Technische Universität Graz
Institut für Maschinelles Sehen und Darstellen

Diplomarbeit

ERSTELLUNG VON SEMANTISCHEN GEBÄUDEMODELLEN AUS CAD-PLÄNEN

Martin Mörth

Graz im April 2010

Begutachter

Univ.-Prof. Dipl.-Ing. Dr.techn. Dieter Schmalstieg

Betreuer

Dipl.-Ing. Gerhard Schall

Kurzfassung

Ein semantisches Gebäudemodell vereint sämtliche, für den Anwender relevanten Daten zu einem Gebäude, in einem einzigen Modell. Ein solches Modell wäre für viele Applikationen in der Gebäudetechnik eine hervorragende Basis. Neben Visualisierungen für Sicherheits- und Gebäudeleitstände könnten auch übergeordnete Aufgaben zur Steuerung des Gebäudes einfacher umgesetzt werden. Wird ein Gebäudemodell zur zentralen Bezugsquelle für gebäudespezifische Informationen, bringt dies viele Vorteile mit sich. Bei Änderungen am System würde ein fehleranfälliges Aktualisieren von dezentralen Datenbeständen entfallen.

Viele wissenschaftliche Arbeiten beschäftigen sich mit der Problematik Gebäudemodelle automatisch aus digitalen und analogen CAD-Plänen zu erfassen. Aufbauend auf diesen Arbeiten, wird ein Ansatz zur Erfassung von semantischen Gebäudemodellen aus CAD-Plänen erarbeitet. In einer Bereinigungsphase werden zunächst topologische Fehler in der geometrischen Repräsentation der Pläne korrigiert. Symbole, die in speziellen Schichten der Pläne zur Repräsentation von Türen und Fenstern eingezeichnet sind, werden zur semantischen Anreicherung der Plandaten verwendet. Von den geometrischen Elementen des Plans aufgespannte Flächen werden ermittelt und anhand dieser semantischen Attribute klassifiziert.

Das so gewonnene semantische Flächenmodell eines Gebäudes wird in weiterer Folge in einer relationalen Datenbank abgespeichert. Es kommt dabei ein Datenbanksystem zum Einsatz, das die Repräsentation geometrischer Merkmale in den Tabellen unterstützt. Spezielle Funktionen können in Abfragen verwendet werden, um einen Mehrwert aus der geometrischen Repräsentation der Objekte zu erzielen.

Im letzten Teil der Arbeit soll der praktische Nutzen des Gebäudemodells verdeutlicht werden. In einer Beispielanwendung werden aus den Daten der relationalen Datenbank 3D-Modelle zur Visualisierung des repräsentierten Gebäudes generiert.

Abstract

Semantic building models combine all the information about a building which are of relevance to a specific user in one single model. Such models would form an excellent basis for a lot of applications in building services engineering. Besides visualization tasks for safety systems as well as building services management systems also higher-ranking tasks for the control of the building itself could be implemented more easily. Using a semantic building model as central storage for building specific information brings a lot of benefits. Error prone updates of distributed information representations could be omitted when undertaking system changes.

A lot of scientific research has been conducted in the field of generating semantic building models from digital and analog CAD-drawings. Based on this research an approach to generating such models from CAD-drawings has been worked out. Firstly, in a correction phase, topological errors in geometrical representations of the drawing's features are fixed. Then symbols stored in special layers of the drawing representing doors and windows are used to enrich the model with semantic information. Lastly, faces from geometrical features of the drawings are deduced and classified by using the semantic information from the previous stage.

This so created face-based semantic building model is then imported into a relational database managment system. The database system being used is able to support the representation of geometrical features in the tables themselves. Special functions can be used in queries to achieve an added value from the geometric representation of the objects.

The last part of the thesis shall demonstrate the practical usefulness of the system. In an example, three dimensional models are generated from a generated building model for an interactive visualization of the represented building.

Danksagung

Großer Dank gilt meinem Betreuer Dipl.-Ing. Gerhard Schall für die stets verlässliche Unterstützung beim Verfassen der schriftlichen Arbeit. Danken möchte ich Univ.-Prof. Dipl.-Ing. Dr.techn. Dieter Schmalstieg für die Freiheit, die er mir beim Ausarbeiten meiner Diplomarbeit zugesprochen hat und meinem Firmenchef, Dipl.-Ing. Heinz Christian Sigl, für etliche Stunden, die ich während der Arbeitszeit in meinen Abschluss investieren durfte.

Besonderer Dank gilt meiner Familie, die meinen Werdegang mit ihrer stetigen Unterstützung überhaupt erst möglich gemacht hat.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Ein semantisches Gebäudemodell vereint sämtliche, für alle Anwender relevanten Daten zu einem Gebäude, in einem einzigen Modell. Wichtig dabei ist, dass Objekte im Modell stets mit semantischer Information verknüpft sind. Beispielsweise wird die geometrische Information über den Verlauf einer Wand mit der semantischen Information verknüpft, dass es sich um eine Wand handelt. In einem herkömmlichen CAD-Plan ist diese Information meist nicht vorhanden, da eine Wand nur durch Liniensegmente repräsentiert ist, denen keine maschinenlesbare, semantische Bedeutung zugewiesen wurde. Der wichtigste Unterschied zwischen der Repräsentation eines Gebäudes durch eine Menge von CAD-Plänen und der Repräsentation durch ein semantisches Gebäudemodell ist also die erhöhte, maschinelle Lesbarkeit und Verarbeitbarkeit.

Semantische Gebäudemodelle sind schon seit langer Zeit ein aktives Forschungsgebiet. Der Ansatz ein semantisches Gebäudemodell als ganzheitliches zentrales Datenmodell im Bauwesen zu etablieren wird bereits seit mehr als 30 Jahren verfolgt. Für spezielle Aufgabenstellungen wie Evakuierungssimulationen, Simulation von Bränden, Strukturanalyse im Bauwesen sowie CAAD Applikationen, sind Gebäudemodelle seit jeher integraler Bestandteil von Softwarelösungen. Viele wissenschaftliche Arbeiten beschäftigen sich mit der Problematik Gebäudemodelle automatisch aus digitalen und analogen CAD-Plänen zu erfassen.

1.1 Motivation

Ein semantisches Modell eines Gebäudes wäre für viele Applikationen in der Gebäudetechnik eine hervorragende Basis. Neben Visualisierungen für Sicherheits- und Gebäudeleitstände könnten auch übergeordnete Aufgaben zur Steuerung des Gebäudes einfacher umgesetzt werden. Gewisse Funktionen in der Lichttechnik, wie zentrales Ein- und Ausschalten von Beleuchtungskörpern in ganzen Gebäudeteilen wären mit Hilfe eines hierarchischen Gebäudemodells einfach umzusetzen.

Wird ein Gebäudemodell zur zentralen Bezugsquelle für gebäudespezifische Informationen, bringt dies viele Vorteile mit sich. Vor allem bei Änderungen der Datenbestände würde ein fehleranfälliges Aktualisieren von dezentralen Datenbeständen entfallen.

CAD-Pläne, die in der Planungsphase eines Bauvorhabens entstehen, enthalten bereits einen Großteil der Informationen, die für die Erstellung eines Modells notwendig sind. Gelingt es diese Informationen zu verwerten, könnten Gebäudemodelle relativ einfach und automatisiert erstellt werden.

1.2 Zielsetzung

Ziel dieser Arbeit ist es, aus einer Reihe von CAD-Plänen ein semantisches Modell eines Gebäudes zu erfassen (Kapitel 3: Erfassung des Gebäudemodells aus CAD-Plänen). Das

Modell soll so beschaffen sein, dass Objekte durch Polygone repräsentiert sind. Es ist also nicht Ziel der Arbeit, ein realitätsgetreues Modell des Gebäudes zu erzeugen, sondern ein abstraktes, vereinfachtes Flächenmodell.

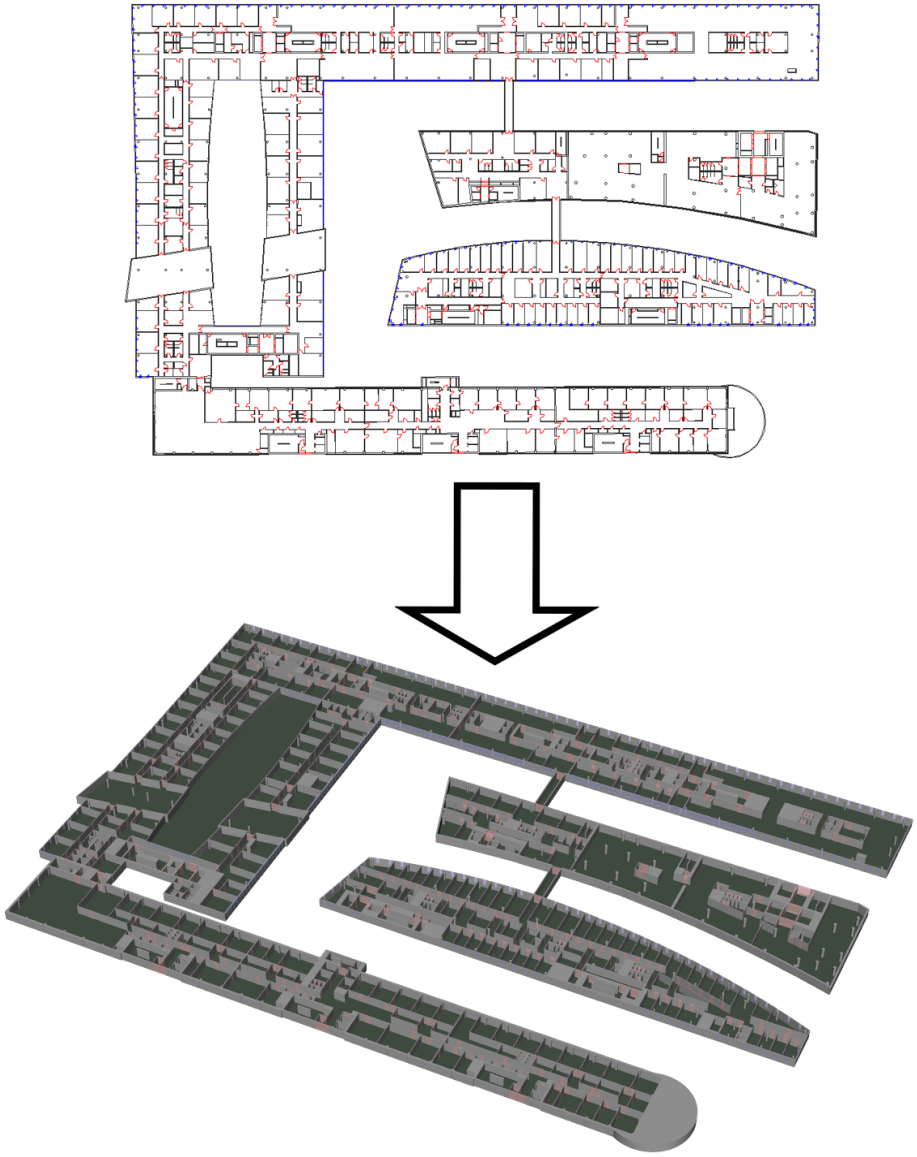


Abbildung 1: Vom CAD-Plan zur dreidimensionalen Visualisierung

Die Repräsentation des Modells soll dann in einer relationalen Datenbank erfolgen. Das Datenbanksystem soll mit Erweiterungen zur Verarbeitung geometrischer Daten ausgestattet sein. Dadurch wird sichergestellt, dass die in der Datenbank abgelegten Informationen optimal genutzt werden können (Kapitel 4: Repräsentation des Gebäudemodells in einer relationalen Datenbank).

Das Gebäudemodell soll in weiterer Folge als Basis für eine dreidimensionale Visualisierung des Gebäudes dienen (Kapitel 5: Übergabe des Gebäudemodells zur Visualisierung und Ergebnisse).

2 Semantische Gebäudemodelle

In engem Zusammenhang mit semantischen Gebäudemodellen steht der Begriff des Building Information Modeling (Gebäudedaten Modellierung, BIM). Bezeichnet wird damit ein integrierter Gesamtprozess, der die Planung, den Bau und die Bewirtschaftung von Gebäuden im Sinne aller Beteiligten innovativ umgestalten soll. Die Basis für die Umsetzung dieses Gesamtprozesses ist ein semantisches Gebäudemodell, das sämtliche Angaben zu Geometrie, räumlicher Anordnung und verbauten Komponenten beinhaltet. Ähnlich wie in den Bereichen des Fahrzeug-, Maschinen- und Anlagenbaus erwartet man sich von der Anpassung der Arbeitsprozesse eine Effektivitätssteigerung, Risikominimierung und Qualitätsverbesserung bei den Resultaten ([Moe06]).

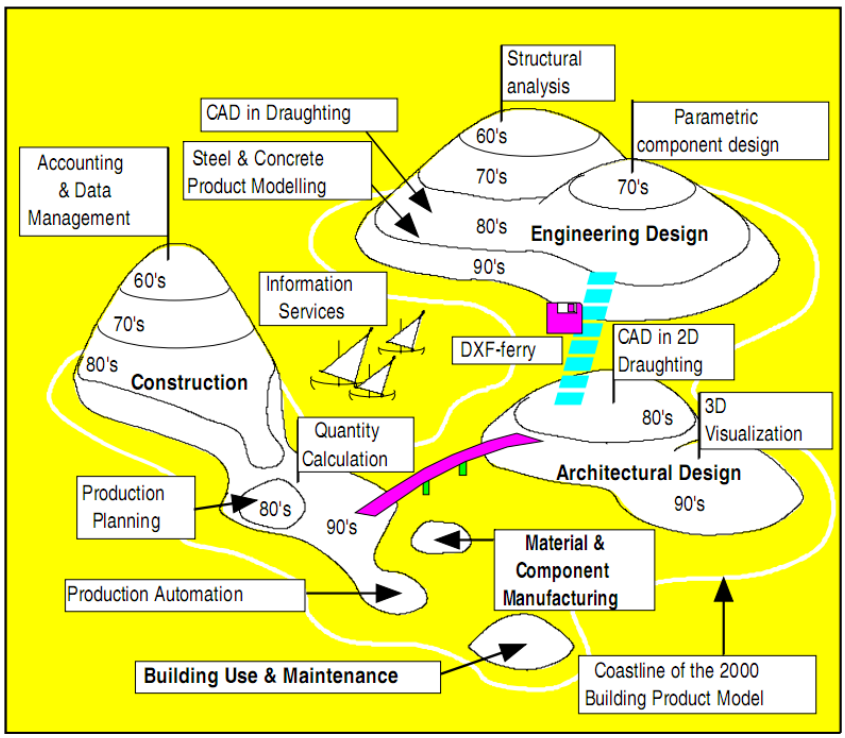


Abbildung 2: Insellösungen einzelner Fachbereiche im Bauwesen (aus [Bjö95])

In Abbildung 2 soll verdeutlicht werden, wie Insellösungen einzelner Fachrichtungen im Bauwesen über die Zeit immer mehr zusammenwachsen. Das Zusammenführen der Datenbestände aller Fachrichtungen, über ein gemeinsam genutztes, zentrales Gebäudemodell, ist eines der Ziele von BIM.

Die Idee den Entwicklungsprozess im Bauwesen auf die Basis eines ganzheitlichen Objektmodells zu stellen wird mittlerweile seit mehr als 30 Jahren verfolgt. Müssen Daten unter einer Vielzahl von Anwendungen und Beteiligten ausgetauscht werden, bieten sich grundsätzlich zwei Möglichkeiten, dies zu bewerkstelligen.

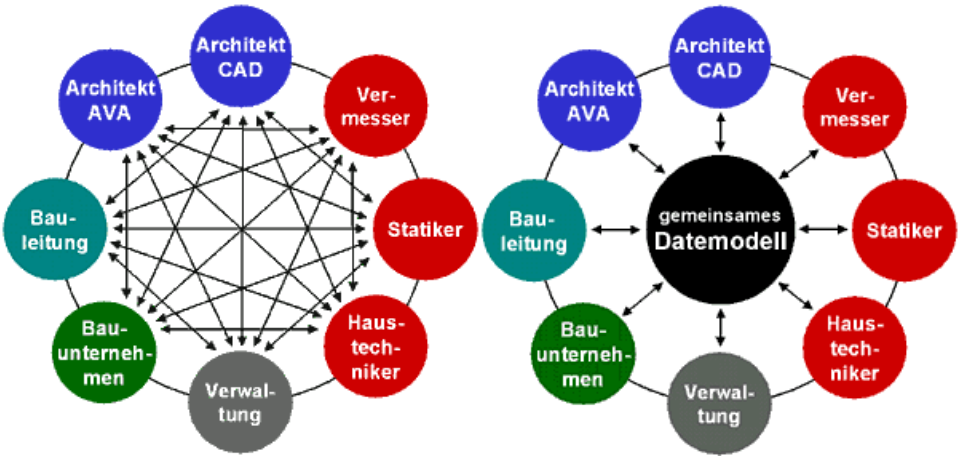


Abbildung 3: Datenaustausch zwischen mehreren Anwendungen (aus [Oeb97])

Wie aus Abbildung 3 ersichtlich, erfolgt der Austausch entweder direkt über eine Vielzahl spezieller Formate oder über ein gemeinsam genutztes, zentrales Datenmodell. Um ein solches Modell etablieren zu können, bedarf es zunächst eines umfassenden Standardisierungsprozesses.

Erste wissenschaftliche Arbeiten zum Thema semantische Gebäudemodelle (in der Literatur auch als Building Product Models, Gebäude Produktmodelle bezeichnet) wurden bereits in den späten 70er Jahren verfasst. Mitte der 80er Jahre konnten erste Fortschritte erzielt werden, als das „Standard for the Exchange of Product Model Data“ (Standard für den Austausch von Produktmodelldaten, STEP/ISO10303) Standardisierungsprojekt gestartet wurde. Ab 1995 übernahm die Industrieallianz für Interoperabilität (IAI) die Federführung bei den Standardisierungsbestrebungen im Bauwesen. Viele der im Rahmen von STEP entwickelten Ideen wurden auch für die Entwicklung des spezialisierten Standards weiterverwendet. Beispielsweise wurde weiterhin die Modellierungssprache EXPRESS zur Definition der Produktmodelle verwendet. Die erste Version des neuen, als Industry Foundation Classes (IFC/ISO16739) bezeichneten Objektmodells, wurde 1997 veröffentlicht. Seither wurde der Standard kontinuierlich weiterentwickelt und dem Ziel, als ganzheitliches Objektmodell im Bauwesen zu gelten, näher gebracht. Eine genauere Beschreibung des durch IFC spezifizierten Modells befindet sich in Kapitel 2.2.1 dieser Arbeit.

Obwohl mittlerweile einige Softwarelösungen über Schnittstellen verfügen und Referenzprojekte in verschiedenen Ländern umgesetzt wurden (z.B. [KFH+03]), ist die durchgehende Verwendung von IFC im gesamten Lebenszyklus von Gebäuden noch nicht sehr weit verbreitet. Einige Gründe, warum sich BIM und die damit verbundenen Änderungen der Arbeitsprozesse nur langsam durchsetzen, seien an dieser Stelle kurz aufgelistet (aus [HB05] und [HB07]):

- Projektteams haben ihre Arbeitsprozesse seit langer Zeit mit vorhandenen technologischen Mitteln optimiert (den eingesetzten fachspezifischen Softwarelösungen fehlen die Schnittstellen um sie in den Prozess integrieren zu können)

- die mit der gemeinsamen Nutzung eines zentralen Objektmodells impliziten Anforderungen wie Synchronisationsmechanismen und effiziente Verwaltung großer Datenmengen sind technologisch nicht leicht lösbar
- alternative Entwurfsideen sind mit einem einzigen, gemeinsam genutzten detaillierten Objektmodell nur schwierig umsetzbar
- in den Projektteams fehlen oft Koordinatoren die sich nur um den reibungslosen Ablauf der Arbeitsprozesse kümmern
- die Komplexität des zugrunde liegenden Objektmodells wird von den Softwarelösungen für den Benutzer meist nicht abstrakt und verständlich genug aufbereitet

Betrachtet man einzelne Fachrichtungen oder Problemstellungen im Speziellen, so sind auf den jeweiligen Zweck zugeschnittene, semantische Gebäudemodelle bereits seit langer Zeit integraler Bestandteil von Softwarelösungen. Für den Bereich des computergestützten Architekturentwurfs (Computer Aided Architectural Design, CAAD) ist mit ArchiCAD bereits 1987 einer der ersten Vertreter erschienen. Einige weitere Beispiele für mögliche Anwendungen von semantischen Gebäudemodellen sind:

- Evakuierungssimulationen ([ZLF03] und [HLZY08])
- Simulation von Bränden / Rauchausbreitung ([Lew96])
- Ausbreitung elektromagnetischer Wellen in Gebäuden ([DM99])
- Strukturanalyse im Bauwesen ([Rom05])
- Simulation von Raumluftströmungen ([vT04])
- Navigationshilfen ([SAT08] und [WBT07])
- Visualisierungen ([HD07])
- Augmented Reality Anwendungen ([SFJS05])

In weiterer Folge wird in diesem Kapitel noch genauer auf Erfassung, Repräsentation und Verwendung semantischer Gebäudemodelle eingegangen. Diese Unterteilung und Abbildung 4, welche diese verdeutlichen soll, ist einem Artikel zur Verwaltung komplexer Modelle im Bereich der Augmented Reality entnommen ([SSW+07]).

