abs(Degree_Path_Point)) Prefix=1;
            if(abs(Degree_Ideal_Line)<=abs(Degree_Path_Point)) Prefix=-1;
          }
          if(Change_Paths==2)
          {
            if(abs(Degree_Ideal_Line)>abs(Degree_Path_Point)) Prefix=-1;
            if(abs(Degree_Ideal_Line)<=abs(Degree_Path_Point)) Prefix=1;
          }
        }
        else
        {
          if(Change_Paths==1)
          {
            if(abs(Degree_Ideal_Line)>abs(Degree_Path_Point)) Prefix=-1;
            if(abs(Degree_Ideal_Line)<=abs(Degree_Path_Point)) Prefix=1;
          }
          if(Change_Paths==2)
          {
            if(abs(Degree_Ideal_Line)>abs(Degree_Path_Point)) Prefix=1;
            if(abs(Degree_Ideal_Line)<=abs(Degree_Path_Point)) Prefix=-1;
          }
        }
        if(Change_Paths==0) Prefix=0;
        Zufallszahl_3=Zufall->Get01();
        Zufallszahl_3=Zufallszahl_3+1;
        New_Path_3_Point_Crd=Path_Point_Crd;
        New_Path_3_Point_Crd.x=Path_Point_Crd.x+Prefix*((WheelChair_Width/4)*Zufallszahl_3)*Normal_Degr
        New_Path_3_Point_Crd.z=Path_Point_Crd.z+Prefix*((WheelChair_Width/4)*Zufallszahl_3);
      }
      Path_3_Object->SetPoint(i,New_Path_3_Point_Crd);
    }
    Path_3_Object->Message(MSG_UPDATE);
    }
  }


  //MODIFY BRIDGE GEOMETRIE
  CHANGE_BRIDGE_PARAMETERS()
  {
    println(" ");
    println("BRÜCKENGEOMETRIE VERÄNDERN!");
    var a;
    //Was dient als Entscheidungsgrundlage für Veränderungen?

    println("Entscheidungsgrundlagen:");
    println("Auslastung Arm_1 durch Durchbiegung: ",Bending_of_Arm_1_Efficiency*100,"%.");
    println("Auslastung Arm_2 durch Durchbiegung: ",Bending_of_Arm_2_Efficiency*100,"%.");

    //Variablen auf Grundwerte setzen bevor sie neu beschrieben werden
    if(Beam_Number_Intern<=2) //1-2
    {
      Formula_Divident_Arm_1_Extern=Formula_Divident_Arm_1;
      Formula_Divident_Arm_2_Extern=Formula_Divident_Arm_2;
      Torque_Curve_Calc_Mode=1;
      Change_Bridge_Thickness=1;
      Section_Shape_Mode_Extern=1;
      Faktor_1_Enable_Extern=0;
      Faktor_2_Enable_Extern=0;
      println("MODIFY_PATHS_ON=True");
      MODIFY_PATHS(); //Pfade verändern.
      println("CHECK_SPLINE_DEGREES_ON=True");
      CHECK_SPLINE_DEGREES_ON=True;
      /*
      println("KNEAD_TO_PATHS_ON=True");
      KNEAD_TO_PATHS_ON=True;
      */
      println("PATH_CLEARANCE_ON=True");
      PATH_CLEARANCE_ON=True;
    }
    if(Beam_Number_Intern>3)//ab 4
    {
      //Vorbemessungsformel Ändern:
      println("Formula_Divident_Arm_1:",Formula_Divident_Arm_1," Bending_of_Arm_1_Efficiency: ",Bending_of_Arm_1_Efficiency);
      //if(Bending_of_Arm_1_Efficiency>0) Formula_Divident_Arm_1_Extern=Formula_Divident_Arm_1(Bending_of_Arm_1_Efficiency*1.1(
      if(Bending_of_Arm_1_Efficiency>0) Formula_Divident_Arm_1_Extern=Formula_Divident_Arm_1;//BLEIBT GLEICH
      println("Formula_Divident_Arm_2:",Formula_Divident_Arm_2," Bending_of_Arm_2_Efficiency: ",Bending_of_Arm_2_Efficiency);
      // if(Bending_of_Arm_2_Efficiency>0) Formula_Divident_Arm_2_Extern=Formula_Divident_Arm_2/(Bending_of_Arm_2_Efficiency
      //1.1 sind 10% mehr oder weniger Veränderung
      if(Bending_of_Arm_2_Efficiency>0) Formula_Divident_Arm_2_Extern=Formula_Divident_Arm_2;//BLEIBT GLEICH
      Formula_Divident_Arm_1_Extern=Formula_Divident_Arm_1;
      Formula_Divident_Arm_2_Extern=Formula_Divident_Arm_2;
      if(Bending_of_Arm_1_Efficiency==0.0) Formula_Divident_Arm_1_Extern=0;
      if(Bending_of_Arm_2_Efficiency==0.0) Formula_Divident_Arm_2_Extern=0;
      //Änderungen nach oben oder untenhin ausschließen
      //Change_Bridge_Thickness=1;
      //Change_Bridge_Symmetry=True;
      Torque_Curve_Calc_Mode=0;
      //Grundsätzliche Profil Kontur:
      Section_Shape_Mode_Extern=1;
      //KorrekturFaktor 1 erlauben 1:Ja / 2: Nein
      Faktor_1_Enable_Extern=0;
      //KorrekturFaktor 1 erlauben 1:Ja / 2: Nein
      Faktor_2_Enable_Extern=0;