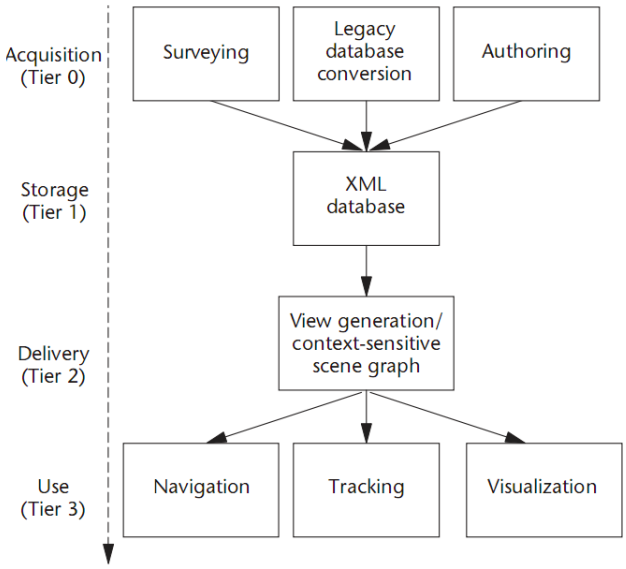


Abbildung 4: Von der Erfassung bis zur Verwendung eines Modells (aus [SSW+07])

Dabei werden im Kapitel über Erfassung („Acquisition“) Möglichkeiten diskutiert, semantische Gebäudemodelle mit Daten anzureichern. Die Repräsentation („Storage“) des Gebäudemodells bestimmt die Art und Weise wie die Daten des Gebäudemodells intern strukturiert und permanent gespeichert werden. Das Kapitel zur Übergabe („Delivery“) bezieht sich darauf, wie Daten aus semantischen Gebäudemodellen bezogen und diese zur Verwendung transformiert werden können. Eine Zusammenfassung und der Ansatz der gewählt wurde, um die Ziele dieser Arbeit zu erreichen, wird schließlich in den letzten beiden Teilen dieses Kapitels diskutiert.

2.1 Erfassung

Wie die Daten erfasst werden, die ein semantisches Gebäudemodell schlussendlich ausmachen, hängt letztendlich auch vom Verwendungszweck des Modells ab. Für den Fall des ganzheitlichen Gebäudemodells im Bauwesen werden Daten im Idealfall über den gesamten Entwicklungs- und Lebenszyklus des Gebäudes hinweg von Projektbeteiligten in das Modell eingegeben. Den Grundstein müssen Architekten mit ihren CAAD Anwendungen legen. In weiterer Folge wird das Modell von einer Vielzahl an Fachplanern mit ihren spezifischen Tools verfeinert („Authoring“ in Abbildung 3).

Eine weitere wichtige Anwendung von semantischen Gebäudemodellen ist die Bildung eines zugrunde liegenden Modells für Augmented Reality Systeme. Der Benutzer eines solchen Systems bewegt sich in der vom Modell repräsentierten realen Umgebung und bekommt, ortsbezogene, virtuelle Inhalte auf einem Anzeigegerät über dem realen Hintergrund eingeblendet.

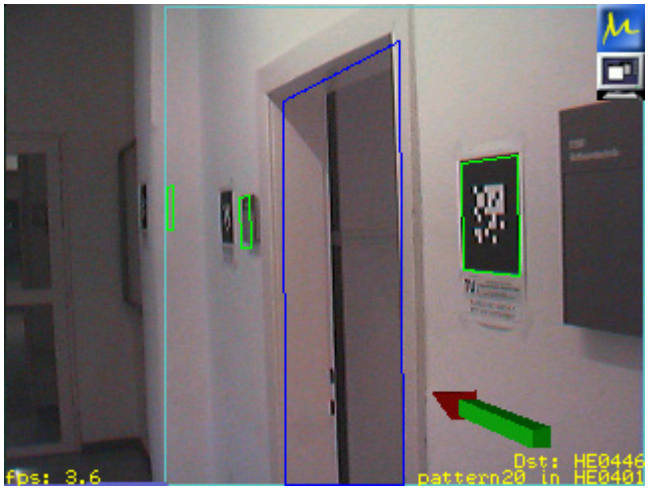


Abbildung 5: Signpost Augmented Reality Anwendung (aus [Sig03])

Modelle für solche Anwendungen beinhalten meist Positionen von Markierungen in der realen Welt, welche von Algorithmen aus dem Bereich des maschinellen Sehens in einem Kamerabild erkannt werden (siehe grüne Markierungen in Abbildung 5). Durch das Erkennen der Markierungen und das Wissen über deren Positionen aus dem Modell, kann so die Position und Blickrichtung des Betrachters bestimmt werden. Dies ist eine Voraussetzung dafür, dass virtuelle Inhalte, perspektivisch korrekt über das reale Bild geblendet werden können. Da die Positionsangaben für die Markierungen sehr genau sein müssen, wird das Modell im Fall von Augmented Reality Anwendungen oft auch mit händischen Vermessungsmethoden erstellt ([SFJS05], „Surveying“ in Abbildung 3).

Eine weitere Möglichkeit ein semantisches Gebäudemodell zu erzeugen, ist die Analyse einer Menge bereits vorhandener Repräsentationen, die als Ganzes implizit ebenfalls ein mehr oder weniger komplettes semantisches Modell eines Gebäudes definieren („Legacy Database Conversion“ in Abbildung 3).

Abbildung 6: Erfassung von Gebäudemodellen aus CAD-Plänen (aus [YWR09])

In diesem Zusammenhang, beschäftigt sich eine Vielzahl wissenschaftlicher Arbeiten mit der Gewinnung von semantischen Gebäudemodellen aus CAD-Plänen (Abbildung 6). Da dies auch die Methode sein wird, mit der sich diese Arbeit beschäftigt, wird an dieser Stelle genauer auf bestehende, wissenschaftliche Arbeiten zu dieser Herangehensweise eingegangen.

Bei der Erfassung semantischer Gebäudemodelle aus CAD-Plänen, stellt sich zunächst die Frage, in welcher Form die Pläne vorliegen. Sind nur handgezeichnete, oder ausgedruckte Pläne vorhanden, müssen diese zunächst digitalisiert werden.

2.1.1 Vorverarbeitung analoger CAD-Pläne

Nach dem Erfassen mit einem Scanner oder dem Abfotografieren mit einer digitalen Kamera liegen die Pläne als Rasterbilder vor, welche als Quelle für alle weiteren Verarbeitungsschritte dienen. Da das Digitalisieren von analogen Plänen nicht im Fokus dieser Arbeit liegt, seien die für die digitale Aufbereitung notwendigen Schritte nachfolgend aus [YWR09] kurz zusammengefasst.

2.1.1.1 Störungsbeseitigung

Für CAD-Pläne ergibt sich eine Vielzahl von unerwünschten Störungen. Im Prinzip gilt jeder Pixel als Störung, welcher nicht für die Generierung des Modells herangezogen werden kann. Einige Beispiele sind Annotationslinien, Dimensionierungslinien und Hintergrundmuster wie Gitternetzlinien. Eine Möglichkeit zwischen Störung und nützlicher Information zu unterscheiden, ist die Klassifikation von Linien nach Stärke. Den Verarbeitungsschritt der Störungsbeseitigung ohne Intervention des Benutzers vollkommen automatisch zu gestalten, ist jedoch so gut wie unmöglich.

2.1.1.2 Texterkennung

Überschneidungen zwischen Text und anderen Elementen, wie sie in CAD-Plänen häufig vorkommen können, erschweren die Arbeit von Texterkennungsalgorithmen. Im Großen und Ganzen gibt es zwei Ansätze zur Texterkennung:

- Strukturbasiert: Algorithmen machen sich die unterschiedliche strukturelle Ausprägung von Text und anderer grafischer Elemente zu nutze.
- Pixelbasiert: Aus einer gewissen Anzahl benachbarter Pixel werden Zusammenhangskomponenten gebildet, welche in weiterer Folge nach bestimmten Kriterien gefiltert werden (Größe, Farbintensität, Seitenverhältnis, Fläche, etc.).

2.1.1.3 Vektorisierung

In diesem Arbeitsschritt wird versucht einer Menge von Pixeln ein entsprechendes grafisches Element zuzuordnen. Traditionell sind Algorithmen zur Vektorisierung zweistufig umgesetzt:

- Erzeugung von Pixelketten aus dem Rasterbild
- Umwandlung der Pixelketten in Linien und Kreisbögen

Komplexere grafische Elemente wie Freiformkurven werden von Algorithmen zur Vektorisierung meist nicht erkannt.

2.1.1.4 Symbolerkennung

Dieser Schritt ist ein zentraler Punkt bei der Analyse von grafischen Dokumenten. Algorithmen zur Symbolerkennung werden in pixelbasierte und vektorbasierte Algorithmen unterteilt:

- Vektorbasiert: Eine Menge von benachbarten, grafischen Elementen (z.B. Linien und Kreisbögen) werden hinsichtlich ihrer Struktur und Charakteristik untersucht
- Pixelbasiert: Pixelformationen im Rasterbild werden direkt für statistische Auswertungen herangezogen

Ein Vorteil von vektorbasierten Algorithmen ist, dass skalierte und rotierte Symbole relativ leicht erkannt werden können. Pixelbasierte Verfahren sind meist präziser beim Erkennen von Symbolen, da zuvor keine Vektorisierung durchgeführt werden muss.

An diesem Punkt wird davon ausgegangen, dass sämtliche grafischen Elemente des CAD-Plans maschinenlesbar abgelegt und gegebenenfalls einem Symboltyp zugeordnet wurden. In weiterer Folge stellt sich die Frage, wie aus diesen noch immer sehr primitiven Informationen ein semantisches Gebäudemodell abstrahiert werden kann. In einem solchen Modell sollen ja alle grafischen Elemente durch semantische Objekte repräsentiert sein.

2.1.2 Building Model Generator von Rick Lewis

Der von Rick Lewis, in seiner Masterarbeit ([Lew96]) beschriebene Building Model Generator (Gebäudemodell Generator, BMG), beinhaltet eine Reihe von Werkzeugen, die es ermöglichen aus CAD-Grundrissplänen automatisiert ein detailliertes, semantisches und topologisch korrektes 3D-Modell eines Gebäudes zu erzeugen.

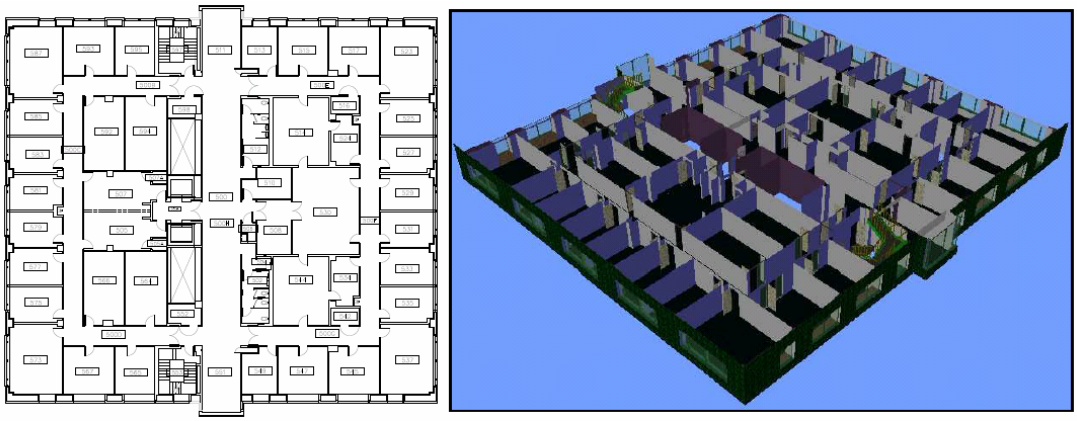


Abbildung 7: Building Model Generator von Rick Lewis (aus [Lew96])

Als Referenzgebäude dient das sieben Stockwerke und mehr als 300 Räume umfassende Soda Hall Gebäude der University of California, Berkeley. Das automatisch generierte Modell (siehe Abbildung 7) kann sowohl für interaktive Visualisierungen mit dem WALKTHRU System als auch für Feuersimulationen mit dem CFAST Feuersimulator eingesetzt werden, was den Nutzen deutlich macht. Eine genauere Betrachtung der möglichen Anwendungen befindet sich in Kapitel 2.3.1.

Als Eingangsdaten unterstützt der BMG 2D Grundrisspläne im Format AutoCAD DXF. Die Informationen aus den Plänen werden zunächst eingelesen und in eine interne Datenstruktur gespeichert, die effiziente geometrische Operationen unterstützt.

Danach werden zunächst kleine, lokal begrenzte, geometrische Inkonsistenzen korrigiert. Anhand der topologisch korrekten und gesäuberten Grundrisspläne wird eine semantische Analyse bezüglich Räumen und Portalen (Fenster und Türen) durchgeführt. In einem interaktiven Zwischenschritt kann der Benutzer noch kleinere Änderungen am Modell wie Größenanpassungen von Fenstern und Zuweisungen von Materialeigenschaften vornehmen. Im letzten Schritt werden die Modelle der einzelnen Stockwerke zu einem Gesamtmodell des Gebäudes kombiniert.

Im übrigen Teil dieses Kapitels werden für diese Arbeit relevante Lösungsansätze des Systems noch im Detail betrachtet.

2.1.2.1 Bereinigung der Geometrie

Am Computer gezeichnete CAD-Pläne beinhalten typischerweise eine Vielzahl an Fehlern und Mehrdeutigkeiten. Begrenzungen von Räumen bestehen meist nicht aus einem Linienzug, sondern aus einer Vielzahl einzelner Liniensegmente, welche sich auch

überlappen, überschneiden oder überhaupt nicht berühren können. Symbole die in CAD-Plänen Fenster und Türen repräsentieren passen oft nicht in die vorgesehenen Auslassungen in Wänden. Geometrische Elemente können doppelt im Plan eingezeichnet sein oder überhaupt fehlen.

Die Kanten aus einem CAD-Plan werden zunächst in einen ungerichteten Graphen eingefügt. Der erste Korrekturschritt sorgt dafür, dass kleine Lücken und Überschneidungen ausgebessert werden, indem alle Knoten des Graphen auf ein Flächengitter mit konstanten Abständen gezwungen werden. Die Abstände werden dabei automatisch aus den Elementen des Plans abgeleitet. Hierzu werden die am häufigsten auftretenden Knotenabstände ermittelt. Der größte gemeinsame Teiler dieser Abstände wird als Abstand für das Flächengitter herangezogen. Für die Pläne des Referenzgebäudes konnten mit dieser Methode sehr gute Ergebnisse erzielt werden.

Treten Lücken auf, die um einen gewissen Faktor größer sind als der Abstand des Flächengitters, können diese mit der oben beschriebenen Methode nicht korrekt geschlossen werden. Knotenpaare mit Grad eins, die einen genügend kleinen Abstand zueinander vorweisen, sind ein guter Indikator für solche Lücken.

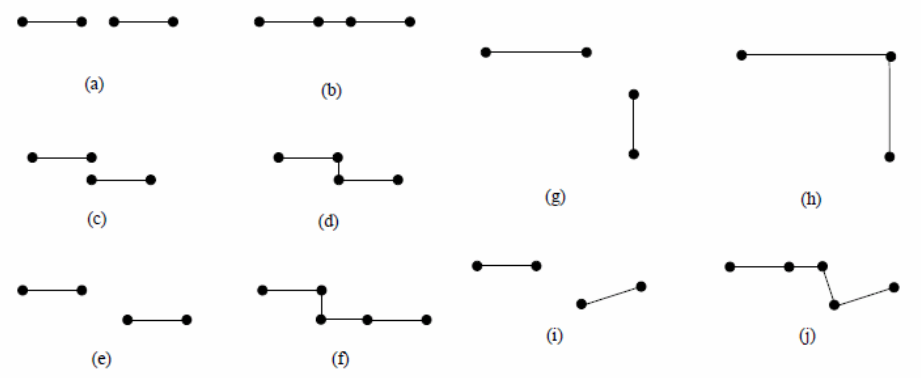


Abbildung 8: Behandlung von Lücken zwischen Kanten (aus [Lew96])

Sind die, von den Knoten eines solchen Paares ausgehenden Kanten parallel und kollinear (Abbildung 8a), können diese durch eine neue Kante verbunden werden (Abbildung 8b).

Sind ausgehende Kanten zwar parallel aber nicht kollinear (Abbildung 8c und e) werden die Kanten im rechten Winkel verbunden (Abbildung 8d und f).

Stehen die Kanten im rechten Winkel, oder nahezu im Rechten Winkel zueinander (Abbildung 8g), werden die Knoten des Paares an den Schnittpunkt der ausgehenden Kanten verschoben (Abbildung 8h).

In allen anderen Fällen (Abbildung 8i) wird zunächst die Verlängerung der einen ausgehenden Kante mit der Kante, die im rechten Winkel zur anderen ausgehenden Kante steht, geschnitten. Danach werden zwei neue Kanten eingefügt (Abbildung 8j).

Manchmal befindet sich in der Nähe eines Knotens vom Grad eins kein zweiter Knoten, der diese Bedingung erfüllt. In solchen Fällen wird der Benutzer aufgefordert den zweiten Punkt händisch auszuwählen.

Ebenfalls problematisch für die Weiterverarbeitung der Daten mit den Algorithmen zur Zuweisung semantischer Information sind überlappende Kanten.

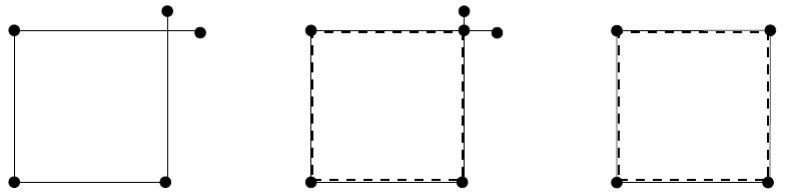


Abbildung 9: Behandlung von überlappenden Kanten (aus [Lew96])

Für diesen Fall wird der Schnittpunkt der Kanten berechnet und an dieser Stelle ein neuer Knoten eingefügt (Abbildung 9). Die ursprünglichen Endpunkte der Kanten werden in den Schnittpunkt verschoben. Für die überstehenden Kantenstücke werden nur Kanten eingefügt, wenn diese eine gewisse Mindestlänge überschreiten.

2.1.2.2 Gewinnung von semantischer Information

Die semantische Analyse der Geometriedaten läuft grundsätzlich in zwei Schritten ab:

- Im ersten Schritt werden die in den Plänen eingezeichneten Symbole für Fenster und Türen in topologisch und geometrisch verwertbare Elemente transformiert.
- Im zweiten Schritt wird versucht, abgeschlossene Linienzüge zu ermitteln, die alle im Plan eingezeichneten Bereiche begrenzen.

Die nachfolgend beschriebene Methode zur Konvertierung von Symbolen kann angewendet werden, weil die CAD-Pläne bereits ein gewisses Maß an semantischer Information beinhalten. Kanten und Kreisbögen, die zu Symbolen gehören sind bereits speziellen Schichten des Plans zugeordnet. Anhand dieser Zuordnung kann zwischen den unterschiedlichen Symboltypen (z.B. Fenster und Türen) unterschieden werden.