      var Zufall, Zufallszahl_1,Zufallszahl_2,Cross_Efficiency;
      Zufall=new(Random);
      Zufall->Init(time());
      Zufallszahl_1=Zufall->Get01();
      Zufallszahl_2=Zufall->Get01();
      if(Zufallszahl_2>=0.9)
      {
        println("Pfade werden auch im Grundriss verändert");
        Change_Bridge_Symmetry=True;
        println("KNEAD_TO_PATHS_ON=True");
        KNEAD_TO_PATHS_ON=True;
      }
      println("MODIFY_PATHS_ON=True");
      MODIFY_PATHS(); //Pfade verändern.
      println("CHECK_SPLINE_DEGREES_ON=True");
      CHECK_SPLINE_DEGREES_ON=True;
      if(Zufallszahl_1<=0.1)
      {
        println("KNEAD_TO_PATHS_ON=True");
        KNEAD_TO_PATHS_ON=True;
      }
      println("PATH_CLEARANCE_ON=True");
      PATH_CLEARANCE_ON=True;
      Formula_Divident_Arm_1_Extern=Formula_Divident_Arm_1;
      Formula_Divident_Arm_2_Extern=Formula_Divident_Arm_2;
      Torque_Curve_Calc_Mode=0;
      Change_Bridge_Thickness=1;
      Section_Shape_Mode_Extern=1;
      Faktor_1_Enable_Extern=0;
      Faktor_2_Enable_Extern=0;
    }
```

```
}

//-------------------------------------------------------------------------------------------
//-------------------------------------------------------------------------------------------
main()
{
  var SplineName=("OriginalSpline1");
  doc=GetActiveDocument();
  if (ONOFF==TRUE && doc->FindObject("LoftObjekt")&& doc->FindObject(SplineName))
  {
    var cnt,a; // Zählvariablen
    BRIDGE_POINTS_ARRAY = new (array,Anzahl_Splines,SplinePunkte);
    BRIDGE_VOLUME_POINTS_ARRAY=new(array,Anzahl_Splines*2,SplinePunkte);
    Sum_of_Pyramides_Volumes=new (array,Anzahl_Splines-1,SplinePunkte-1);//Eigengewicht Annahme
    Ideal_Beam_Heights=new(array,Anzahl_Splines,SplinePunkte);
    //Eigengewicht annehmen.
    for(cnt=0;cnt<Anzahl_Splines-1;cnt++)
    {
      for(a=0;a<SplinePunkte-1;a++) Sum_of_Pyramides_Volumes[cnt][a]=Structure_Load; //Eigengewicht Annal
    }
    //NullObjekte als Ordner ERSTELLEN
    var STRUCTURE_NullObject;
    if(!doc->FindObject("STRUCTURE_CALC"))
    {
      var STRUCTURE_NullObject_Name=("STRUCTURE_CALC");
      STRUCTURE_NullObject=new(NullObject);
      STRUCTURE_NullObject->SetName(STRUCTURE_NullObject_Name);
      doc->InsertObject(STRUCTURE_NullObject,Null,Null);
      //Geometry_BalancePoints Ordner erstellen
    }
    println("-----------------------------------------------------------------------------------------------------");
    println("TRAGWERKSBEURTEILUNG START!!!");
    println("-----------------------------------------------------------------------------------------------------");

    //-------------------------------------------------------------------------------------------------------------
    //STARTBEDINGUNGEN
    //-------------------------------------------------------------------------------------------------------------
    Section_Shape_Mode=1; //Definiert wie sich das Profil zum Rand hin verläuft;0=rechteckig,1=konkav,2=konve
    //für die Proportien der Profile
    if(Material_Typ==0) //STAHL
    {
      Obergurt_Mindesthoehe=2;
      Untergurt_Mindesthoehe=2;
      OG_Height_Percent=0.03;//0.12
      UG_Height_Percent=0.10;
      Left_Width_Percent=0.15; //Der Wert darf nicht höher als über 0.5 sein
      Right_Width_Percent=0.15;//Der Wert darf nicht höher als über 0.5 sein
      //Für Aushöhlungsprogramm
      Stegbreiten_Faktor=0.12; //Stegbreite muss mindestens z.B.: 8% der Höhe besitzen.
      Minimal_Construction_Height=Obergurt_Mindesthoehe+Untergurt_Mindesthoehe; //Concrete_Entwurf=35
    }
    if(Material_Typ==1) //STAHLBETON
    {
      Obergurt_Mindesthoehe=10;
      Untergurt_Mindesthoehe=7;
      OG_Height_Percent=0.05;//0.12
      UG_Height_Percent=0.30;
      Left_Width_Percent=0.25; //Der Wert darf nicht höher als über 0.5 sein
      Right_Width_Percent=0.25;//Der Wert darf nicht höher als über 0.5 sein
      //Für Aushöhlungsprogramm
      Stegbreiten_Faktor=0.12; //Stegbreite muss mindestens z.B.: 8% der Höhe besitzen.
      Minimal_Construction_Height=Obergurt_Mindesthoehe+Untergurt_Mindesthoehe; //Concrete_Entwurf=35
    }

    Beam_Proportion_Faktor=2.5; //Der Wert "0" bedeutet deaktivieren.//Der Wert 3 wurde für die Trophy verw
Stegabstands_Verhältnis = 1/0.4=2.25=>H/B=2.5 //Hat wesentlichen Einfluss auf Kurven verlauf in der Steitena
    Faktor_1_Enable==True; //1. Berücksichtigungsfaktor Ein/Aus = True/False -> Hier wird ein ideales Rechteckts
Brückenbreite (und einer Idealhöhe) berechnent und dann wird über das Widerstandsmoment eine neue Höhe b
    Faktor_2_Proportion_Value=2.5; //
    Faktor_2_Enable=False; //2. Berücksichtigungsfaktor Ein/Aus = True/False
    //Für den Stabilitätsnachweis
    Global_Safty_Factor=1.4;//Normal=1.4;//Globaler Sicherheitsbeiwert = 1.4; -> Für Vorbemessung ausreichen
    Waterlevel=Wasserpegel;
    Change_Bridge_Thickness=0;

    //Punkte_Array von der Brückenflächengeompetrie erzeugen
    GENERATE_BRIDGE_GEOMETRY_ARRAY(); //ERGEBNIS: BRIDGE_POINTS_ARRAY[][];->[SplineAnzahl][Sp

    //SURFACE_CALC
    BRIDGE_SURFACE_CALC(); //ERGEBNIS: EntirePlane,SinglePlane_BalancePoint[cnt][a],SinglePlanes[cnt][a]

    //Vom tiefsten Uferpunkt das Wasserlevel abziehn
    WATERLEVEL_POINT_CALC();

    Beam_Number_Intern=Beam_Num_In;
    if(Beam_Num_In<=1) //Externe BeamNummer
    {
      println("Es werden die Daten für den IDEALTRÄGER erzeugt! BeamNr.: ",Beam_Num_In,"!");
      println("Minimalhöhe vergeben!");
      if(Material_Typ==0) Formula_Divident_Arm_1=20; //Stahl
      if(Material_Typ==0) Formula_Divident_Arm_2=20; //Stahl
      if(Material_Typ==1) Formula_Divident_Arm_1=16; //Beton
      if(Material_Typ==1) Formula_Divident_Arm_2=16; //Beton
      //Volumshöhenpunkte zuweisen -> Für die Erste Volumsberechnung - Momentenverlauf
      for (a=0;a<SplinePunkte;a++)
      {
        for (cnt=0;cnt<Anzahl_Splines;cnt++)
        {
          BRIDGE_VOLUME_POINTS_ARRAY[cnt][a]=BRIDGE_POINTS_ARRAY[cnt][a];
          BRIDGE_VOLUME_POINTS_ARRAY[(Anzahl_Splines*2-1)-cnt][a]=BRIDGE_POINTS_ARRAY[cnt][a];
          BRIDGE_VOLUME_POINTS_ARRAY[(Anzahl_Splines*2-1)-cnt][a].y=BRIDGE_VOLUME_POINTS_ARRAY[(Anzahl_S
Minimal_Construction_Height;
        }
      }

      //START DER BERECHNUNG DES IDEALTRÄGERS -> 1.Vorbemessung
      //-------------------------------------------------------------------
      //Profiltypen pro Segmente vergeben
      DEFINE_PROFILETYPES();

      //Berechnung der Einzel_Volumen und des Gesamten Trägers
      BRIDGE_VOLUME_CALC();

      //LOADCALC
      LOAD_CALC(); //Achtung Load stimmt nicht da konstante Höhenannahme -> Minimal_Construction_Height

      //TORQUE_CALC
      TORQUE_CALC(); //Achtung Torque stimmt nicht da konstante Höhenannahme -> Minimal_Construction_Height

      //IDEAL_BEAM_CALC
      Section_Shape_Mode=0; //Rechtecksquerschnitt;
      Faktor_1_Enable==False;//Für Idealträger wird dieser Faktor ausgeschalten -> Sollte aber eigentlich eingeschalten we
      Torque_Curve_Calc=True;
      BEAM_CALC(Beam_Number_Intern);
      //Section_Shape_Mode=1; //Konkaver Querschnitt
    } //if(Beam_Num_In<=1) ENDE