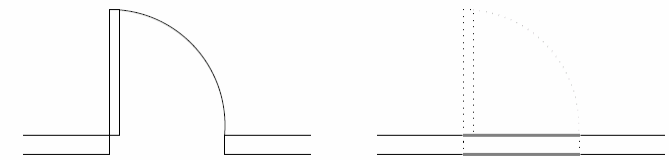


Abbildung 10: Umwandlung eines Türsymbols in zwei Kanten (aus [Lew96])

Typischerweise in CAD-Plänen eingezeichnete Türsymbole sind nicht für die direkte Extrusion von dreidimensionaler Geometrie geeignet. Ebenfalls sind topologische Begrenzungen nicht auf möglichst einfache Art und Weise, durch sie repräsentiert. Aus diesem Grund werden Türsymbole vom BMG durch zwei speziell gekennzeichnete parallele Kanten ersetzt (Abbildung 10).

Umgesetzt wird dies durch die Suche nach zwei parallelen Türstockkanten, die sich in der Nähe von Türsymbolen befinden. Die beiden Endknoten der gefundenen Kanten werden gegebenenfalls durch neue Spezialkanten verbunden. Da Fenstersymbole bereits korrekte

Begrenzungslinien beinhalten, müssen diese nicht speziell behandelt werden.

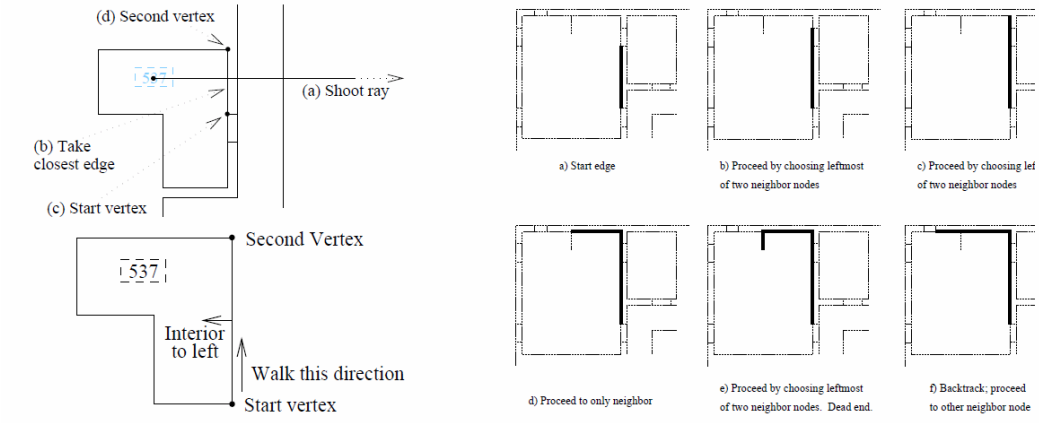


Abbildung 11: Suche nach internen Begrenzungslinien für Räume (aus [Lew96])

Im zweiten Schritt der semantischen Analyse werden die Begrenzungslinien für alle Bereiche des Plans ermittelt (Abbildung 11).

Grundsätzlich wird, von einem Startknoten ausgehend, ein Pfad im Graphen gesucht, der einen Bereich gegen den Uhrzeigersinn vollständig umschließt. Für Bereiche, die Räumen entsprechen, wird der Startknoten so gewählt, dass ein Strahl, von einer Raumbezeichnung ausgehend, in positive X-Richtung geschossen wird. Der gesuchte Knoten ist dann der weiter unten (in negativer Y-Richtung) gelegene Knoten, der ersten Kante, die vom Strahl getroffen wird.

Die Suche nach der Begrenzungslinie wird in positive Y-Richtung gestartet und linksdrehend fortgesetzt, wodurch sich die Begrenzung in Knotenreihenfolge gegen den Uhrzeigersinn ergibt.

Führt ein eingeschlagener Weg nicht zum Startknoten zurück, wird die Suche nach Rückschreiten zur letzten Abzweigung (Backtracking), mit der nächsten Kante fortgesetzt.

Nachdem alle Begrenzungslinien ausgehend von Raumbezeichnungen gefunden sind, werden mit demselben Algorithmus auch die äußeren Begrenzungen von Gebäuden ermittelt.

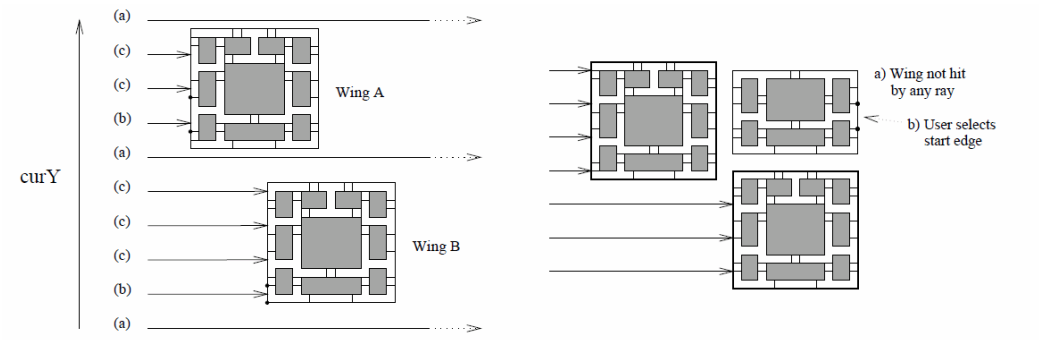


Abbildung 12: Suche nach externen Begrenzungslinien (aus [Lew96])

Der Startknoten wird in diesem Fall mit einem Strahl in positiver X-Richtung an der

Fassade des Gebäudes gesucht.

Es wird nicht nur ein Strahl geschossen, sondern mehrere Strahlen mit kontinuierlich erhöhter Y-Koordinate um alle Gebäudeteile zu finden. Wird ein Gebäude von einem anderen Gebäude in positiver X-Richtung verdeckt, muss der Startknoten vom Benutzer des Systems händisch gewählt werden (Abbildung 12). Im Unterschied zur Suche nach Raumbereichen, wird in diesem Fall vom Startknoten ausgehend in positiver Y-Richtung und rechtsdrehend, also im Uhrzeigersinn gesucht.

Parallel zur Suche nach den Bereichen wird ein Graph aufgebaut, der die abstrakte Semantik und Topologie des Gebäudes repräsentiert. Jeder Knoten dieses Graphen entspricht einem gefundenen Bereich und Kanten repräsentieren Nachbarschaften zwischen den Bereichen. Beispielsweise wird für jeden Bereich, der einem Raum entspricht, ein so genannter Raumknoten eingefügt. In diesem Knoten wird neben der Begrenzungslinie des Bereichs auch die Raumbezeichnung gespeichert, welche ja durch den Beschriftungsknoten bekannt ist, der als Ausgangspunkt für den Suchalgorithmus dient.

Neben den Begrenzungslinien für Raumbereiche und Gebäudeumrisse, werden mit dem zuvor beschriebenen Algorithmus auch Bereiche erkannt, die durch die speziell eingefügten Türkanten entstehen (siehe Abbildung 10). Sind zwei Räume durch eine Tür miteinander verbunden, so finden sich im Bereichsgraphen jeweils Kanten von den Raumknoten zum entsprechenden Portalknoten.

Optional können, in einem letzten Schritt, die Höhen der einzelnen Räume aus speziellen Geschoßdeckenplänen heraus, automatisch definiert werden.

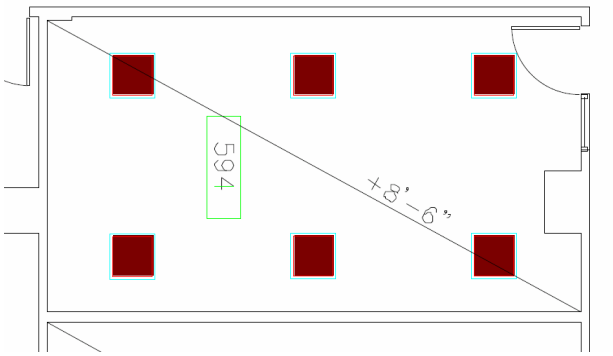


Abbildung 13: Ausschnitt aus einem Geschoßdeckenplan (aus [Lew96])

Die Information ist im Plan als Raumdiagonale inklusive Beschriftung repräsentiert (Abbildung 13). Die von den Diagonalen aufgespannten Rechtecke stimmen meist nicht exakt mit den zuvor gefundenen Raumbereichen überein. Mit Hilfe von booleschen Operationen wird die Raumgeometrie daher entsprechend den von den Diagonalen aufgespannten Rechtecken, aufgeteilt. Die dabei entstehenden kleineren Räume erben bereits zuvor festgelegte Attribute wie Raumbezeichnungen vom ursprünglichen Knoten des Bereichsgraphen.

Wie das jetzt, durch den Bereichsgraphen vollständig bestimmte, semantische

Gebäudemodell für Visualisierungen und Feuersimulationen eingesetzt werden kann, wird im Kapitel zur Übergabe von Gebäudemodellen (2.3.1) noch genauer betrachtet.

2.1.3 Automatic Unit Generator von Zhi, Lo und Fang

Im Jahr 2003 veröffentlichten Zhi, Lo und Fang ([ZLF03]) einen Artikel über die graphentheoretische Analyse von CAD-Plänen. Die Grundidee dabei ist es, den CAD-Plan in einen planaren Graphen umzuwandeln, aus dem zunächst eine Menge von Zyklen mit speziellen Eigenschaften ermittelt wird. In weiterer Folge werden aus diesen Basiszyklen Begrenzungslinien für die Räume abgeleitet. In einer Anwendung werden aus dem Gebäudemodell Fluchtrichtungen für Portale bestimmt, die in weiterer Folge für Evakuierungssimulationen genutzt werden (mehr dazu im Kapitel 2.3.2).

Der graphenbasierte Ansatz des Automatic Unit Generator (automatischer Bereichserzeuger, AUG), wie das System im Artikel bezeichnet wird, stützt sich im Wesentlichen auf folgende Feststellungen:

- Der CAD-Grundrissplan ist eine Kombination aus Bereichen
- Ein Bereich ist eine Kombination aus Zyklen
- Alle Zyklen des Grundrissplans bilden einen Vektorraum, dessen Basis aus einer spezielle Menge von Zyklen (Minimum Area Fundamental Loop Set, MAFL) gebildet wird
- Jeder gültige Bereich hat zumindest eine Öffnung, die mit einem anderen Bereich verbunden ist

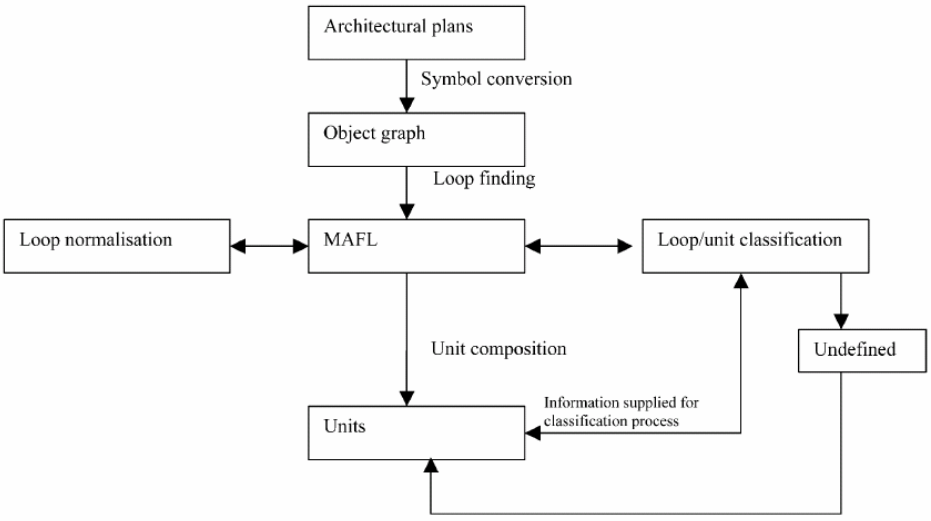


Abbildung 14: Arbeitsablauf im Automatic Unit Generator (aus [ZLF03])

Der Arbeitsablauf des AUG lässt sich grob in sechs Schritte einteilen, welche in den nun folgenden Absätzen genauer betrachtet werden (siehe Abbildung 14).

2.1.3.1 Bildung eines Objektgraphen aus den Plandaten

Beim Einfügen von Knoten und Kanten in einen Objektgraphen werden Korrekturmaßnahmen an den Plandaten vorgenommen. Der AUG sorgt dafür, dass Punkte, deren Distanzen zueinander unter einem bestimmten Grenzwert liegen, einfach zu einem Knoten zusammengefasst werden. Dieser Grenzwert wird zuvor empirisch ermittelt. Berühren sich zwei Kanten so, dass eine der Kanten am Berührungspunkt nicht durch einen Knoten geteilt ist, so wird die entsprechende Kante am Endknoten der anderen Kante aufgeteilt. Überschneiden sich zwei Kanten, so werden diese am Schnittpunkt geteilt. Türsymbole im Plan werden im Graphen durch entsprechende Trennkanten ersetzt (siehe gleicher Vorgang beim BMG in Abbildung 10).

Der resultierende Graph wird Objektgraph genannt, weil er mehr als nur geometrische Informationen enthält. Beispielsweise sind Türsymbole als nichtgeometrische Objekte enthalten, welche mit den bei der Symbolkonvertierung erzeugten geometrischen Trennkanten verknüpft sind. Der Objektgraph ermöglicht es so die ursprüngliche Darstellung der Elemente im CAD-Plan wiederherzustellen.

2.1.3.2 Identifikation der Zyklen aus dem Objektgraphen

Die geometrischen Kanten aus dem Objektgraphen werden für die Identifikation der Zyklen zunächst in einen neuen Graphen übergeführt. In diesem Graphen wird jede der ungerichteten Kanten aus dem Objektgraphen durch zwei gegensinnig gerichtete Kanten ersetzt (vergleiche Datenstruktur in Kapitel 3.4.1, Abbildung Abbildung 52). Während eines Durchlaufs des Identifikationsalgorithmus wird jede dieser gerichteten Kanten genau einmal abgearbeitet.

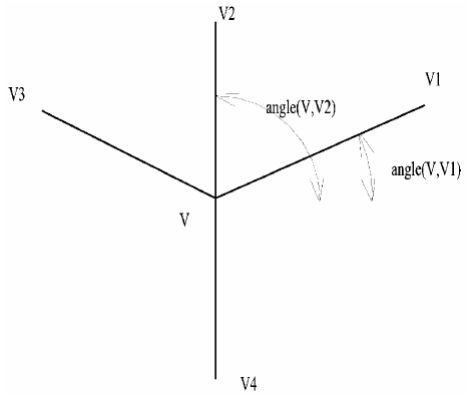


Abbildung 15: Sortierung der Kanten nach ihren Winkeln (aus [ZLF03])

In einem Vorverarbeitungsschritt werden zunächst für alle Knoten des Graphen die abgehenden Kanten, aufsteigend nach ihren Winkeln, sortiert (siehe Abbildung 15). Der Algorithmus wählt eine beliebige nicht markierte Startkante aus (als Beispiel nehmen wir an die Kante *(V2,V)* ist nicht markiert) und markiert diese.

Die Nachfolgekante für den Zyklus wird folgendermaßen bestimmt:

Ausgehend von der zur Startkante gegensinnig gerichteten Kante *(V,V2)* wird die, in der Liste der von *V* abgehenden Kanten, im Uhrzeigersinn nächste Kante ausgewählt. Diese Kante ist immer die korrekte Nachfolgekante für den zu bildenden Zyklus. Für die im Beispiel markierte Kante *(V2,V)* ist die von *V* ausgehende Kante *(V,V3)* der korrekte Nachfolger. Die Suche nach Nachfolgekanten wird solange fortgesetzt, bis der Algorithmus wieder an der Startkante angelangt ist. Sind alle Kanten markiert, so wurde jede Kante des Graphen einem Zyklus zugeordnet und der Algorithmus terminiert.

2.1.3.3 Vereinheitlichung der Zyklen

Da die im vorhergehenden Schritt gebildeten Zyklen noch nicht den gewünschten Basiszyklen entsprechen, müssen diese noch einer Vereinheitlichung unterzogen werden.

Als Klassifizierungsmerkmal dient der vorzeichenbehaftete Flächeninhalt des vom Zyklus umschlossen Polygons. Dieser Flächeninhalt wird nach folgender Formel berechnet:

$$A = \frac{1}{2} \cdot \sum_{i=0}^{n-1} \left(y_i + y_j\right)\left(x_i - x_j\right) \qquad \text{mit } j = \begin{cases} i+1 & \text{if } i < n \\ 0 & \text{if } i = n \end{cases} \tag{1}$$

Die Vereinheitlichung der Zyklen erfolgt nach folgenden Regeln:

1. $A = 0$: Degenerierter Zyklus (wird ignoriert)
2. $A < 0$: Äußere Begrenzungslinie (wird ignoriert)
3. $A > 0$: Es handelt sich um einen gültigen Basiszyklus

2.1.3.4 Klassifikation der Basiszyklen

Nun wird versucht den zuvor ermittelten Basiszyklen einen bestimmten Typ zuzuordnen. Die Klassifikation erfolgt mit Hilfe einfacher Regeln, die auf Eigenschaften der Zyklen angewendet werden. Je nach Anzahl von Tür- oder Fensterkanten und Flächeninhalt des eingeschlossenen Polygons, wird versucht jedem Zyklus einen der folgenden fünf Typen zuzuordnen:

1. Raum: Ein Bereich in dem sich Personen aufhalten können
2. Hindernis: Nicht begehbare Bereiche wie Mauern und Säulen
3. Tür: Zyklus bestehend aus zwei im Objektgraphen über ein Türsymbol verlinkten Türkanten und zwei Wandkanten
4. Fenster: Zyklus bestehend aus zwei im Objektgraphen über ein Fenstersymbol verlinkten Türkanten und zwei Wandkanten
5. Undefiniert: Zyklus konnte anhand der Regeln nicht eindeutig identifiziert werden

Meist können mit den oben genannten Regeln nicht alle Zyklen eindeutig bestimmt werden.

In einem zweiten Klassifizierungsschritt wird versucht die Bedeutung eines Zyklus anhand bereits klassifizierter Nachbarzyklen zu bestimmen. Ist ein undefinierter Zyklus von einem Raumzyklus nur durch eine Kante getrennt, wird angenommen, dass es sich ebenfalls um einen Raum handelt (die Kante dient nur zur Unterteilung der Räume).

2.1.3.5 Zusammensetzung der Bereiche aus den Basiszyklen

Die nun klassifizierten Basiszyklen müssen in einem letzten Schritt noch in Bereiche umgewandelt werden. Problematisch ist dies nur, wenn sich ein Zyklus geometrisch innerhalb eines anderen Zyklus befindet. Zyklen, die mit anderen Zyklen zwar benachbart sind, aber von keinem Zyklus umschlossen werden, können sofort in Bereiche umgewandelt werden.

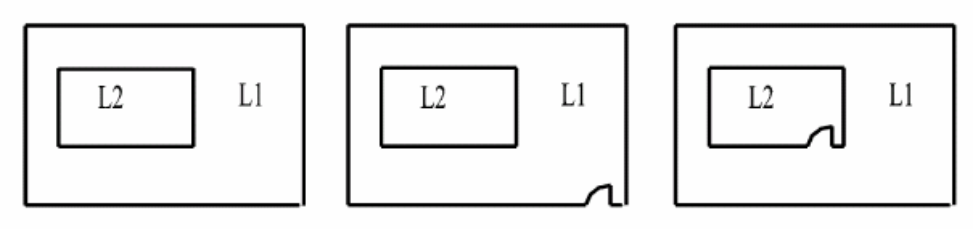


Abbildung 16: Hierarchische Anordnungen von Zyklen (aus [ZLF03])

Für die hierarchische Anordnung von zwei Zyklen *L1* und *L2* gibt es, abhängig davon ob diese Türkanten enthalten, einige mögliche Kombinationen, die speziell behandelt werden müssen. Um das Prinzip zu verdeutlichen werden nachfolgend drei häufig auftretende Kombinationen genauer betrachtet:

1. Beide Zyklen *L1* und *L2* beinhalten keine Türkanten: Zyklus *L2* wird entfernt und Zyklus *L1* bildet einen Bereich, der als Hindernis zu klassifizieren ist
2. Nur der äußere Zyklus *L1* beinhaltet eine Türkante: Zyklus *L1* und Zyklus *L2* bilden einen speziellen Bereich, wobei die Eckpunkte des inneren Zyklus *L2* im Uhrzeigersinn angeordnet werden (der Anordnung der Eckpunkte des äußeren Zyklus *L1* entgegengesetzt)
3. Nur der innere Zyklus *L2* beinhaltet eine Türkante: Es wird angenommen, dass es sich um einen Fehler im Plan handelt und *L1* wird in einen als Hindernis zu klassifizierenden Bereich umgewandelt

Sind alle Zyklen in Bereiche umgewandelt worden, ist das Gebäudemodell fertig bestimmt.