    //-------------------------------------------------------------------
    //IDEALTRÄGER WURDE BERECHNET -> ES WURDE EIN ERSTES VOLUMEN VERGEBEN -> BRIDGE_VOLUME_POIN
    //-------------------------------------------------------------------
    println("-----------------------------------------------------------------------------------------------------");

    //ACHTUNG: DER VORHERGENDE TEIL KÖNNTE ÜBERSPRUNGEN WORDEN SEIN -> ES WIRD EIN AKTUELLES VOL

    if(Beam_Number_Intern>1)
    {
      Beam_Number_Intern=Beam_Number_Intern-1; //Da die vorhergehende Geomtrie eingelesen werden muss

      //DEN FITTESTEN TRÄGER SUCEHN
      CHOOSE_FITTEST_BEAM();

      //DATEN DES FITTESTEN TRÄGER EINLESEN
      READ_VARIABLE_FOLDER();

      //DAS BRIDGE_VOLUME_POINTS_ARRAY BERECHNEN
      READ_BRIDGE_SHAPE();
      Beam_Number_Intern=Beam_Number_Intern+1;
      //-------------------------------------------------------------------------------------------------------
      //VORHER ERZEUGTES TRÄGERVOLUMEN WURDE EINGELESEN -> Soll nun beurteilt werden!!!

      //CHANGE_Bridge_Shape;
      CHECK_AND_CHANGE_BRIDGE_SHAPE();

      //Profiltypen pro Segmente vergeben
      DEFINE_PROFILETYPES(); //Wenn die Beam_Number_Intern<=2 dann wird ein Vollquerschnitt berechnent

      //Externe Daten den Internen übergeben
      Formula_Divident_Arm_1=Formula_Divident_Arm_1_Extern;
      Formula_Divident_Arm_2=Formula_Divident_Arm_2_Extern;
      if(Torque_Curve_Calc_Mode==1) Torque_Curve_Calc=True;
      if(Torque_Curve_Calc_Mode==0) Torque_Curve_Calc=False;
      if(Faktor_1_Enable_Extern==0) Faktor_1_Enable=False;
      if(Faktor_1_Enable_Extern==1) Faktor_1_Enable=True;
      if(Faktor_2_Enable_Extern==0) Faktor_2_Enable=False;
      if(Faktor_2_Enable_Extern==1) Faktor_2_Enable=True;
      if(Section_Shape_Mode_Extern==0) Section_Shape_Mode=0; //Rechtecksquerschnitt
      if(Section_Shape_Mode_Extern==1) Section_Shape_Mode=1; //Konkav
      if(Section_Shape_Mode_Extern==2) Section_Shape_Mode=2; //Konvex
```

```
7312
7313      //NEUEN TRÄGER BERECHNEN ODER/UND ZEICHEN
7314      BEAM_CALC(Beam_Number_Intern);
7315      println("Nun wird dieser berechnet!");
7316      //Berechnung der Einzel_Volumen und des Gesamten Trägers
7317
7318      BRIDGE_VOLUME_CALC();
7319      //LOADCALC
7320      LOAD_CALC();
7321
7322      //TORQUE_CALC
7323      TORQUE_CALC();
7324    }
7325
7326    //MOMENTS OF INERTIA CALC
7327    MOMENTS_OF_INERTIA_CALC();
7328
7329    //BENDING_CALC
7330    BENDING_CALC();
7331
7332    //STABILITY CHECK------
7333    STABILITY_CHECK();
7334
7335    //MODIFY BRIDGE PARAMETERS
7336    CHANGE_BRIDGE_PARAMETERS();
7337
7338    //WRITE_VARIABLE_FOLDER
7339    WRITE_VARIABLE_FOLDER();
7340    Beam_Num_Out=Beam_Number_Intern;
7341    println("FIN");
7342    //-----------------------------------------------------------------------------------------------
7343    //Werte an Ausgang weitergeben (durchgeschliffen oder verändert)
7344    Anz_Splines_Out=Anzahl_Splines;
7345    Anz_Spline_Punkte_Out=SplinePunkte;
7346    Pedestrian_Width_Out=Pedestrian_Width;
7347    Biker_Width_Out=Biker_Width;
7348    WheelChair_Width_Out=WheelChair_Width;
7349    Barrier_Free_Path_Degree_Out=Barrier_Free_Path_Degree_In;
7350    Spline_Deformer_Factor_Out=Spline_Deformer_Factor_In;
7351    Path_Overlapping_Factor_Out=Path_Overlapping_Factor_In;
7352    //LoftNurbs-Updaten
7353    var LoftnurbsNEW,Loftname;
7354    Loftname=("LoftObjekt");
7355    LoftnurbsNEW=doc->FindObject(Loftname);
7356    LoftnurbsNEW#ID_BASEOBJECT_GENERATOR_FLAG=False;
7357    LoftnurbsNEW#ID_BASEOBJECT_GENERATOR_FLAG=True;
7358    OFF=FALSE;
7359  } // Ende ONOFF==TRUE
7360  else
7361  {
7362  if (ONOFF==TRUE)
7363  {
7364    println("Tragwerk kann nicht beurteilt werden, da kein LoftObjekt und keine Spline vorhanden sind!!!");
7365  }
7366 }
7367
7368
7369
7370 //---------------------------------
7371 //CREATE_HANDRAIL_SCRIPT  |
7372 //---------------------------------
7373 //by Christian Pichlkastner
7374 //
7375 //In diesem Code wird das Geländer für die Brücke generiert
7376 //
7377 //Eingang: Anzahl_Splines, SplinePunkte, Pedestrian_Width, Biker_Width, WheelChair_Width, ONOFF, Beam_Nu
7378 //Ausgang: OFF
7379
7380 //GLOBALE VARIABLEN
7381 var doc; // Damit Objekte in aktueller Szene benützt/gesucht werden können
7382 var Dist_current_Points;
7383 var BRIDGE_POINTS_ARRAY;
7384 var V_Angle;
7385 var Handrail_Height;
7386 var Thickness; //Brückengeländer_Dicke
7387 var Spitzen_Neigung; //In Grad
7388 //Punkte_Array von der Brückenflächengeompetrie erzeugen
7389
7390 //MAKE A 6 Points defined Spline
7391 MAKE_6_POINT_POLYGON(P1,P2,P3,P4,P5,P6,Name,Parent)
7392 {
7393   var Rectangle_Object;
7394   var Rectangle_Points=new(array,6);
7395
7396   if(!doc->FindObject(Name))
7397   {
7398     Rectangle_Object=new(SplineObject);
7399     var vc=new(VariableChanged);
7400     var bt=new(BackupTags);
7401     bt->Init(Rectangle_Object);    // required because we are changing the number of points
7402     vc->Init(0,6); // it used to have 0 points, now has 6
7403     if (Rectangle_Object->Message(MSG_POINTS_CHANGED, vc))
7404     {
7405       bt->Restore();
7406     }
7407     doc->InsertObject(Rectangle_Object,Parent, null);
7408     Rectangle_Object->SetName(Name);
7409     Rectangle_Object#SPLINEOBJECT_TYPE=0; // set the type of spline here
7410     Rectangle_Object#SPLINEOBJECT_CLOSED=True;
7411   }
7412
7413   Rectangle_Points[0]=P1;
7414   Rectangle_Points[1]=P2;
7415   Rectangle_Points[2]=P3;
7416   Rectangle_Points[3]=P4;
7417   Rectangle_Points[4]=P5;
7418   Rectangle_Points[5]=P6;
7419   //println(Path_Punkte);
7420   Rectangle_Object=doc->FindObject(Name);
7421   Rectangle_Object->SetPoints(Rectangle_Points);
7422   Rectangle_Object->Message(MSG_UPDATE);
7423 }
7424
7425 //HANDRAIL
7426 GENERATE_HANDRAIL()
7427 {
7428   //1BEAMKONTUR19
7429   //AUßENHÜLLEN (Profil-)PUNKTE abtasten
7430   var a,cnt,i;
7431   var Beam_Kontur_Name,Beam_Kontur_Object;
7432   var Kontur_Points=new(array,Anzahl_Splines*2);
7433   var Abweichung,Mittlere_Spline_1,Mittlere_Spline_2,Middle_Spline_Point;
7434   var X_Deg,Y_Deg,Z_Deg;
7435   var P1,P2,P3,P4,P5,P6;
7436   var Distance_to_Surf_Mid;
7437   var Profil_Balance_Point;
7438   var Rail_Height=4;
7439
7440   var Left_Rail_Folder_Object_Name,Left_Rail_Folder_Object;
7441   Left_Rail_Folder_Object_Name=(tostring(Beam_Num)+"_Left_Rail_Folder_Object");
7442
7443   if(!doc->FindObject(Left_Rail_Folder_Object_Name))
7444   {
7445     var STRUCTURE_NullObject_Name=(tostring(Beam_Num)+"_HANDRAIL");
7446     var STRUCTURE_NullObject=doc->FindObject(STRUCTURE_NullObject_Name);
7447     Left_Rail_Folder_Object=new(LoftObject);
7448     Left_Rail_Folder_Object->SetName(Left_Rail_Folder_Object_Name);
7449     doc->InsertObject(Left_Rail_Folder_Object,STRUCTURE_NullObject,Null);
7450   }
7451
7452   var Right_Rail_Folder_Object_Name,Right_Rail_Folder_Object;
7453   Right_Rail_Folder_Object_Name=(tostring(Beam_Num)+"_Right_Rail_Folder_Object");
7454
7455   if(!doc->FindObject(Right_Rail_Folder_Object_Name))
7456   {
7457     var STRUCTURE_NullObject_Name=(tostring(Beam_Num)+"_HANDRAIL");
7458     var STRUCTURE_NullObject=doc->FindObject(STRUCTURE_NullObject_Name);
7459     Right_Rail_Folder_Object=new(LoftObject);
7460     Right_Rail_Folder_Object->SetName(Right_Rail_Folder_Object_Name);
7461     doc->InsertObject(Right_Rail_Folder_Object,STRUCTURE_NullObject,Null);
7462   }
7463
7464   Left_Rail_Folder_Object=doc->FindObject(Left_Rail_Folder_Object_Name);
7465   Right_Rail_Folder_Object=doc->FindObject(Right_Rail_Folder_Object_Name);
7466
7467   for (a=0;a<SplinePunkte;a++)
7468   {
7469     Beam_Kontur_Name=(tostring(Beam_Num)+"BEAMKONTUR"+tostring(a));
7470     Beam_Kontur_Object=doc->FindObject(Beam_Kontur_Name);
7471     for (cnt=0;cnt<(Anzahl_Splines*2);cnt++)
7472     {
7473       Kontur_Points[cnt]=Beam_Kontur_Object->GetPoint(cnt);
7474     }
7475
7476     //Durchschnittshöhe berechnen
7477     var P1,P2,Temp_Height;
7478     var Middle_Profil_Height=0.0;
7479     Profil_Balance_Point=vector(0,0,0);
```