Vergleichbar mit dem BMG basiert auch dieses Modell auf der Semantik der klassifizierten Bereiche und der Topologie der Räume, welche durch die Nachbarschaften der Bereiche gegeben sind.

2.1.4 Weitere Arbeiten zur Erfassung aus CAD-Plänen

In einem Nachfolgeartikel zu seiner Masterarbeit beschreibt Rick Lewis die Erweiterung des BMG um die Möglichkeit, auch ineinander verschachtelte Begrenzungslinien zu verarbeiten ([LS98]). Dies ist eine wichtige Erweiterung, da diese Art der Anordnungen in CAD-Plänen häufig vorkommt.

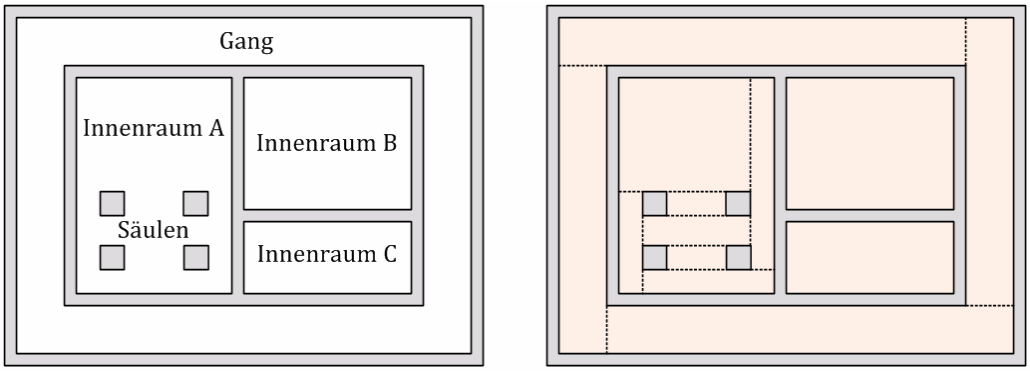


Abbildung 17: Ineinander verschachtelte Begrenzungslinien (inhaltlich aus [LS98])

Liegt ein Eckpunkt einer Begrenzungslinie innerhalb einer anderen Begrenzungslinie, handelt es sich um verschachtelte Begrenzungslinien. Diese Bedingung ist hinreichend, da in einem Vorverarbeitungsschritt sichergestellt wird, dass sich Begrenzungslinien nicht überschneiden.

Liegt eine Verschachtelung vor, werden mit booleschen Geometrieoperationen disjunkte, konvexe Teilbereiche gebildet, die in Summe die gewünschten eingeschlossenen Flächen ausmachen (siehe Abbildung 17).

Eine Arbeit von Alberto Palouzzi aus dem Jahr 2007 ([PS07]) beschreibt die Erfassung eines dreidimensionalen Gebäudemodells, mit Hilfe händischer Zuordnung einer Linienanordnung zu einem Muster. Das System ist dann in der Lage ähnlichen Anordnungen im Plan dasselbe Muster zuzuordnen. Eine Besonderheit der Arbeit ist, dass die Definition der Muster und ihrer geometrischen Entsprechung generativ mit Hilfe einer funktionalen Programmiersprache (PLaSM, [PPV95]) erfolgt.

Ein alternativer graphentheoretischer Ansatz zur Analyse von CAD-Plänen wird von Huang et al. in einem Artikel aus dem Jahr 2008 beschrieben ([HLZY08]). In diesem Fall werden die Zyklen nicht algorithmisch bestimmt, sondern algebraisch mit Hilfe von Matrizenoperationen berechnet.

2.2 Repräsentation

Nachdem das semantische Gebäudemodell erfasst ist, stellt sich die Frage, wie dieses in Datenstrukturen abgebildet und dauerhaft abgespeichert werden kann. In den bereits vorgestellten Arbeiten zur Erfassung von Gebäudemodellen wird meist nicht explizit auf die Speicherung des Modells eingegangen. Da im Zuge der Erfassung oft Graphen verwendet werden kann man davon ausgehen, dass zur Speicherung ähnliche Strukturen zum Einsatz kommen.

Im Bauwesen gibt es bereits seit langer Zeit umfangreiche Bestrebungen standardisierte Modelle für Gebäude zu erarbeiten. Auf diese Standards und artverwandte wissenschaftliche Arbeiten wird in den nun folgenden Abschnitten genauer eingegangen.

2.2.1 Industry Foundation Classes

Wie bereits in der Einleitung zu diesem Kapitel erwähnt, handelt es sich bei IFC um ein Objektmodell, welches den gesamten Projektentwicklungs- und Lebenszyklus eines Gebäudes abdecken soll. Der nachfolgende Abschnitt fasst die für diese Arbeit wichtigen Aspekte von IFC aus [Khe04] zusammen.

Da die von IFC abgedeckte Domäne und damit auch die Modellierung sehr komplex sind, soll zunächst anhand der Gesamtarchitektur ein Überblick verschafft werden.

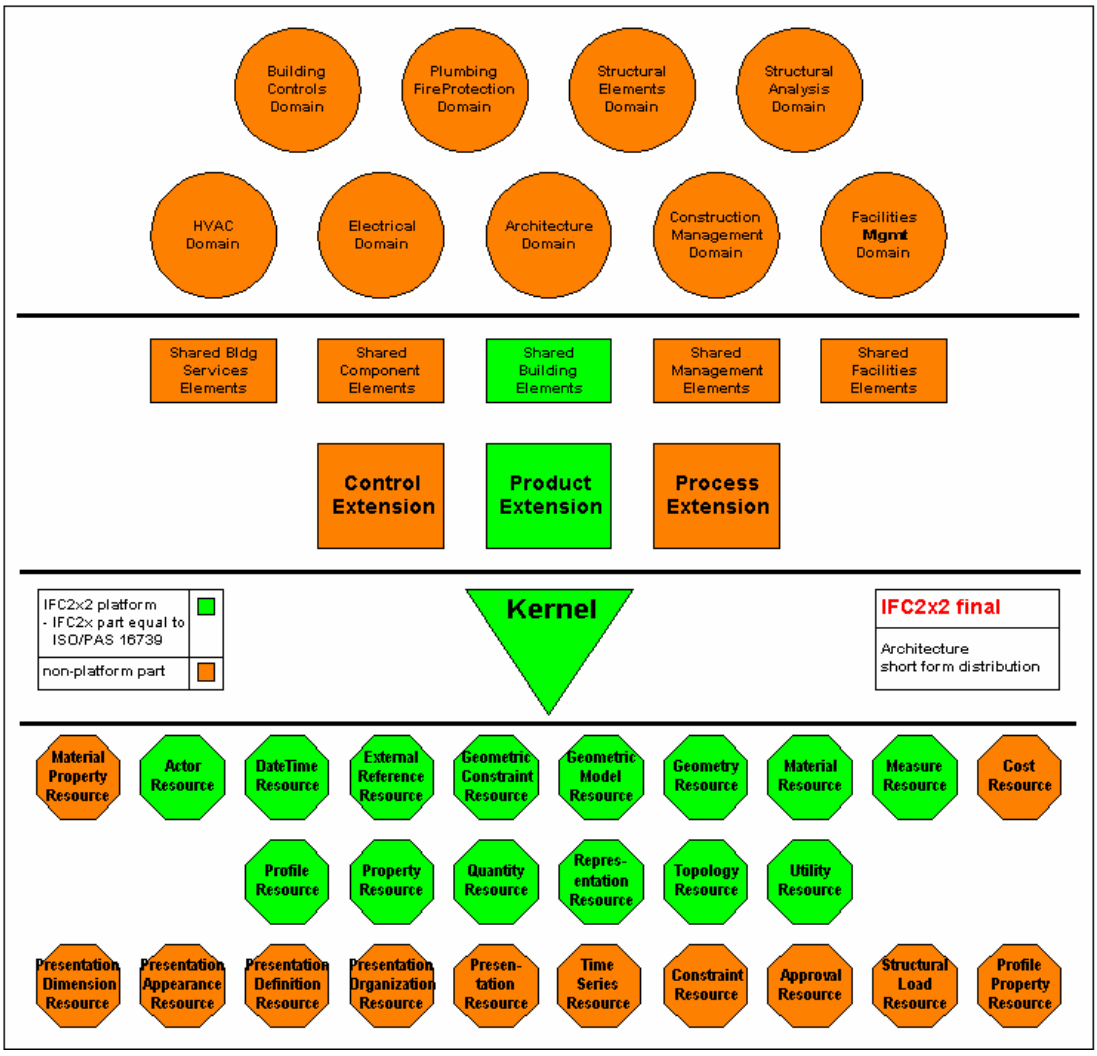


Abbildung 18: Gesamtarchitektur der Industry Foundation Classes

Grundsätzlich ist das IFC Objektmodell in vier Ebenen eingeteilt. Jede dieser Ebenen enthält Kategorien, die wiederum Objekttypen mit bestimmten Eigenschaften definieren (siehe Abbildung 18).

Es folgt eine Auflistung der vier Hauptebenen (beginnend Abbildung 18 unten) mit kurzer Beschreibung:

- Resource Layer (Ressourcenebene): Definiert grundlegende Objekttypen, die Eigenschaften wie Geometrie, Material etc. beschreiben.
- Core Layer (Kernebene): Definiert Objekttypen für abstrakte Konzepte wie Aktoren, Gruppen, Prozesse und Beziehungen. Hier werden auch gebäudespezifische Konzepte wie Raum, Ort, Gebäude und Gebäudeteil definiert.
- Interoperability Layer (Interoperabilitätsebene): Definiert konkrete Objekttypen wie Träger, Säulen, Wände und Türen, die von verschiedenen am Projekt beteiligten Personen ausgetauscht werden.

- Domain Layer (Fachrichtungsebene): Definiert Objekttypen die nur für bestimmte Fachrichtungen relevant sind (z.B. Boiler für den Bereich der Heizung, Lüftung und Klimatisierung).

Die im Sinne dieser Arbeit relevanten, konkreten Objekttypen sind fast ausschließlich in der Kategorie „Shared Building Elements“ (gemeinsam genutzte Gebäudekomponenten) der Interoperabilitätsebene definiert. Die in Abbildung 18 grün eingefärbten Kategorien sind Teil des Standards ISO16739. Diese Zertifizierung zeigt eine gewisse Reife des Modells und ist grundlegende Voraussetzung für Akzeptanz des Modells in der Industrie.

Nach der Betrachtung der Gesamtarchitektur soll nun die interne Strukturierung des Modells anhand einiger wichtiger Objekttypen und deren Beziehung zueinander verdeutlicht werden.

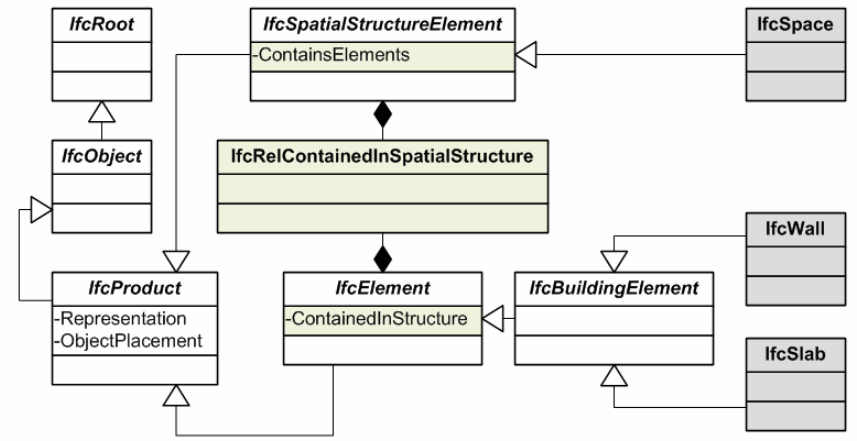


Abbildung 19: IFC Objekttypen in informeller UML Notation

Die ausgewählten Objekttypen sind zur Diskussion in einer informellen UML Notation dargestellt (Abbildung 19).

Ein wichtiger Objekttyp in der Ableitungshierarchie ist `IfcProduct`. Dieser Typ definiert mit der Eigenschaft `Representation` unter anderen die geometrische Repräsentation und mit der Eigenschaft `ObjectPlacement` die örtliche Platzierung eines Objekts.

Darunter teilen die Objekttypen `IfcSpatialStructureElement` und `IfcElement` die Ableitungshierarchie in zwei Äste.

Typen die von `IfcElement` und in weiterer Folge von `IfcBuildingElement` abgeleitet sind, bezeichnen konkrete Komponenten des Gebäudes, wie Wände (`IfcWall`) und Böden beziehungsweise Decken (`IfcSlab`).

Von `IfcSpatialStructureElement` abgeleitete Objekttypen beschreiben abstrakte Zusammengehörigkeiten dieser konkreten Typen. Mit Hilfe des Relationstyps `IfcRelContainedInSpatialStructure` können beispielsweise mehrere Wände (`IfcWall`) einem Raum (`IfcSpace`) zugeordnet werden. Der Zusammenhang zwischen Objekten ist also durch die Verknüpfung mit unabhängigen Relationsobjekten gegeben und nicht durch direkte Referenzierung der Objekte untereinander.

Die Speicherung und der Austausch von IFC Modelldaten können beispielsweise mit XML Dokumenten erfolgen. Zu diesem Zweck wird von der IAI, parallel zum IFC Standard, eine XML-Schemadefinition herausgegeben, die den Informationsgehalt von IFC vollständig abdeckt.

Weitere Bestrebungen gehen in die Richtung Modelldaten, ausgehend von einem Zentralen Modellserver, über Webservices auszutauschen (z.B. [BiM]). Ziel dieses Ansatzes ist es die Probleme in den Griff zu bekommen, die das gleichzeitige Arbeiten von mehreren Beteiligten an einem Gebäudemodell mit sich bringt.

2.2.2 City Geography Markup Language

Die City Geography Markup Language (CityGML) ist ein offener Standard für die geometrischen, topologischen und semantischen Aspekte eines 3D-Stadtmodells. Umgesetzt ist der Standard als Erweiterung der Geography Markup Language (GML). GML ist ein auf XML basierter Standard zur Repräsentation von Geodaten, der vom Open Geospatial Consortium (OGC) entwickelt wird. Im OGC haben sich um die 400 Privatunternehmen, Regierungsorganisationen und Universitäten mit dem Ziel zusammengeschlossen, offene Austauschformate für Geodaten zu entwickeln. Relevante Aspekte von CityGML werden nachfolgend aus [OZMF05] zusammengefasst.

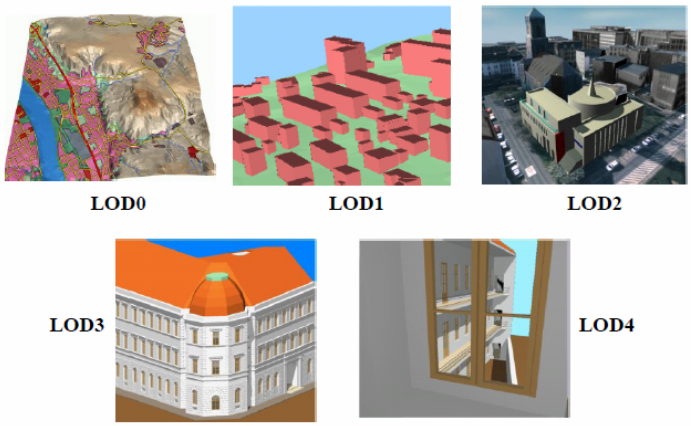


Abbildung 20: Unterschiedliche Detailstufen (LoDs) in der CityGML

Grundsätzlich ist die Repräsentation von Objekten in CityGML in unterschiedliche Detailstufen kategorisiert. Jede dieser Detailstufen (oder Level of Detail, LoD) spezifiziert, wie genau ein Objekt im Modell repräsentiert ist (Abbildung 20):

- LoD0 spezifiziert im Wesentlichen ein 2.5D Elevationsmodell der Landschaft über das Luftbilder gelegt werden können.
- In LoD1 werden Gebäude als Blöcke ohne Details wie Dächer dargestellt. LoD2 beinhaltet neben Vegetation und modellierten Dächern auch Texturen für die Gebäude.

- In LoD3 werden Gebäude bereits mit detaillierten Modellen der Architektur und der Dächer dargestellt. Da dieser Detailgrad mit Räumen, Türen, Treppen und Möblierung auch die interne Struktur von Gebäuden erschließt, ist er für diese Arbeit interessant.

Die konkrete Repräsentation der Objekte ist in zwei parallelen Hierarchien umgesetzt:

- Eine Hierarchie beschreibt die Geometrie und topologische Relationen und
- die zweite Hierarchie beschreibt die semantischen Zusammenhänge zwischen Objekten (z.B. mehrere Räume bilden eine Raumgruppe).

Beide Hierarchien können unabhängig voneinander navigiert werden und es kann zwischen den Hierarchien gewechselt werden.

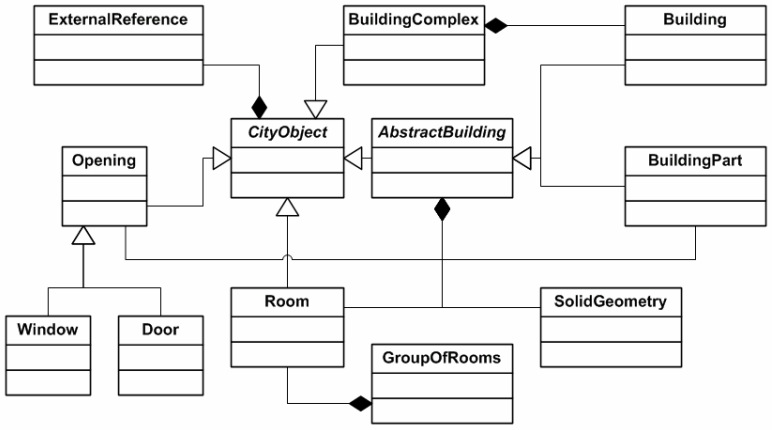


Abbildung 21: Wichtige Teile des CityGML Gebäudemodells in UML-Notation

Die Zentrale Klasse des CityGML Objektmodells ist die abstrakte Klasse `AbstractBuilding`. Von ihr abgeleitet sind die konkreten Klassen `Building` und `BuildingPart`. Da Instanzen von `AbstractBuilding` aus mehreren `BuildingParts` bestehen, welche wiederum `AbstractBuildings` sind, können beliebige Gebäudehierarchien umgesetzt werden.

Die meisten Klassen im Objektmodell sind von der Klasse `CityObject` abgeleitet. Ein wichtiges Attribut der Klasse `CityObject` ist `ExternalReference`. Stadtmodelle werden meist aus einer Vielzahl verschiedener Datenquellen generiert (z.B. Katasterpläne, IFC-Gebäudemodelle), die mit Hilfe des `ExternalReference`-Attributs bezeichnet werden. Durch diese Verbindung können die Stadtmodelle zu einem späteren Zeitpunkt aus den ursprünglichen Datenquellen aktualisiert werden.

Komplettiert wird das Gebäudemodell von den übrigen in `AbstractBuilding` definierten Attributen: Neben der Geometrie, werden Öffnungen (Klassen `Door` und `Window` abgeleitet von `Opening`) und die Innenräume (Klassen `Room` inklusive Gruppierung in `GroupOfRooms`) modelliert. Instanzen der Klasse `Room` definieren ebenfalls die Geometrischen Begrenzungen der Räume über `Ceiling`-, `InnerWall`- und `InnerFloorSurface` Objekte.