```
  for (cnt=0;cnt<Anzahl_Splines;cnt++)
  {
    P1=Kontur_Points[cnt];
    P2=Kontur_Points[cnt+Anzahl_Splines];
    XYZ_Dist_2_Points_Calc(P1,P2);
    Temp_Height=Dist_current_Points;
    Middle_Profil_Height=Middle_Profil_Height+Temp_Height;
    Profil_Balance_Point=Kontur_Points[cnt]+Kontur_Points[cnt+Anzahl_Splines];
  }
  Middle_Profil_Height=Middle_Profil_Height/Anzahl_Splines;
  Profil_Balance_Point=Profil_Balance_Point/(Anzahl_Splines*2);
  Abweichung=Anzahl_Splines/2;
  if(Abweichung!=int(Abweichung))
  {//ungerade Splineanzahl
    Mittlere_Spline_1=int(Anzahl_Splines/2)+1; //wenn immer abgerundet wird...dann stimmt dies //Ansonsten
    Mittlere_Spline_2=int(Anzahl_Splines/2)+1; //= Spline 1
  }
  else
  {//gerade Splineanzahl=2 Mittelsplines
    Mittlere_Spline_1=int(Anzahl_Splines/2)-1;
    Mittlere_Spline_2=int(Anzahl_Splines/2);
  }
Mittlere_Spline_1=Mittlere_Spline_1+Anzahl_Splines;
Mittlere_Spline_2=Mittlere_Spline_2+Anzahl_Splines;
P1=Kontur_Points[Mittlere_Spline_1];
P2=Kontur_Points[Mittlere_Spline_2];
Middle_Spline_Point=(P1+P2)/2;
//Middle_Spline_Point.y=Middle_Spline_Point.y-Middle_Profil_Height;
//LINKES GELÄNDER
var Handrail_Object_Name,Parent;
var Length;
Parent=doc->FindObject(tostring(Beam_Num)+"_HANDRAIL");
var Handrail_Folder_Object,Handrail_Folder_Name;
Handrail_Folder_Name=(tostring(Beam_Num)+"_RIGHT_HANDRAIL_LOFT"+tostring(a));
if(!doc->FindObject(Handrail_Folder_Name))
{
  var STRUCTURE_NullObject_Name=(tostring(Beam_Num)+"_RIGHT_HANDRAIL_LOFT");
  Handrail_Folder_Object=new(LoftObject);
  Handrail_Folder_Object->SetName(Handrail_Folder_Name);
  doc->InsertObject(Handrail_Folder_Object,Parent,Null);
}
Handrail_Folder_Object=doc->FindObject(Handrail_Folder_Name);
P1=Kontur_Points[(2*Anzahl_Splines)-1]; //Unterkante Rand=2.Polygonpunkt
XYZ_Dist_2_Points_Calc(P1,Middle_Spline_Point);
Length=Dist_current_Points;
X_Deg=(P1.x-Middle_Spline_Point.x)/Length;
Y_Deg=(P1.y-Middle_Spline_Point.y)/Length;
Z_Deg=(P1.z-Middle_Spline_Point.z)/Length;
P1=Kontur_Points[0];
P2=Kontur_Points[(2*Anzahl_Splines)-1];
var v1,v2;
v1=Kontur_Points[Mittlere_Spline_2-Anzahl_Splines];
v2=P2;
//Middle_Spline_Point
Angle_between_Vectors_Calc(Profil_Balance_Point,v1,v2);
Length=Handrail_Height*cos(V_Angle*(pi/180));
v1=P1;
v2=Kontur_Points[Mittlere_Spline_1];
XYZ_Dist_2_Points_Calc(v1,v2);
Distance_to_Surf_Mid=abs(Dist_current_Points);
P6=P1;
P6.x=P1.x+(X_Deg*(Length))/(Distance_to_Surf_Mid/62);
P6.y=P1.y+(Y_Deg*(Length))/(Distance_to_Surf_Mid/62);
P6.z=P1.z+(Z_Deg*(Length))/(Distance_to_Surf_Mid/62);
XYZ_Dist_2_Points_Calc(P1,P2);
Length=abs(Dist_current_Points);
P5=P1;
P5.x=P5.x-10;
P5.z=P5.z-10;
P5.y=P1.y+Handrail_Height;
P4=P5;
/*
P4.x=P5.x+(X_Deg*Length/3);
P4.y=P5.y+(Y_Deg*Length/3);
P4.z=P5.z+(Z_Deg*Length/3);
Spitzen Breite 3cm  */