2.2.3 Weitere Arbeiten zum Thema Repräsentationen

Eine Arbeit von Benner aus dem Jahr 2005 ([BL05]) beschreibt das Quartierdaten-Managementsystem (QUASY) Gebäudemodell. Der eigentliche Fokus der Arbeit liegt auf dem Import von IFC Gebäudemodellen in QUASY und dem automatischen Generieren von Varianten der Gebäudemodelle zur effizienten Modellierung von Städten. Da das Modell dem Gebäudemodell in CityGML sehr ähnlich ist, wird hier nicht näher darauf eingegangen.

Über die Jahre haben verschiedene Hersteller von CAAD Software immer wieder eigene Gebäudemodelle zum Zweck des Datenaustauschs hervorgebracht. Ein Beispiele für ein solches in XML notiertes Gebäudemodell ist Green Building XML (gbXML, [gbX]).

2.3 Übergabe

Der letzte allgemeine Teil zum Thema semantische Gebäudemodelle beschäftigt sich damit, wie die Daten aus dem Modell verwendet werden können. Je nach Art der Anwendung müssen die Daten im Modell einer entsprechenden Transformation unterzogen werden.

2.3.1 Building Model Generator

Im Falle des BMG von Rick Lewis ([Lew96]) haben sich zur Zeit der Entstehung prinzipiell zwei mögliche Anwendungen für das Gebäudemodell ergeben: Einerseits die Erzeugung dreidimensionaler Geometrie zur interaktiven Visualisierung und andererseits die Erzeugung von Eingangsdaten für einen Feuersimulator.

2.3.1.1 Visualisierung

Im Falle der Visualisierung werden ausgehend vom Bereichsgraphen mit definierten Raumhöhen je nach Typ des vorliegenden Bereichs dreidimensionale Körper erzeugt.

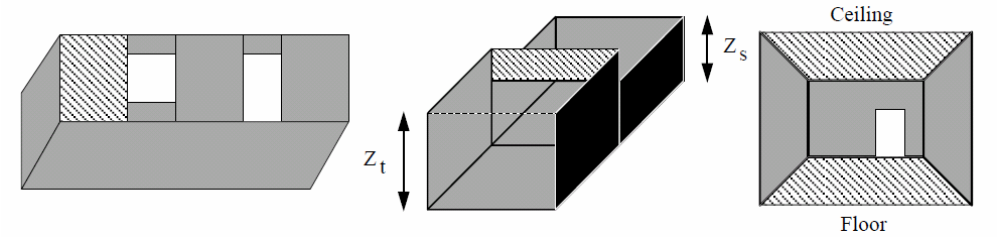


Abbildung 22: Erzeugung von Geometrie aus Bereichen (aus [Lew96])

Für normale Raumbereiche muss jede Kante der Begrenzungslinie, die nicht an einen Portalbereich anschließt, extrudiert werden. Kanten zur Raumtrennung müssen natürlich je nach Raumhöhenkonstellation im oberen Bereich extrudiert werden, um unterschiedliche Deckenhöhen auszugleichen. Für Böden und Decken werden nur dann Polygone erzeugt, wenn dies notwendig ist. Bereiche, die Stiegenhäuser repräsentieren,

benötigen beispielsweise nur im obersten Stockwerk Deckenpolygone (Abbildung 22).

Auf die Ausrichtung der erzeugten Flächen muss besonders geachtet werden: Normalvektoren für Flächen die aus Begrenzungslinien innerer Bereiche erzeugt wurden, zeigen in Richtung Bereichsmittelpunkt. Normalvektoren von Flächen, die aus äußeren Begrenzungslinien für Gebäude extrudiert werden, zeigen vom Gebäude weg.

Vor dem Zusammenfügen der geometrischen Entsprechungen der einzelnen Stockwerke zu einem Gebäude werden noch detaillierte 3D-Modelle für spezielle Objekte eingefügt. Anstelle von Portalbereichen werden beispielsweise prozedural erzeugte detaillierte Modelle von Türen eingefügt. Ebenfalls im BMG enthalten ist ein Tool zum Erzeugen von detaillierten 3D-Modellen für Stiegen. Die Parameter für die Erzeugung der Stiegen werden dabei ebenfalls aus den Abmessungen des entsprechenden Bereichs ausgelesen.

2.3.1.2 Feuersimulation

Eingangsdaten für den Feuersimulator unterliegen der Einschränkung, dass für die Repräsentation von Räumen nur achsenparallele, disjunkte Quader unterstützt werden. Die im Gebäudemodell repräsentierten Räume müssen also einer Vereinfachung unterzogen werden, damit sie verwendet werden können.

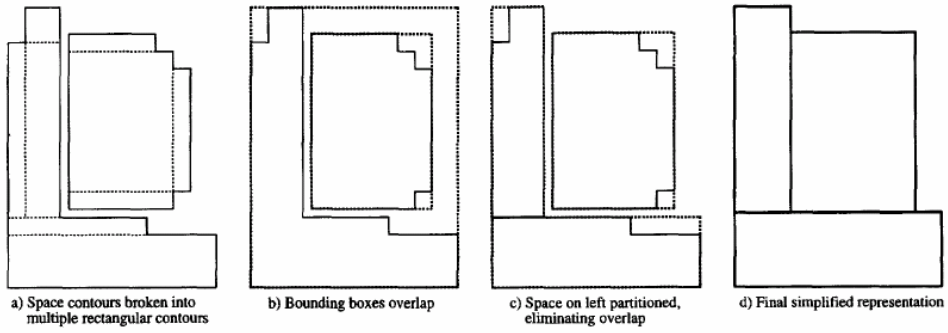


Abbildung 23: Repräsentation von Räumen durch disjunkte Quader (aus [LS98])

2.3.2 Evakuierungssimulation mit dem Automatic Unit Generator

Für die Anwendung des mit dem AUG gewonnenen Gebäudemodells zur Evakuierungssimulation, wird für jedes Portal eine Evakuierungsrichtung bestimmt.

Dafür wird zunächst jedem begehbaren Bereich im Gebäudemodell eine Sicherheitsstufe zugeordnet. Beispielsweise haben Stiegenhäuser eine höhere Sicherheitsstufe als Korridore oder Räume, die nur über eine Tür verfügen. Für benachbarte Bereiche mit unterschiedlichen Sicherheitsstufen ist so implizit eine Evakuierungsrichtung festgelegt.

Sind mehrere benachbarte Räume sicherheitstechnisch gleich eingestuft, wird mit Hilfe eines Algorithmus der kürzeste Weg zum nächsten sicheren Bereich bestimmt. Der Graph in dem der kürzeste Weg bestimmt wird, ist ein dualer Graph zum Bereichsgraphen, der durch das Gebäudemodell festgelegt ist. Portale sind in diesem Graphen als Knoten repräsentiert. Zwei Portalknoten sind dann durch eine Kante verbunden, wenn sie vom

gleichen Raum aus erreichbar sind. Das Gewicht der Kanten entspricht der euklidischen Distanz, die innerhalb des Raumes zwischen den Portalen zurückgelegt werden muss.

2.4 Diskussion

Die vorgestellten Arbeiten zeigen, dass sich mit Hilfe graphentheoretischer Algorithmen sehr gute Ergebnisse bei der Erfassung von Gebäudemodellen aus CAD-Plänen erzielen lassen. Der Ablauf zur Erfassung lässt sich stets in zwei Phasen gliedern:

- Bereinigungsphase: Die im Plan enthaltenen geometrischen Elemente werden von Fehlern bereinigt, die beim Zeichnen eines CAD-Plans entstehen. Elemente des Plans die für das Gebäudemodell nicht unmittelbar verwertbar sind, werden entfernt oder so transformiert, dass sie in das Modell aufgenommen werden können
- Analysephase: Die bereinigten Plandaten werden zunächst hingehend der Topologie untersucht. Dabei ergibt sich stets eine Menge von Flächen, die mit semantischer Information versehen werden müssen. Ein wichtiges Kriterium für diese semantische Analyse sind Spezialkanten, die in der Bereinigungsphase bei der Transformation von Plansymbolen gewonnen werden.

Ein wichtiges Merkmal der Flächen, die in der Analysephase entstehen, ist, dass diese nicht immer disjunkt sind. Ineinander verschachtelte Flächen müssen daher explizit repräsentiert und speziell behandelt werden.

Für die Repräsentation des Gebäudemodells müssen drei wichtige Aspekte bedacht werden:

- Geometrie: Begrenzungslinien für Flächen im Gebäudemodell
- Topologie: Nachbarschaften zwischen den Flächen im Gebäudemodell
- Semantik: Bedeutung einer Fläche im Gebäudemodell (Raum, Tür, Fenster)

Der konkrete Detailgrad des Gebäudemodells wird von den bei der Übergabe benötigten Daten bestimmt. Das Gebäudemodell muss also zumindest jene für die Anwendung notwendigen Daten beinhalten. Enthält das Modell mehr Daten als notwendig können diese bei der Transformation zur Übergabe verworfen werden.

2.5 Gewählter Ansatz

Die Gesamtarchitektur des in weiterer Folge in dieser Arbeit beschriebenen Systems wird im Folgenden kurz erläutert.

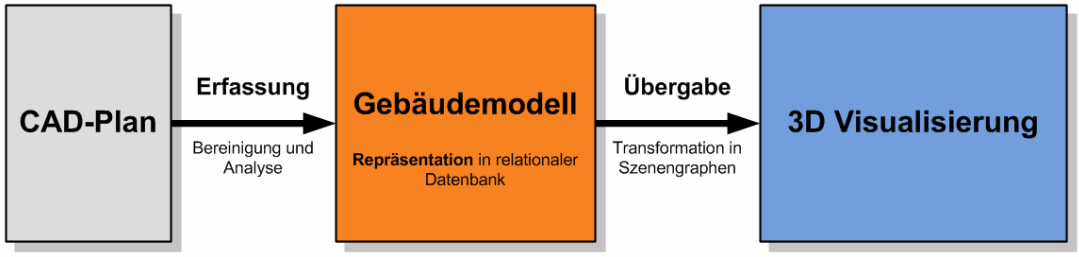


Abbildung 24: Von der Erfassung über Repräsentation bis zur Übergabe

Die konkrete Umsetzung der einzelnen Arbeitsschritte bis zur Verwendung des Gebäudemodells wird in Abbildung 24 schematisch dargestellt.

Für die Bereinigung der Rohdaten aus dem CAD-Plan werden, ähnlich wie in den vorgestellten Arbeiten, geometrische und graphentheoretische Verfahren zum Einsatz kommen (Kapitel 3.3: Bereinigung der Geometrie). Für die Analyse der Flächeninformationen kommen 2D-Anordnungen der bestehenden freien Geometriebibliothek CGAL zum Einsatz. Es handelt sich dabei um eine Datenstruktur mit zugehörigen Algorithmen, die sämtliche Aspekte der Flächenanalyse abdeckt (Kapitel 3.4: Analyse der Flächeninformationen).

Die Geometriebibliothek CGAL selbst ist in C++, nach modernen objektorientierten Gesichtspunkten implementiert. In weiterer Folge wird nur mehr oberflächlich auf Implementierungsdetails der Bibliothek eingegangen. Zu Spezialitäten der Implementierung steht eine Vielzahl wissenschaftlicher Arbeiten zur verfügbar (z.B. [FHH+00], [HH01] und [WFZH07]).

Freie Datenbanklösungen mit geometrischen Erweiterungen sind bereits seit einigen Jahren verfügbar (z.B. MySQL und PostgreSQL mit PostGIS). Da sich daraus interessante Aspekte für diese Arbeit ergeben, soll das gewonnene Modell in einer solchen relationalen Datenbank abgelegt werden (Kapitel 4: Repräsentation des Gebäudemodells in einer relationalen Datenbank).

Als Beispiel für die Übergabe und in weiterer Folge die Verwendung des Modells wird eine einfache isometrische Darstellung des Gebäudes in einer Visualisierung verwirklicht (Kapitel 5: Übergabe des Gebäudemodells zur Visualisierung).

Ziel für die Umsetzung ist es jedenfalls, möglichst viele bestehende und frei verfügbare Bibliotheken und Softwaresysteme einzusetzen.

3 Erfassung des Gebäudemodells aus CAD-Plänen

Dieses Kapitel widmet sich dem gewählten Ansatz zur Erfassung des semantischen Gebäudemodells aus CAD-Plänen. Die dafür notwendigen Schritte sollen in nachfolgender Abbildung verdeutlicht werden.

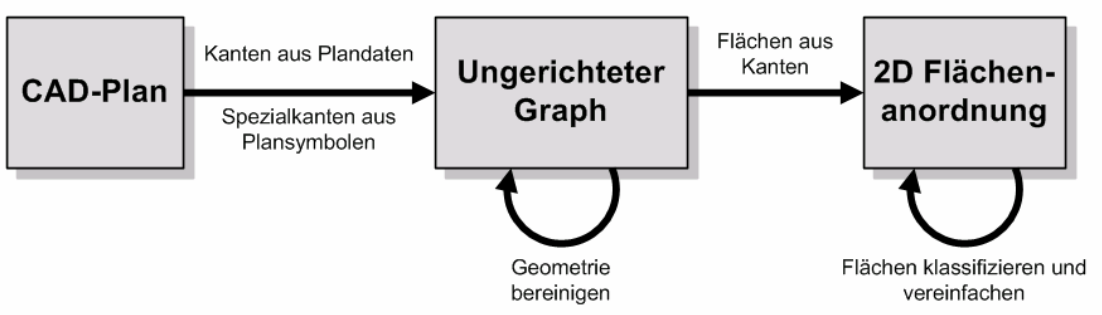


Abbildung 25: Übersicht über die Erfassung des Gebäudemodells aus CAD-Plänen

Die aus dem CAD-Plan gewonnenen Plandaten (Kapitel 3.1: Informationsgehalt von CAD-Plänen) werden zur weiteren Verarbeitung zunächst als Kanten in einen ungerichteten Graphen eingefügt (Kapitel 3.3.1: Ungerichteter Graph als Datenstruktur und Kapitel 3.3.2.1: Kanten aus Plandaten erzeugen bzw. 3.3.2.6: Spezialkanten aus Plansymbolen erzeugen).

Mit Hilfe von unterschiedlichen Operationen werden die Geometriedaten anschließend bereinigt (Kapitel 3.3.2.2, 3.3.2.3, 3.3.2.4 und 3.3.2.5).

Die Kanten im ungerichteten Graphen werden im nächsten Schritt verwendet, um Flächen zu bilden (Kapitel 3.4.2.1: Flächen aus Kanten erzeugen). Zur Analyse und Repräsentation der Flächen kommt ein Modul der bestehenden Geometriebibliothek CGAL zum Einsatz. Ein Überblick über die theoretischen Grundlagen zu diesem Modul befindet sich in Kapitel 3.4.1 (2D Anordnungen in CGAL).

Danach erfolgt die Klassifikation der Flächen anhand speziell markierter Kanten (3.4.2.2: Flächen durch Spezialkanten klassifizieren). In einem letzten Schritt werden diese klassifizierten Flächen dann noch vereinfacht (3.4.2.3: Flächen vereinfachen).

3.1 Informationsgehalt von CAD-Plänen

Der relevante Informationsgehalt eines CAD-Plans wurde anhand einer Reihe von Grundrissplänen bereits umgesetzter Bauvorhaben ermittelt. Im Grunde lässt sich die im Plan enthaltene Information in drei Kategorien einteilen:

- Basiselemente (Geometrie und Beschriftungen)
- Blöcke (Kombination aus Basisinformationen und Attributen)
- Zuweisung von Basiselementen und Blöcken zu Schichten

Geometrische Inhalte beschränken sich im Wesentlichen auf Linien, Kreisbögen und

Kreise. Oft sind aneinander anliegende Linien und Kreisbögen in einem Verbund zusammengeschlossen. Auf diese Art können beispielsweise Rechtecke aus vier verbundenen Linien gebildet werden.

Beschriftungen werden im Plan verwendet um eingezeichnete Objekte näher zu beschreiben.

Mit Hilfe von Blöcken können komplexere Objekte aus einer Menge von Basiselementen zusammengesetzt werden. Unterschieden wird dabei zwischen Blockdefinition und Blockreferenz. Eine Blockdefinition legt einen Typ fest, der dann später durch eine Blockreferenz instanziert werden kann. Zusätzlich können in einer Blockdefinition auch Attribute festgelegt werden. Attribute dienen zur näheren Beschreibung einer konkreten Blockreferenz. Informationen zu Räumen werden häufig mit Blöcken umgesetzt. Die Geometrie beschränkt sich dann auf eine Umrandungen, welche die Darstellung von Attributen wie Raumname, Raumnummer und ähnliches umschließt. Ist ein Block einmal definiert, kann er in der Zeichnung beliebig oft verwendet werden. Interessant an Blöcken ist die Tatsache, dass diese implizit immer eine gewisse Bedeutung haben. Sie sind daher meist gut geeignet den semantischen Informationsgehalt des abgeleiteten Gebäudemodells zu verbessern.

Zu guter letzt wird jedes Element einer Schicht des Plans zugeordnet. Der Zweck einer Schicht ist es, artverwandte Elemente zu gruppieren. Aus der Zuordnung von geometrischen Elementen des Plans zu Schichten wird in weiterer Folge die Semantik des Gebäudemodells abgeleitet. Geometrische Elemente, die Türen symbolisieren, müssen dazu einer bestimmten Schicht zugeordnet werden. Dasselbe gilt auch für Fenstersymbole.

Für die konkrete Einteilung der Schichten hat es bereits einige Versuche zur Standardisierung gegeben ([BLK97]). Über die Jahre hat sich allerdings gezeigt, dass sich diese Standards nur schwer durchsetzen lassen ([HB07]).

3.1.1 XML Schemadefinition für Plandaten

Die für diese Arbeit relevanten Elemente eines CAD-Plans wurden mit Hilfe einer XML Schemadefinition spezifiziert. Eine Schemadefinition ermöglicht es gültige Inhalte für XML Dokumente zu spezifizieren. Die Notation eines Schemas erfolgt wiederum in XML mit bestimmten vordefinierten Tags.

Sind gültige Inhalte eines XML Dokuments spezifiziert, kann automatisch überprüft werden, ob ein beliebiges XML Dokument einer erwarteten Struktur entspricht. Schlägt die Validierung eines Dokuments fehl, kann eine Anwendung den Ladevorgang vorzeitig abbrechen. Eine weitere Interpretation des Dokumentinhalts würde höchstwahrscheinlich fehlschlagen, da die Struktur des Dokuments nicht der erwarteten Struktur entspricht.

Im weiteren Verlauf dieses Abschnitts folgen eine inhaltliche Beschreibung des Schemas und eine genauere Betrachtung der Spezifikation eines ausgewählten Elements.

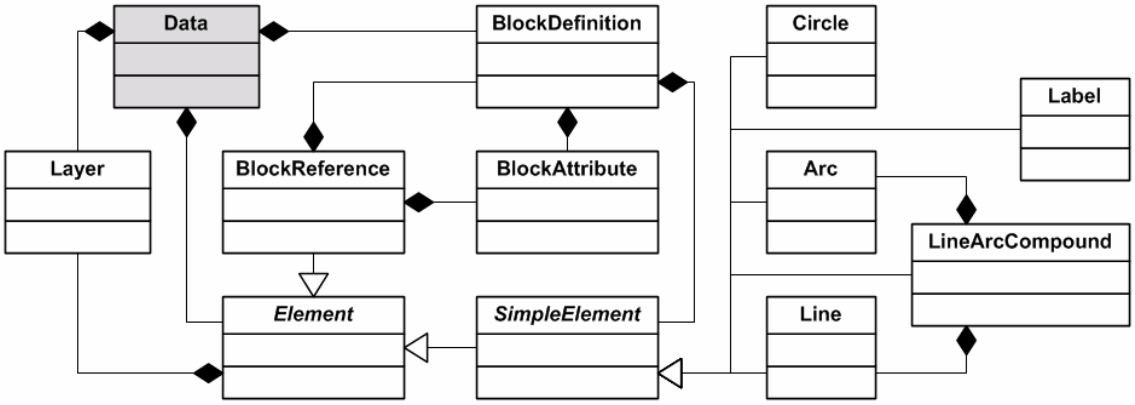


Abbildung 26: Elemente der Plandaten XML Schemadefinition

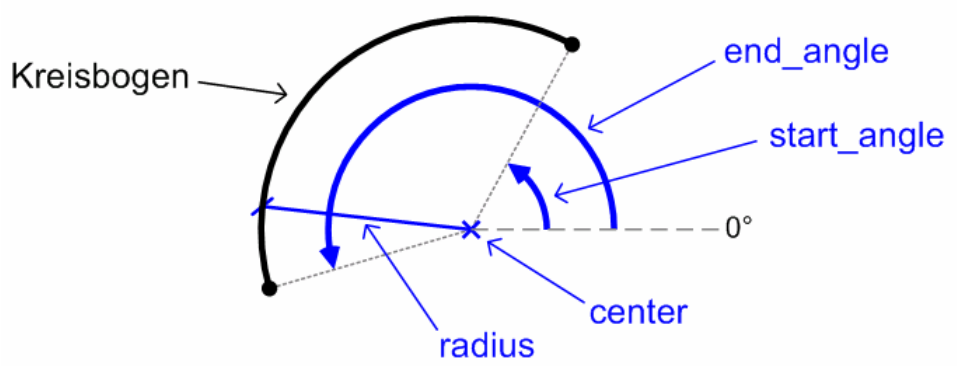
In Abbildung 26 sind die Elemente eines CAD-Plans in UML Notation dargestellt. Die Klasse `Data` stellt die Dokumentwurzel dar. Diese hält jeweils Listen aller Schichten (Klasse `Layer`), Blockdefinitionen (Klasse `BlockDefinition`) und Elemente (von `Element` abgeleitete Klassen) des Plans.