P4.x=P5.x+(X_Deg*3);
P4.y=P5.y+(Y_Deg*3);
P4.z=P5.z+(Z_Deg*3);
//Mitte - Breite=Length*3.33
P3=P6;
P3.x=P6.x+(X_Deg*(Length*3.33));
P3.y=P6.y+(Y_Deg*(Length/1));
P3.z=P6.z+(Z_Deg*(Length*3.33));

Handrail_Object_Name=(tostring(Beam_Num)+"_LEFT_A_HANDRAIL"+tostring(a));
MAKE_6_POINT_POLYGON(P1,P2,P3,P4,P5,P6,Handrail_Object_Name,Handrail_Folder_Object);
var Rail_Object_Name=(tostring(Beam_Num)+"_LEFT_HANDRAIL"+tostring(a));
MAKE_RECTANGLE(P4,P5,Rail_Height,Rail_Object_Name,Left_Rail_Folder_Object);
P1.x=P1.x+Z_Deg*Thickness;
P1.z=P1.z+X_Deg*Thickness;
P2.x=P2.x+Z_Deg*Thickness;
P2.z=P2.z+X_Deg*Thickness;
P3.x=P3.x+Z_Deg*Thickness;
P3.z=P3.z+X_Deg*Thickness;
P4.x=P4.x+Z_Deg*Thickness;
P4.z=P4.z+X_Deg*Thickness;
P5.x=P5.x+Z_Deg*Thickness;
P5.z=P5.z+X_Deg*Thickness;
P6.x=P6.x+Z_Deg*Thickness;
P6.z=P6.z+X_Deg*Thickness;
Handrail_Object_Name=(tostring(Beam_Num)+"_LEFT_B_HANDRAIL"+tostring(a));
MAKE_6_POINT_POLYGON(P1,P2,P3,P4,P5,P6,Handrail_Object_Name,Handrail_Folder_Object);

//RECHTES GELÄNDER
Parent=doc->FindObject(tostring(Beam_Num)+"_HANDRAIL");
var Handrail_Folder_Object,Handrail_Folder_Name;
Handrail_Folder_Name=(tostring(Beam_Num)+"_LEFT_HANDRAIL_LOFT"+tostring(a));
if(!doc->FindObject(Handrail_Folder_Name))
{
  var STRUCTURE_NullObject_Name=(tostring(Beam_Num)+"_RIGHT_HANDRAIL_LOFT");
  Handrail_Folder_Object=new(LoftObject);
  Handrail_Folder_Object->SetName(Handrail_Folder_Name);
  doc->InsertObject(Handrail_Folder_Object,Parent,Null);
}

Handrail_Folder_Object=doc->FindObject(Handrail_Folder_Name);
//P1=Kontur_Points[2*Anzahl_Splines]; //Unterkante Rand=2.Polygonpunkt
var Handrail_Object_Name;
var Length;
P1=Kontur_Points[(2*Anzahl_Splines)-1]; //Unterkante Rand=2.Polygonpunkt
XYZ_Dist_2_Points_Calc(P1,Middle_Spline_Point);
Length=Dist_current_Points;
X_Deg=(P1.x-Middle_Spline_Point.x)/Length;
Y_Deg=(P1.y-Middle_Spline_Point.y)/Length;
Z_Deg=(P1.z-Middle_Spline_Point.z)/Length;
P1=Kontur_Points[Anzahl_Splines-1];
P2=Kontur_Points[Anzahl_Splines];
P3=P2;

v1=Kontur_Points[Mittlere_Spline_1-Anzahl_Splines];
v2=P2;