Bei Planelementen wird zwischen einfachen Elementen (abgeleitet von `SimpleElement`) und Blockreferenzen (Klasse `BlockReference`) unterschieden. Diese Unterscheidung ist notwendig, um rekursive Definitionen von Blöcken auszuschließen. Alle Elemente des Plans referenzieren die jeweilige Schicht, der sie zugeordnet wurden (Verbindung zur Klasse `Layer`).

Einfache Planelemente sind, wie bereits erwähnt, Linien (Klasse `Line`), Kreisbögen (Klasse `Arc`) und Kreise (Klasse `Circle`). Aneinanderreihungen von Linien und Kreisbögen werden durch die Klasse `LineArcCompound` repräsentiert. Beschriftungen im Plan werden durch die Elementklasse `Label` dargestellt.

Blockreferenzen (Klasse `BlockReference`) verweisen ihrerseits natürlich auf die instanzierte Blockdefinition (Klasse `BlockDefinition`). Sowohl Blockdefinitionen als auch Blockreferenzen verfügen über eine beliebige Anzahl an Blockattributen (repräsentiert durch Klasse `BlockAttribute`). Die Idee dabei ist, dass unveränderte Attribute eines Blocktyps nur einmal in der Blockdefinition gespeichert werden. Ändert sich der Wert eines Attributs für eine spezielle Blockreferenz, so wird nur dieses Attribut mit dem neuen Wert zur Blockreferenz abgespeichert.

Abschließend wird die Schemadefinition eines Planelements anhand der Klasse `Arc` für Kreisbögen noch genauer betrachtet.




```
1:    <xsd:complexType name="ArcType">
2:      <xsd:complexContent>
3:        <xsd:extension base="ElementType">
4:          <xsd:sequence>
5:            <xsd:element name="center" type="PointSimpleType" />
6:            <xsd:element name="radius" type="xsd:double" />
7:            <xsd:element name="start_angle" type="xsd:double" />
8:            <xsd:element name="end_angle" type="xsd:double" />
9:          </xsd:sequence>
10:       </xsd:extension>
11:     </xsd:complexContent>
12:   </xsd:complexType>
```

Abbildung 27: Geometrische Parameter und Schemadefinition für Kreisbögen

In XML Schemadefinitionen wird prinzipiell zwischen komplexen und einfachen Typen unterschieden. Komplexe Typen können, im Gegensatz zu einfachen Typen, untergeordnete Elemente und Attribute besitzen. Da ein Kreisbogen durch mehrere Parameter spezifiziert wird, handelt es sich um einen komplexen Elementtyp (`xsd:complexType` in Zeile 1).

Die eigentliche Zusammensetzung des Elements aus Teilelementen wird unterhalb des `complexContent` Knotens spezifiziert. Mit Hilfe des `xsd:extension` Tags lässt sich ein bereits zuvor definierter Typ erweitern (Zeile 3). In diesem Fall wird der Typ `ElementType` erweitert. Wie aus dem UML Übersichtsdiagramm (Abbildung 26) ersichtlich, handelt es sich bei `Element` um die Basisklasse für alle im Plan eingezeichneten Elemente. Durch die Erweiterung der Basisklasse wird automatisch ein Element mit der Bezeichnung `layer` geerbt. Mit diesem Element wird die Schicht bezeichnet, der das Element beim Zeichnen in der CAD-Applikation zugeordnet wurde.

Die für die Definition des Kreisbogens notwendigen zusätzlichen Elemente werden unterhalb eines `xsd:sequence` Tags aufgelistet. Es handelt sich dabei um die Elemente `center`, `radius`, `start_angle` und `end_angle`, welche die Geometrie des Kreisbogens vollständig beschreiben (siehe Abbildung 27). Der konkrete Typ eines Elements wird über das Attribut `type` des `xsd:element` Tags festgelegt. Die Definition des Radius und der beiden Winkel erfolgt als Fließkommazahl (`xsd:double`). Alle Winkel im Kontext der XML Schemadefinition für Plandaten werden stets gegen den Uhrzeigersinn in Radiant angegeben. Für die Definition des Kreisbogenmittelpunkts wird der eigens definierte Typ `PointSimpleType` verwendet.

```
1:    <xsd:simpleType name="DoubleListSimpleType">
2:      <xsd:list itemType="xsd:double" />
3:    </xsd:simpleType>
4:
5:    <xsd:simpleType name="PointSimpleType">
6:      <xsd:restriction base="DoubleListSimpleType">
7:        <xsd:length value="2" />
8:      </xsd:restriction>
9:    </xsd:simpleType>
```

Abbildung 28: Definition eines Typs für 2D Punktkoordinaten

Die Definition des Typs für 2D Punktkoordinaten erfolgt in zwei Stufen (Abbildung 28). Zunächst wird ein allgemeiner Typ für Listen von Fließkommazahlen definiert (`DoubleListSimpleType` in Zeile 1). Der eigentliche Typ für Punkte wird dann als Einschränkung dieses allgemeinen Typs spezifiziert. Die Einschränkung, dass die Liste genau aus zwei Elementen, also zwei Fließkommazahlen besteht, wird in Zeile 7 über den `xsd:length` Tag festgelegt.

Die Spezifikationen für die übrigen Elemente des CAD-Plans wurden analog zu jener des Kreisbogens umgesetzt.

3.1.2 Exportieren von Plandaten aus AutoCAD

Das proprietäre Format DWG des Softwareherstellers Autodesk hat sich im Bereich des Bauingenieurwesens mittlerweile als Quasi-Standard für den Austausch von Plänen etabliert. Um an die in den Plänen enthaltenen Daten zu kommen, gibt es mehrere Möglichkeiten:

- Verwendung des kostenpflichtigen SDKs (Software Development Kit, Softwareentwicklungssatz) *RealDWG* des Herstellers Autodesk
- Verwendung des für Ausbildungszwecke frei verfügbaren SDKs *DWGdirect*
- Schreiben eines Exportskripts für eine Applikation, die DWGs laden kann und das Schreiben von Erweiterungen erlaubt

Da das Schreiben eines Exportskripts für AutoCAD die schnellsten Ergebnisse erwarten ließ, wurde dieser Ansatz gewählt. Die schnellste Möglichkeit AutoCAD mit Skripts zu erweitern ist die Verwendung des integrierten *Visual Basic for Applications* (VBA). Es handelt sich dabei um einen Dialekt der Programmiersprache Basic mit dem sich Abläufe in Anwendungssoftware automatisieren lassen.

Die AutoCAD Programmierschnittstelle für VBA erlaubt den Zugriff auf sämtliche Daten eines geöffneten Plans. Für das Erstellen und Validieren von XML Dokumenten sind in VBA ebenfalls bereits Programmierschnittstellen vorhanden.

Die Analyse und Ausgabe der Plandaten erfolgt über einen rekursiven Abstieg in der Dokumentstruktur. Als einfaches Beispiel wird nachfolgend kurz erklärt wie die Liste, der im Plan enthaltenen Schichten, in das XML Dokument eingefügt wird.

```
1:    Private Sub exportLayerList(
2:      xml As MSXML2.DOMDocument60, data_node As IXMLDOMElement)
3:
4:      Dim layer_list_node As IXMLDOMElement
5:      Set layer_list_node = tools.createNode(xml, data_node, "LayerList")
6:
7:      Dim layer_count As Long
8:      layer_count = ThisDrawing.Layers.Count
9:      Dim layer_idx As Long
10:     For layer_idx = 0 To layer_count - 1
11:       Dim layer As AcadLayer
12:       Set layer = ThisDrawing.Layers.Item(layer_idx)
13:       If Not layer.Freeze And layer.LayerOn Then
14:         Dim layer_node As IXMLDOMElement
15:         Set layer_node = tools.createNode(xml, layer_list_node, "Layer")
16:         layer_node.setAttribute "name", layer.name
17:       End If
18:     Next
19:   End Sub
```

Abbildung 29: Prozedur zum Exportieren der Liste aller Schichten des Plans

Der Prozedur wird das XML Dokument, das gerade erstellt wird (Parameter `xml`) und der aktuelle Elternknoten im rekursiven Abstieg (Parameter `data_node`) übergeben. Zunächst wird ein XML Element erzeugt, welches die einzelnen Schichten wiederum als Kinderelemente beinhaltet (`layer_list_node` in Zeile 5).

Der Zugriff auf das aktuell in AutoCAD geöffnete Dokument erfolgt über das Objekt `ThisDrawing`. Die Anzahl der Schichten des Plans wird zunächst in Zeile 8 ausgelesen. Der Zugriff auf eine ausgewählte Schicht erfolgt über die Methode `Item` des Objekts `Layers` der aktuellen Zeichnung (Zeile 12). Das Exportskript ist so implementiert, dass in der Zeichnung deaktivierte Schichten nicht exportiert werden (Abfrage in Zeile 13).

Schließlich wird für jede zu exportierende Schicht ein XML Element mit dem `name`-Attribut erstellt, welches den Namen der Schicht enthält (Zeilen 15-16).

```
1:    <?xml version="1.0" encoding="UTF-8"?>
2:    <Data>
3:     <LayerList>
4:      <Layer name="0"/>
5:     </LayerList>
6:     <ElementList>
7:      <Arc layer="0">
8:       <center>1649.22330620775 1082.53370275665</center>
9:       <radius>380.020951432923</radius>
10:      <start_angle>0.445093846001796</start_angle>
11:      <end_angle>3.82376093524188</end_angle>
12:     </Arc>
13:    </ElementList>
14:   </Data>
```

Abbildung 30: Einfaches Beispiel für die Ausgabe des Plandaten Exportskripts

In Abbildung 30 ist ein Beispiel für die Ausgabe des Exportskripts zu sehen. Der sehr einfache Beispielplan enthält nur eine Schicht (Zeile 4), in der ein Kreisbogen eingezeichnet wurde (Zeile 7). Die Struktur des XML Dokuments entspricht der Struktur wie sie mit dem UML Diagramm in Abbildung 26 festgelegt wurde.

3.2 Robustheitsprobleme bei geometrischen Berechnungen

Die in den nächsten beiden Abschnitten beschriebenen Vorgehensweisen zur Bereinigung der Geometrie (Kapitel 3.3) und Analyse der Flächeninformationen (Kapitel 3.4) verwenden verschiedenste geometrische Operationen.

Grundsätzlich kann man bei diesen geometrischen Operationen zwischen Konstruktionen und Prädikaten unterscheiden. Ein Beispiel für eine Konstruktion ist die Berechnung des Schnittpunkts zweier Liniensegmente. Prädikate wiederum können verwendet werden um Fragen bezüglich der Lage zweier Körper zueinander zu beantworten. Ein typisches Beispiel für ein Prädikat ist die Fragestellung, ob drei gegebene Punkte auf einer Geraden liegen.

Sind geometrische Operationen mit ungenauer Fließkommaarithmetik (z.B. nach IEEE 784) umgesetzt, kann dies zu falschen Ergebnissen führen. Ein Schnittpunkt zweier Liniensegmente liegt dann nicht zwangsläufig auf beiden von den Liniensegmenten aufgespannten Geraden. Dies ist durch Rundungsfehler begründet, die bei den mathematischen Berechnungen auftreten.

Für sämtliche geometrischen Operationen kommt in weiterer Folge die Geometriebibliothek CGAL zum Einsatz. Da exakte Konstruktionen und Prädikate inklusive darauf aufbauende Algorithmen integraler Bestandteil von CGAL sind, soll nachfolgend kurz auf die Problematik und mögliche Problemlösungen eingegangen werden (aus [KMP+08]).

Als problematisches Beispiel wird ein Prädikat zur Ermittlung der Orientierung dreier Punkte herangezogen. Für zwei Punkte p und q klassifiziert das Prädikat einen dritten Punkt r als links liegend (+1), rechts liegend (-1), oder auf (0) der von den ersten zwei Punkten aufgespannten Geraden. Die Auswertung des Prädikats erfolgt nach folgender Formel:

$$orientation(p,q,r) = sign((q_x - p_x)(r_y - p_y) - (q_y - p_y)(r_x - p_x)) \quad (2)$$

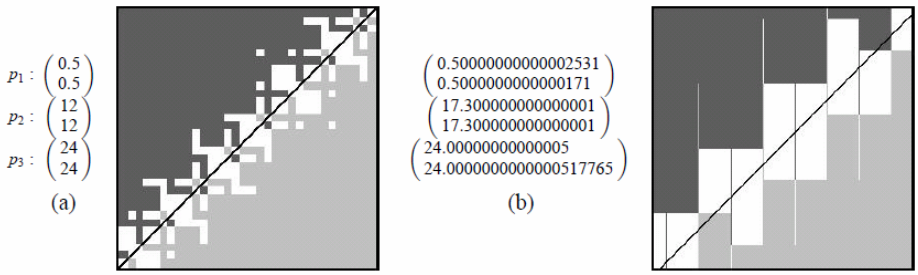


Abbildung 31: Fehler beim Auswerten des Orientierungsprädikats (aus [KMP+08])

Für Abbildung 31 wurde das Prädikat für 256x256 Nachbarpunkte von p_1 ausgewertet. Der Begriff Nachbarpunkt ist im Sinne der kleinsten repräsentierbaren Änderung der Koordinaten zu verstehen. Die exakte Verbindungsgerade zwischen p_2 und p_3 ist durch eine Linie angedeutet. Laut Prädikat sind links liegende Punkte dunkelgrau, rechts

liegende Punkte hellgrau und kollineare Punkte weiß dargestellt.

Aus dem Ergebnisbild ist sofort ersichtlich, dass Punkte aufgrund von Rundungsfehlern falsch klassifiziert werden. Prinzipiell gibt es drei Möglichkeiten mit dieser Problematik umzugehen:

1. Man stellt sicher, dass Prädikate und Konstruktionen immer exakte Ergebnisse liefern
2. Man erweitert Algorithmen (die ungenaue Prädikate und Konstruktionen verwenden) um die Behandlung von Spezialfällen, wobei fehlerhafte Berechnungen erkannt und korrigiert werden
3. Man transformiert die Eingangsdaten in der Art, dass die ungenauen Berechnungen im Endeffekt die erwünschten korrekten Resultate liefern

Für die in weiterer Folge verwendete Geometriebibliothek CGAL wurde der erste Ansatz gewählt. Zahlen werden dabei nicht durch eine herkömmliche Fließkommazahl sondern durch eine alternative Darstellung repräsentiert, mit der sich Berechnungen ohne Rundungsfehler durchführen lassen.

3.3 Bereinigung der Geometrie

Im ersten Verarbeitungsschritt werden die Daten aus den CAD-Plänen bereinigt. Im Wesentlichen geht es darum, Fehler und Ungenauigkeiten zu korrigieren, die bei der Konstruktion der Pläne jederzeit passieren können.

Auftretende Fehler und mögliche Korrekturmaßnahmen wurden bereits bei der Diskussion des BMG (Kapitel 2.1.2.1: Bereinigung der Geometrie) und des AUG (Kapitel 2.1.3.1: Bildung eines Objektgraphen aus den Plandaten) besprochen. Die im Rahmen dieser Arbeit implementierten Korrekturmaßnahmen basieren größtenteils auf diesen Arbeiten.

In den nachfolgenden Abschnitten werden die einzelnen Schritte zur Korrektur der Plandaten im Detail erläutert.

3.3.1 Ungerichteter Graph als Datenstruktur

Zunächst stellt sich die Frage, welche Datenstruktur zur Speicherung der Plandaten in der Bereinigungsphase herangezogen werden soll.

Prinzipiell lassen sich alle geometrischen Elemente des Plans als Menge von Liniensegmenten darstellen. Für die Umsetzung der Korrekturmaßnahmen ist es außerdem wichtig, dass Anfragen bezüglich örtlicher Nachbarschaft von Elementen effizient beantwortet werden können.

Für die Repräsentation der Liniensegmente und deren Zusammenhänge wurde die Datenstruktur eines ungerichteten Graphen gewählt.

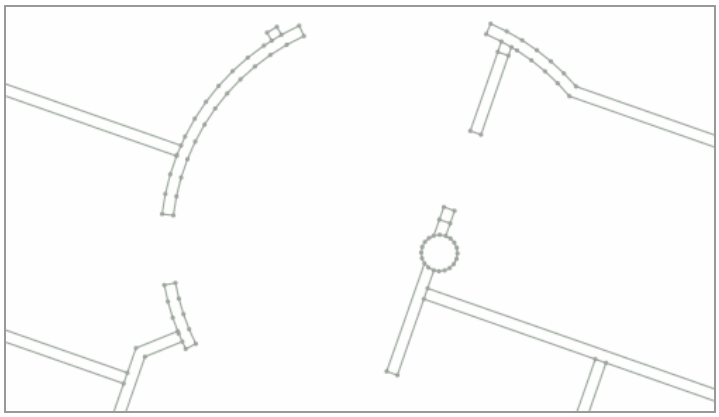


Abbildung 32: Repräsentation der Plandaten als ungerichteter Graph

Ein ungerichteter Graph besteht aus einer Menge V von Knoten und einer Menge E von Kanten:

$$V = \{v_1, v_2, ..., v_{|V|}\}$$
$$E = \{e_1, e_2, ..., e_{|E|}\} \quad (3)$$

Eine Kante wiederum wird durch ein ungeordnetes Paar von Knoten bestimmt:

$$e = \{v_i, v_j\} = \{v_j, v_i\} \quad (4)$$

In Abbildung 32 wird ein Auszug eines CAD-Plans in der Graphenrepräsentation dargestellt. Knoten sind als Punkte erkennbar, die wiederum durch Liniensegmente (die Kanten des Graphen) verbunden sind. Ebenfalls ersichtlich ist die Transformation kreisförmiger Geometrien zu einer Anordnung von Liniensegmenten.

Um geometrische Nachbarschaftsabfragen effizient beantworten zu können, muss die dem Graphen zugrunde liegende Punktmenge in einer speziellen Datenstruktur repräsentiert sein. Für die Punktmengenrepräsentation wurden zwei unterschiedliche Ansätze umgesetzt:

- Der erste Ansatz verwendet eine spezielle Art von Triangulierung (Delaunay Triangulierung)
- Der zweite Ansatz verwendet eine spezielle Art von Suchbaum (k-d Baum)

Der wesentliche geometrische Algorithmus zur Feststellung von Nachbarschaften ist das Ermitteln jener Punkte einer Punktmenge, die sich innerhalb eines gewissen Abstandes zu einer Koordinate befinden. Aufbauend darauf kann beispielsweise auch die Suche nach Punkten innerhalb eines rechteckigen Bereichs implementiert werden.

Wie diese Abfrage in den verwendeten Datenstrukturen realisiert ist, wird in den nachfolgenden Kapiteln 3.3.1.1 „Delaunay Triangulierung als Punktmengenrepräsentation“ und 3.3.1.2 „k-d Baum als Punktmengenrepräsentation“ genauer betrachtet.

Ein Vergleich der beiden Ansätze inklusive Diskussion findet sich abschließend in Kapitel 3.3.3 „Diskussion der Operationen“.

3.3.1.1 Delaunay Triangulierung als Punktmengenrepräsentation

Der russische Mathematiker Boris Nikolajewitsch Delone setzte sich Mitte der 30er Jahre in einer Veröffentlichung mit Verfahren auseinander die Dreiecksnetze aus Punktmengen erzeugen. Dabei beschrieb er ein gebräuchliches Verfahren, das als Delaunay Triangulierung bekannt ist. Die Triangulierung einer Punktmenge *P* in der Ebene ist die Einteilung der konvexen Hülle von *P* in dreieckige Teilflächen.

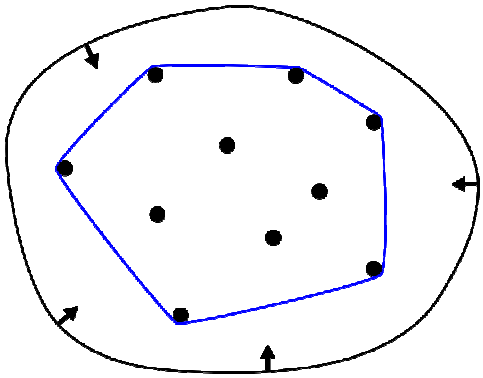


Abbildung 33: Konvexe Hülle - Analogie des elastischen Bandes (aus [Wika])

Der Begriff der konvexen Hülle lässt sich am besten durch die Analogie des elastischen Bandes erklären: Würde man für jeden Punkt aus *P* einen Nagel in ein Brett schlagen und dann ein elastisches Band um alle diese Nägel spannen, würde das Band den Verlauf der konvexen Hülle annehmen (Abbildung 33).

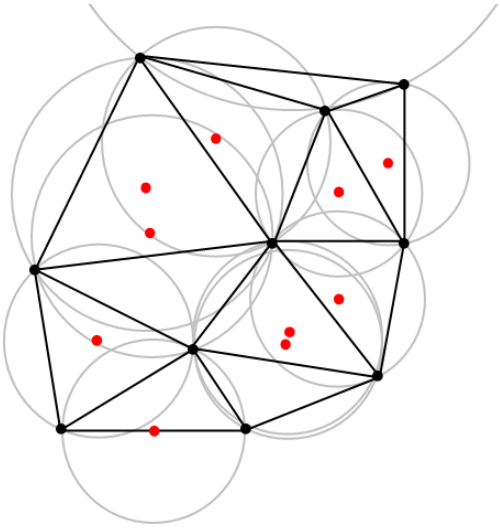


Abbildung 34: Delaunay Triangulierung mit Umkreisen der Dreiecke (aus [Wikb])

Die Aufteilung in Dreiecke muss so erfolgen, dass die Eckpunkte der Dreiecke Punkten aus *P* entsprechen. Eine Delaunay Triangulierung ist eine Triangulierung, bei der die Dreiecke folgende Bedingung erfüllen müssen:

Bildet man den Umkreis für ein Dreieck, so darf sich kein weiterer Punkt aus P innerhalb dieses Umkreises befinden (Abbildung 34).

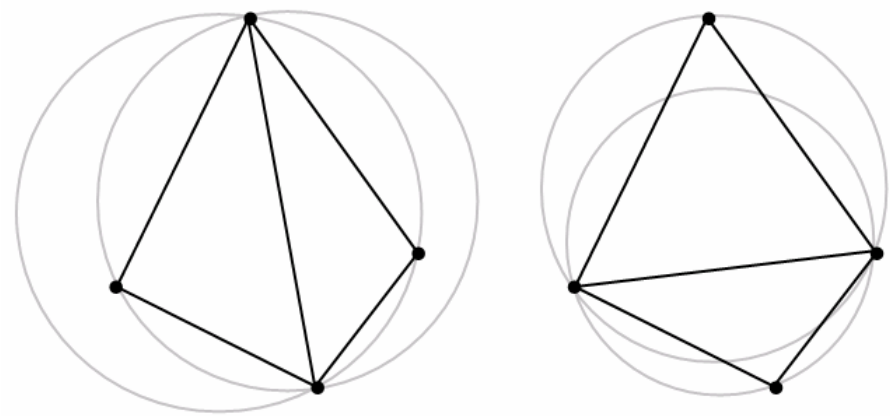


Abbildung 35: Kantentausch zur Erfüllung der Umkreisbedingung (aus [Wikb])

Ist die Umkreisbedingung für zwei benachbarte Dreiecke nicht erfüllt, so muss ein so genannter Kantentausch durchgeführt werden. Nach dem Kantentausch ist die Umkreisbedingung stets für beide Dreiecke erfüllt (Abbildung 35).

Für den Fall, dass sich vier oder mehr Punkte aus *P* exakt auf einem Kreis befinden, ist die Delaunay Triangulierung nicht eindeutig definiert.

Die Suche nach den k-nächsten Nachbarn eines Knotens im Dreiecksnetz erfolgt mit Hilfe einer Prioritätenliste. Die Sortierung der Listenelemente erfolgt nach der Entfernung zum Ausgangsknoten.

Zunächst wird nur der Ausgangsknoten mit Entfernung null in die Prioritätenliste

eingefügt und markiert. In einer Schleife wird der nach Prioritätenliste nächstgelegene Knoten ausgewählt und aus der Liste entfernt. Ist die Entfernung zu diesem Knoten größer als die spezifizierte Maximalentfernung, wird der Suchvorgang abgebrochen. Handelt es sich bei dem gewählten Knoten nicht um den Ausgangsknoten, wird er der Ergebnisliste hinzugefügt. Alle nicht markierten Nachbarknoten des gewählten Knotens, werden in die Prioritätenliste eingefügt und markiert. Ist die Prioritätenliste leer, wird der Vorgang ebenfalls abgebrochen. Eine detaillierte Beschreibung des Algorithmus inklusive Korrektheitsbeweis befindet sich in Kapitel 10.6 von [MN99].

In CGAL ist diese Art von Punktmengenrepräsentation in der Klasse `CGAL::Point_set_2` umgesetzt. Die Suche nach Punkten innerhalb eines rechteckigen Bereichs kann mit der Methode `range_search` dieser Klasse durchgeführt werden.

3.3.1.2 k-d Baum als Punktmengenrepräsentation

Der k-d Baum wurde von Jon Luis Bentley Mitte der 70er Jahre als Verallgemeinerung eines binären Suchbaumes für mehrdimensionale Suchräume konzipiert. Der Suchraum wird dabei hierarchisch in rechteckige Teilbereiche geteilt. Jedes dieser Rechtecke ist durch *k* Intervalle auf den Koordinatenachsen des Suchraums bestimmt ([Ben75]).

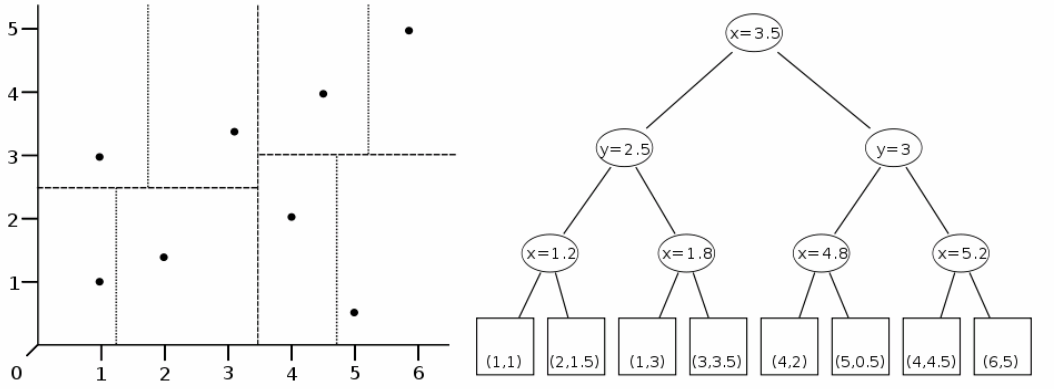


Abbildung 36: Aufteilung des Suchraums im k-d Baum (aus [Wikc])

Für jeden Knoten des Suchbaums wird der Suchraum mit einer Hyperebene in zwei Teile geteilt. Im Falle eines zweidimensionalen Suchraums handelt es sich dabei um eine Gerade in der Ebene. Der konkrete Verlauf der Hyperebene kann nach verschiedenen Regeln festgelegt werden. Im Beispiel aus Abbildung 36 wurden abwechselnd horizontale und vertikale Trenngeraden verwendet.

Die rekursive Teilung des Suchraums wird solange fortgesetzt, bis die Anzahl der Elemente in einem Teilbereich unter einen gewissen Grenzwert fällt. In den so entstehenden Blättern des Suchbaums werden letztendlich die Elemente der Suchmenge gespeichert.

Der Algorithmus zur Bereichssuche führt einen rekursiven Abstieg im Suchbaum durch. Als Startknoten dient stets der Wurzelknoten des binären Suchbaums. In jeder Ebene des Baumes wird entschieden mit welchem Nachfolgeknoten der Algorithmus fortfahren soll.

Überschneidet sich der übergebene Suchbereich mit der teilenden Hyperebene eines Knotens, müssen beide Nachfolgeknoten untersucht werden. Gelangt der Suchalgorithmus an ein Blatt des Baumes, werden alle dort gespeicherten Punkte, die sich innerhalb des Suchbereichs befinden, in einer Prioritätenliste gespeichert. Die Sortierung der Prioritätenliste erfolgt nach Entfernung zum Zentrum des Suchbereichs. Eine detailliertere Betrachtung des Algorithmus findet man in [FBF77].

In CGAL ist der k-d Baum in der Klasse `CGAL::Kd_tree` implementiert. Die Bereichssuche wird mit der Methode `search` dieser Klasse durchgeführt. Zur Suche nach Punkten innerhalb eines rechteckigen Bereichs wird dieser Methode eine Instanz der Klasse `CGAL::Fuzzy_iso_box` übergeben.

3.3.2 Operationen

Nachfolgend sind alle Operationen zur Bereinigung der Plandaten beschrieben. Sinn der Operationen ist es in erster Linie Konstruktionsfehler im Plan zu korrigieren. Topologische Mehrdeutigkeiten müssen ebenfalls eliminiert werden, um die Daten auf die nachfolgende Flächenanalyse vorzubereiten.

Die Operationen sind vollständig mit der beschriebenen Datenstruktur des ungerichteten Graphen umgesetzt. Die meisten Operationen verlassen sich dabei auf die Tatsache, dass der ungerichtete Graph effizient auf Nachbarschaften untersucht werden kann.

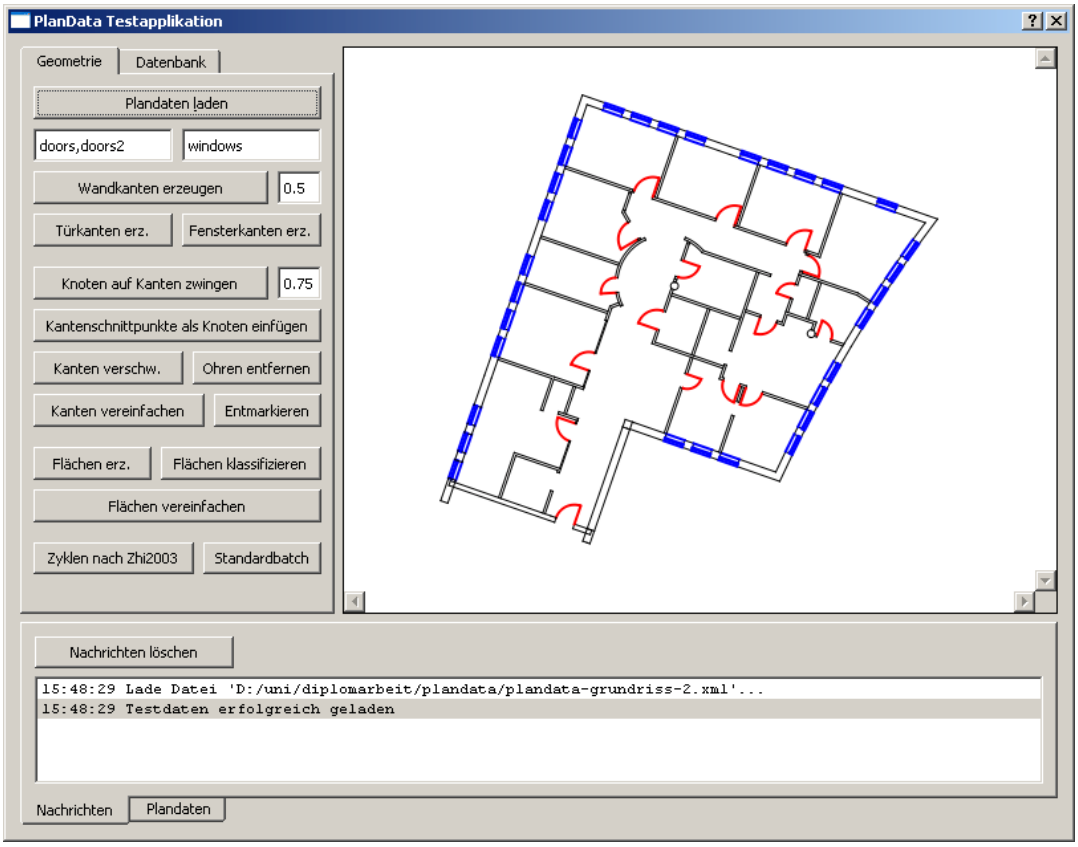


Abbildung 37: Benutzeroberfläche zur Anwendung der Operationen auf Plandaten

Die Anwendung, der Operationen erfolgt interaktiv mit Hilfe einer grafischen Benutzeroberfläche. Neben dem Ausführen der Operationen bietet die Applikation auch visuelles Feedback zum Resultat der Operationen.

Dieselbe Applikation wurde auch verwendet um in der Implementierungsphase Fehler in den Algorithmen zu beheben (Abbildung 37).

3.3.2.1 Kanten aus Plandaten erzeugen

Die erste Operation sorgt dafür, dass die Daten des Plans als Kanten in den ungerichteten Graphen eingefügt werden. Ausgegangen wird immer von XML Dokumenten, in denen die Daten nach Plandatenschema (Kapitel 3.1.1: XML Schemadefinition für Plandaten) abgelegt sind.

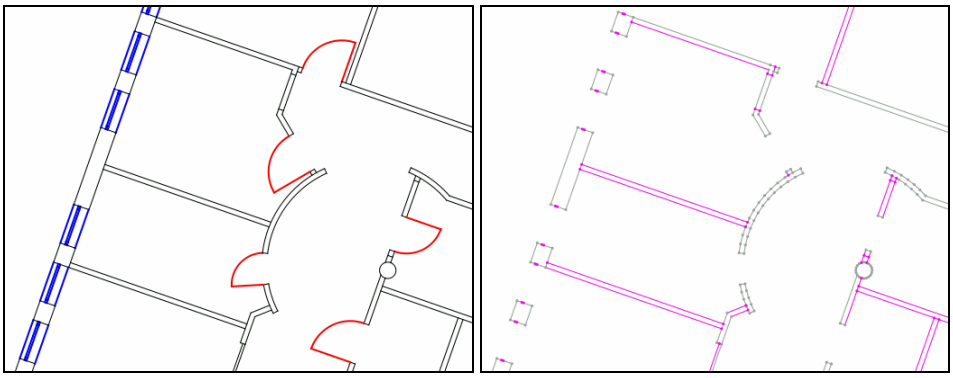


Abbildung 38: Erzeugung von Kanten des ungerichteten Graphen aus Plandaten

Das linke Bild in Abbildung 38 zeigt die Rohdaten aus dem XML Dokument. Die Farbgebung ergibt sich dabei aus der Zuordnung der Schichten des Plans zu den Objekttypen Tür (rot eingefärbt) und Fenster (blau eingefärbt). Die Zuordnung wird vor dem Import der Daten vom Benutzer durchgeführt.

Die Operation wandelt die Elemente des Plans in Kanten um und fügt diese in den ungerichteten Graphen ein. Im Beispiel aus Abbildung 38 wurden nur Elemente umgewandelt, die weder als Tür noch als Fenster klassifiziert wurden. Ähnlich wie beim AUG werden jene Knoten beim Einfügen zusammengefasst, deren Abstand zueinander geringer als ein bestimmter Schwellwert ist (vergleiche Kapitel 2.1.3.1).

In der Darstellung des ungerichteten Graphen werden Knoten mit genau einer anliegenden Kante farblich hervorgehoben. Der Grund dafür ist, dass diese Knoten meist auf Konstruktionsfehler im Plan hindeuten. Diese Fehler müssen durch die Anwendung weiterer Operationen behoben werden.

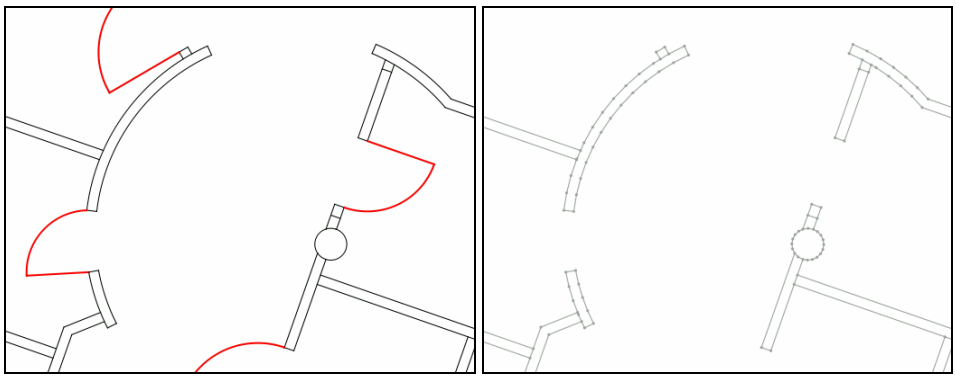


Abbildung 39: Transformation von Kreisen und Kreisbögen in Kanten des Graphen

Sind in den Plandaten Kreise oder Kreisbögen enthalten, müssen diese durch Kanten im ungerichteten Graphen approximiert werden. Die Anzahl der Teilstücke in die ein Kreisbogen oder Kreis aufgeteilt wird, muss eine Funktion der Elementgröße sein.

Bei Tests hat sich gezeigt, dass diese Anzahl mit zunehmendem Radius des Elements erhöht werden muss. Ansonsten sind Elemente mit kleinem Radius zu hoch und Elemente mit großem Radius zu gering aufgelöst. Die Anzahl der Punkte *c* der Diskretisierung wird deshalb nach folgender Formel berechnet:

$$c = \frac{\alpha}{2\pi}\sqrt{\frac{r}{2e}} \tag{5}$$

In die Formel fließen mit α die Bogenlänge des Kreisbogens, mit r der Radius des Kreisbogens und mit e ein Parameter für den Fehler in der Approximation ein.

3.3.2.2 Knoten auf nahe liegende Kanten zwingen

Diese Operation stellt das zentrale Werkzeug zur Korrektur topologischer Fehler in den Plandaten dar. Eine Analyse, der im Graphen vorkommenden Flächen, kann nur gelingen, wenn die topologischen Zusammenhänge korrekt repräsentiert sind.

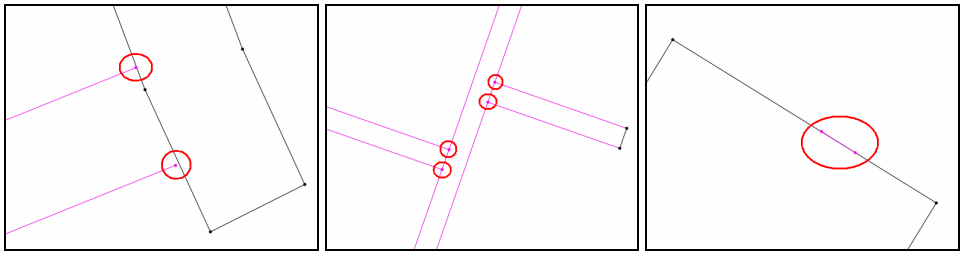


Abbildung 40: Beispiele für topologische Fehler

In Abbildung 40 sind drei häufig vorkommende topologische Fehler in CAD-Plänen dargestellt. Aufgrund von Ungenauigkeiten bei der Konstruktion schließen Linien oft nicht korrekt an andere Linien an (linkes Bild).