//Middle_Spline_Point
Angle_between_Vectors_Calc(Profil_Balance_Point,v1,v2);
Length=Handrail_Height*cos(V_Angle*(pi/180));
v1=P1;
v2=Kontur_Points[Mittlere_Spline_2];
XYZ_Dist_2_Points_Calc(v1,v2);
Distance_to_Surf_Mid=abs(Dist_current_Points);
P6=P1;
P6.x=P1.x-(X_Deg*(Length))/(Distance_to_Surf_Mid/62);
P6.y=P1.y+(Y_Deg*(Length))/(Distance_to_Surf_Mid/62);
P6.z=P1.z-(Z_Deg*(Length))/(Distance_to_Surf_Mid/62);
XYZ_Dist_2_Points_Calc(P1,P2);
Length=abs(Dist_current_Points);
P5=P1;
P5.x=P5.x+10;
P5.z=P5.z+10;
P5.y=P1.y+Handrail_Height;
P4=P5;

//Spitzen Breite 3cm
P4.x=P5.x-(X_Deg*3);
P4.y=P5.y+(Y_Deg*3);
P4.z=P5.z-(Z_Deg*3);

//Mitte - Breite=Length*3.33//Von Brückenbreite Abhängig
P3=P6;
P3.x=P6.x-(X_Deg*(Length*3.33));
P3.y=P6.y+(Y_Deg*(Length/3));
P3.z=P6.z-(Z_Deg*(Length*3.33));
Handrail_Object_Name=(tostring(Beam_Num)+"_RIGHT_A_HANDRAIL"+tostring(a));
MAKE_6_POINT_POLYGON(P1,P2,P3,P4,P5,P6,Handrail_Object_Name,Handrail_Folder_Object);
var Rail_Object_Name=(tostring(Beam_Num)+"_Right_HANDRAIL"+tostring(a));
MAKE_RECTANGLE(P4,P5,Rail_Height,Rail_Object_Name,Right_Rail_Folder_Object);
P1.x=P1.x+Z_Deg*Thickness;
```

```
  P1.z=P1.z+X_Deg*Thickness;
  P2.x=P2.x+Z_Deg*Thickness;
  P2.z=P2.z+X_Deg*Thickness;
  P3.x=P3.x+Z_Deg*Thickness;
  P3.z=P3.z+X_Deg*Thickness;
  P4.x=P4.x+Z_Deg*Thickness;
  P4.z=P4.z+X_Deg*Thickness;
  P5.x=P5.x+Z_Deg*Thickness;
  P5.z=P5.z+X_Deg*Thickness;
  P6.x=P6.x+Z_Deg*Thickness;
  P6.z=P6.z+X_Deg*Thickness;
  Handrail_Object_Name=(tostring(Beam_Num)+"_RIGHT_B_HANDRAIL"+tostring(a));
  MAKE_6_POINT_POLYGON(P1,P2,P3,P4,P5,P6,Handrail_Object_Name,Handrail_Folder_Object);
  }
 }

 //-------------------------------------------------------------------------------------
 //-------------------------------------------------------------------------------------
 main()
 {
  var SplineName=("OriginalSpline1");
  doc=GetActiveDocument();
  if (ONOFF==TRUE && doc->FindObject("LoftObjekt")&& doc->FindObject(SplineName))
  {
   println("GELÄNDER GENERIEREN!!!");
   var cnt,a; // Zählvariablen
   Spitzen_Neigung=60; //90 Grad bedeutet senkrechtes Geländer
   BRIDGE_POINTS_ARRAY = new (array,Anzahl_Splines,SplinePunkte);
   Handrail_Height=100;
   Thickness=3; //Metallkonstruktions_Dicke
   //NullObjekte als Ordner ERSTELLEN
   var STRUCTURE_NullObject;
   if(!doc->FindObject(tostring(Beam_Num)+"_HANDRAIL"))
   {
    var STRUCTURE_NullObject_Name=(tostring(Beam_Num)+"_HANDRAIL");
    STRUCTURE_NullObject=new(NullObject);
    STRUCTURE_NullObject->SetName(STRUCTURE_NullObject_Name);
    doc->InsertObject(STRUCTURE_NullObject,Null,Null);
    //Geometry_BalancePoints Ordner erstellen
   }

   //Punkte_Array von der Brückenflächengeompetrie erzeugen
   GENERATE_BRIDGE_GEOMETRY_ARRAY(); //ERGEBNIS: BRIDGE_POINTS_ARRAY[][];->[SplineAnzahl][Sp

   //HANDRAIL
   GENERATE_HANDRAIL();
   //println("FIN");
   //LoftNurbs-Updaten
   var LoftnurbsNEW,Loftname;
   Loftname=("LoftObjekt");
   LoftnurbsNEW=doc->FindObject(Loftname);
   LoftnurbsNEW#ID_BASEOBJECT_GENERATOR_FLAG=False;
   LoftnurbsNEW#ID_BASEOBJECT_GENERATOR_FLAG=True;
   OFF=FALSE;
  } // Ende ONOFF==TRUE
  else
  {
   if (ONOFF==TRUE)
   {
    println("Es kann kein Geländer erzeugt werden, da keine Geometrie vorhanden ist!!!");
   }
  }
 }
```