Linien, die aneinander anschließen, werden in CAD-Plänen an den Berührungspunkten oft nicht geteilt, da dies für den eigentlichen Zweck des Plans nicht notwendig ist (mittleres Bild).

Bei Konstruktionen bleiben oft kurze Linien als Überreste, welche eine korrekte topologische Analyse verhindern (rechtes Bild).

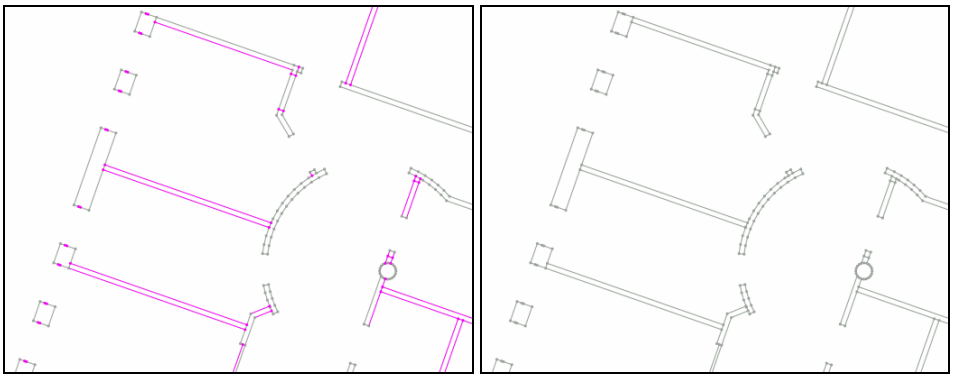


Abbildung 41: Zwingen der Knoten des Graphen auf nahe liegende Kanten

Wie bereits erwähnt, werden in der Darstellung Knoten mit Grad eins inklusive anliegende Kante farblich gekennzeichnet. Werden Knoten auf nahe liegende Kanten gezwungen, sieht man sofort, dass diese Knoten zu Knoten höheren Grades werden (Abbildung 41).

Es folgt nun eine schrittweise Beschreibung des Algorithmus mit dem Knoten auf nahe liegende Kanten gezwungen werden können.

Im ersten Schritt werden für alle Kanten des Graphen nahe liegende Knoten gesucht. Als Parameter fließt in diese Suche der mit ε benannte Maximalabstand zur Kante ein.

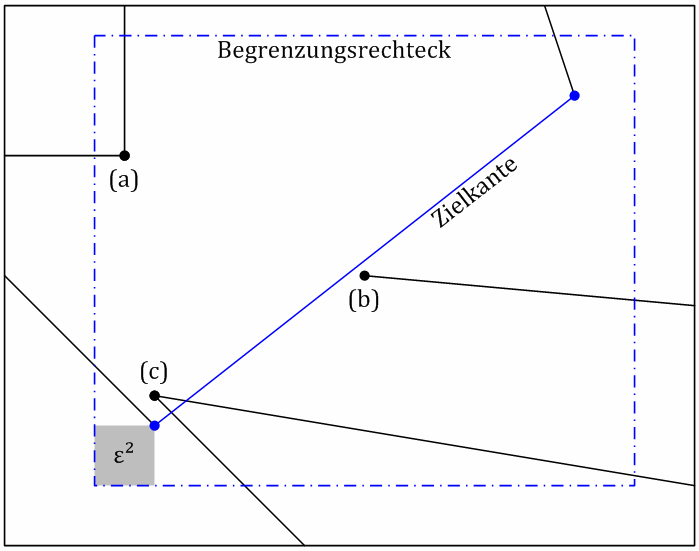


Abbildung 42: Suche nach Knoten die sich in der Nähe von Kanten befinden

Zur Umsetzung der Suche wird um jede Kante ein achsenparalleles Begrenzungsrechteck gelegt. Dieses Rechteck wird entlang der Achsen in alle Richtungen um ε vergrößert (blaues Rechteck in Abbildung 42).

Über die dem ungerichteten Graphen zugrunde liegende Punktmenge wird nun nach allen Knoten durchsucht, die sich innerhalb dieses Rechtecks befinden. Aus dieser Knotenmenge müssen in einem zweiten Schritt jene verworfen werden, deren Abstand zur Kante größer als ε ist (z.B. Knoten (a) aus Abbildung 42).

Aus Abbildung 42 ist auch ersichtlich, dass keineswegs nur Knoten mit Grad eins

Kandidaten für diese Operation sind. Oft müssen auch Knoten höheren Grades auf eine nahe liegende Kante gezwungen werden (Knoten (c) aus Abbildung 42).

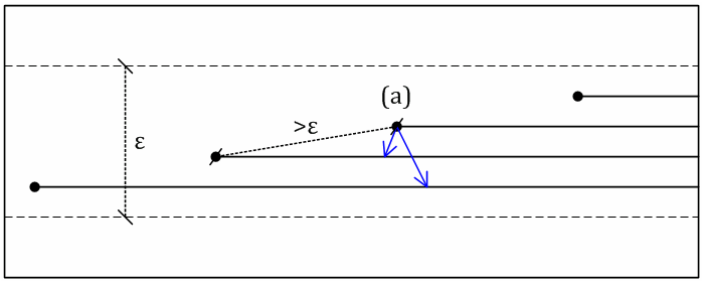


Abbildung 43: Knoten innerhalb der ε-Umgebung zweier Kanten

Resultat der Suche ist eine Zuordnung von Knoten zu einer Liste von nahe liegenden Kanten. Eine Liste ist deswegen notwendig, weil sich Knoten in manchen Fällen in der ε-Umgebung mehrerer Kanten befinden (z.B. Knoten (a) in Abbildung 43).

Dieser Fall tritt vorwiegend auf, wenn aufgrund von Konstruktionsfehlern mehrere längere Liniensegmente übereinander liegen. Würde der Knoten nur auf die nächstgelegene Kante gezwungen werden, wären mehrere Durchläufe des Algorithmus erforderlich. So können alle unerwünschten Nachbarschaften in einem Durchlauf eliminiert werden.

Im nächsten Schritt muss jeder gefundene Knoten auf alle nahe liegenden Kanten gezwungen werden. Dazu muss zunächst ein Zielpunkt definiert werden, an den der Knoten verschoben wird.

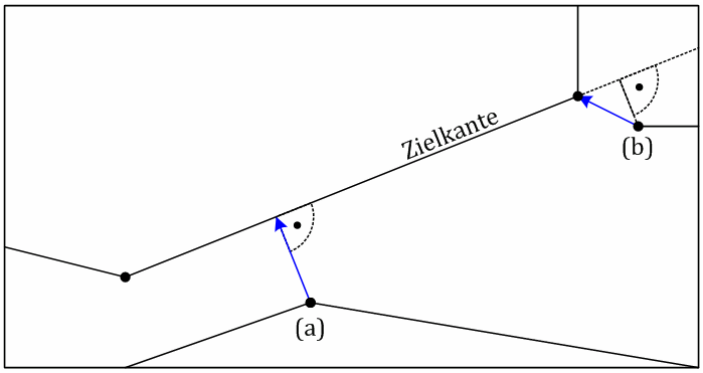


Abbildung 44: Konstruktion eines Zielpunkts für den Knoten

Wie aus Abbildung 44 ersichtlich, wird der Zielpunkt durch Schnitt der Zielkante mit einer auf die Zielkante normalen Gerade durch den Knoten ermittelt (Knoten (a) in Abbildung 44). Kann durch diese Konstruktion kein Schnittpunkt gefunden werden, wird der nähere Endpunkt der Zielkante als Zielpunkt festgelegt (Knoten (b) in Abbildung 44).

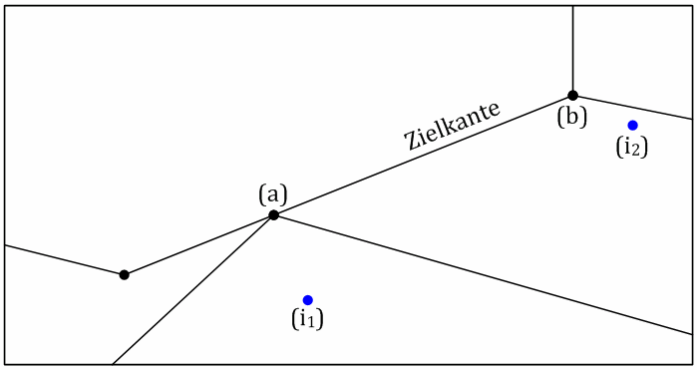


Abbildung 45: Teilen der Zielkante und Verschieben des Knotens

Ist an der Zielposition noch kein Knoten vorhanden, wird ein neuer Knoten in den Graphen eingefügt (Knoten (a) in Abbildung 45). Liegt der neue Knoten auf der Zielkante, wird diese am Knoten geteilt (ebenfalls Knoten (a) in Abbildung 45).

Wird eine Kante aufgeteilt, die ihrerseits Zielkante für einen anderen Knoten ist (z.B. Knoten (b) aus Abbildung 45) muss die Zielkante für diesen Knoten aktualisiert werden. Die neue Zielkante ist dann die nähere der beiden, durch die Aufteilung entstandenen, neuen Teilkanten.

In einem zweiten Schritt müssen noch alle, am bearbeiteten Knoten anliegenden Kanten auf den neuen Knoten umgehängt werden. Isolierte Knoten, die bei diesem Vorgang entstehen, werden nach Durchlauf des Algorithmus aus dem Graphen entfernt (i_1 und i_2 in Abbildung 45).

3.3.2.3 Kanten vereinfachen

Ein relativ einfacher Algorithmus dient dazu, überflüssige Knoten aus dem Graphen zu entfernen. Ein Knoten der zwei parallele, kollineare Kanten trennt, wird als überflüssig betrachtet.

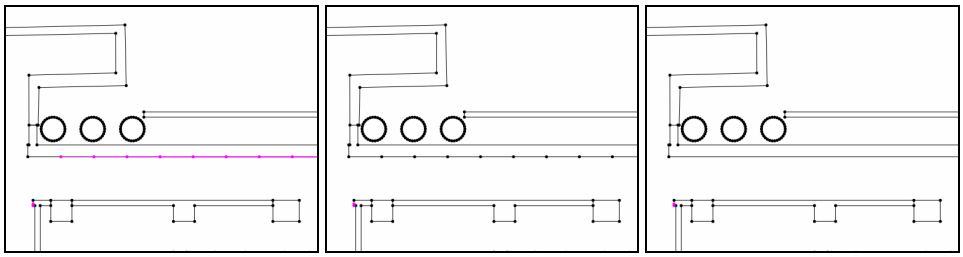


Abbildung 46: Vereinfachung mehrerer übereinander liegender Kanten

Im Beispiel aus Abbildung 46 wurde eine Kante achtmal, leicht verschoben, eingezeichnet (linkes Bild). Das mittlere Bild wurde aufgenommen nachdem Knoten auf nahe liegende Kanten gezwungen wurden. Im rechten Bild wurde die Operation zum Vereinfachen der Kanten angewendet. Man sieht, dass die überflüssigen Knoten aus dem ungerichteten Graphen entfernt wurden.

3.3.2.4 Kantenschnitte entfernen

Neben Ungenauigkeiten bei der Konstruktion, sind sich schneidende Kanten das zweite große Problem bei der Repräsentation der topologischen Zusammenhänge im Graphen.

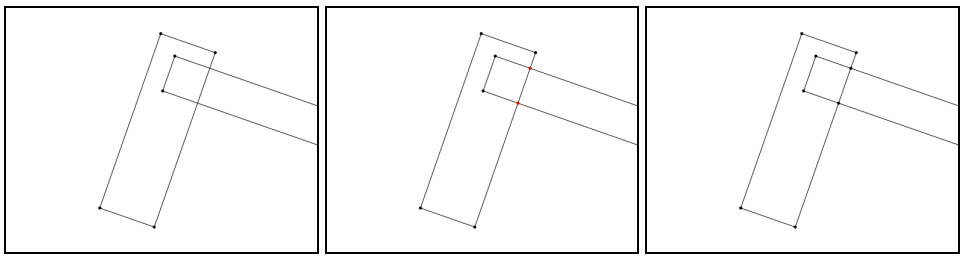


Abbildung 47: Entfernen von Kantenschnitten durch Einfügen isolierter Knoten

Gelöst wird das Problem indem zunächst mit Hilfe eines Sweep-Line Verfahrens alle Schnittpunkte der Kanten des Graphen berechnet werden. An den Schnittpunkten werden isolierte Knoten in den Graphen eingefügt (mittleres Bild in Abbildung 47). In einem zweiten Schritt werden die Kanten an diesen Knoten aufgeteilt. Dieses Aufteilen der Kanten wird mit der bereits beschriebenen Operation zum Zwingen von Knoten auf nahe liegende Kanten durchgeführt (rechtes Bild in Abbildung 47).

Der Sweep-Line Algorithmus zur Bestimmung der Schnittpunkte ist in CGAL in der Funktion `CGAL::compute_intersection_points` implementiert. Eine effiziente Implementierung eines Sweep-Line Algorithmus wird beispielsweise in [BMN97] beschrieben.

3.3.2.5 Kanten verschweißen

Sind in einem CAD-Plan aufgrund von Ungenauigkeiten bei der Konstruktion größere Lücken vorhanden, wird versucht diese mit Hilfe einer weiteren Operation zu füllen.

Diese Operation zum Verschweißen von Kanten ist eine Implementierung der Vorgehensweise, die bereits in Kapitel 2.1.2.1 besprochen wurde.

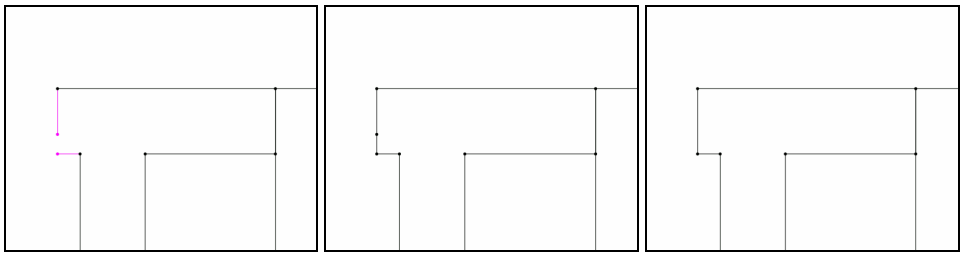


Abbildung 48: Füllen von Löchern durch Verschweißen von Kanten

In Abbildung 48 ist ein Beispiel für einen relevanten Konstruktionsfehler im Plan zu sehen. Für das Verschweißen von Kanten sind nur Knoten vom Grad eins relevant (linkes Bild). Beim Verschweißen wird die Lücke im Plan korrekt geschlossen (mittleres Bild). Meistens kann die Geometrie noch durch das Vereinfachen von Kanten weiter verbessert werden (rechtes Bild).

3.3.2.6 Spezialkanten aus Plansymbolen erzeugen

Eine sehr wichtige Operation ersetzt abstrakte Symbole im Plan durch entsprechende Spezialkanten im ungerichteten Graphen (vergleiche Kapitel 2.1.1.4: Symbolerkennung und Kapitel 2.1.2.1: Bereinigung der Geometrie). Spezialkanten werden in weiterer Folge verwendet um die vom Graphen aufgespannten Flächen zu klassifizieren.

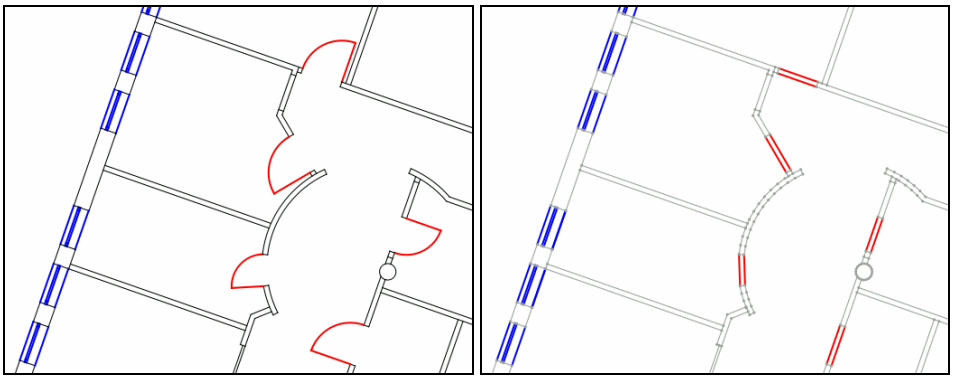


Abbildung 49: Transformation von Plansymbolen in Spezialkanten

Die Transformation von Türsymbolen zu zwei parallelen Spezialkanten ist in Abbildung 49 dargestellt.

Fenstersymbole entsprechen meist bereits der eigentlichen Geometrie der Fenster und können so nahezu unverändert in Spezialkanten umgewandelt werden. Der nachfolgend beschriebene Algorithmus wird sowohl zur Transformation von Fenstersymbolen als auch zur Transformation von Türsymbolen eingesetzt.

Im ersten Schritt werden Planelemente, die einem bestimmten Symboltyp zugeordnet wurden, in Kanten eines ungerichteten Graphen transformiert (Operation „Kanten aus Plandaten erzeugen“). Die Geometrie im resultierenden Graphen wird dann mit den Operationen „Knoten auf nahe liegende Kanten zwingen“ und „Kantenschnitte entfernen“ bereinigt.

Um einzelne Symbole zu isolieren, werden dann alle Zusammenhangskomponenten des Graphen gebildet. Jede dieser Komponenten entspricht jeweils einem Symbol des Plans.

Im nächsten Schritt wird einmal über alle diese Zusammenhangskomponenten iteriert. Die Punkte jeweils einer Zusammenhangskomponente werden dann als Knoten in einen neuen ungerichteten Graphen eingefügt.

Mit dem bereits vorgestellten Verfahren zur Ermittlung von kantennahen Knoten (siehe Abbildung 42 in Kapitel 3.3.2.2) werden jene Knoten von Grad eins, oder Grad zwei ermittelt, die sich in der Nähe von Wandkanten des ursprünglichen Graphen befinden.

Werden für ein Symbol zwei solcher kantennaher Knoten gefunden, werden diese durch eine Spezialkante verbunden. Von den beiden Knoten ausgehend wird nun versucht zwei ungefähr gleich lange, normal auf die Spezialkante stehende Kanten zu finden. Ist die Suche erfolgreich, werden die Endpunkte dieser gefundenen Kanten ebenfalls durch eine Spezialkante verbunden. So kann die zweite parallele Spezialkante eines Symbols automatisch gefunden werden.

Werden bei der Suche nach kantennahen Knoten bereits vier Knoten gefunden, so wird sofort versucht zwei Spezialkanten einzufügen. Zunächst werden zufällig drei der vier Knoten ausgewählt. Einer dieser drei Knoten wird mit jenem der beiden anderen Knoten verbunden, der weiter von diesem entfernt liegt. Schneidet sich dieses gerichtete Segment mit dem gerichteten Segment, das von den beiden anderen Knoten in einer beliebig wählbaren Reihenfolge aufgespannt wird, so müssen die Endpunkte der Segmente vertauscht werden.

Um diese Vertauschung korrekt vornehmen zu können, muss mit Hilfe des Innenprodukts die Richtung der Segmente zueinander bestimmt werden. Ist das Innenprodukt (also die Länge der Normalprojektion des einen Segmentvektors auf den anderen) größer Null, so sind die Vektoren gleich gerichtet (der eingeschlossene Winkel ist kleiner als 90°). In diesem Fall reicht es die beiden Anfangs- oder Endpunkte der gerichteten Segmente zu vertauschen. Liefert das Innenprodukt einen negativen Wert, sind die Segmente ungleich gerichtet und der Anfangspunkt des einen Segments muss mit dem Endpunkt des anderen Segments vertauscht werden.

Stehen die Verbindungssegmente in etwa parallel zueinander und ist der Abstand größer als ein festgelegter Schwellwert, werden diese als Spezialkanten in den ungerichteten Graphen eingefügt.

3.3.3 Diskussion der Operationen

Vergleicht man die Laufzeiten der Operationen bei unterschiedlichen Repräsentationen für die Punktmengen zeigt sich, dass der k-d Baum wesentlich effizienter ist. Der Grund dafür liegt in der Tatsache, dass für den k-d Baum nicht mit exakten Zahlentypen gerechnet werden muss. Die Delaunay Triangulierung hingegen liefert nur dann sicher korrekte Ergebnisse, wenn Konstruktionen und Prädikate exakt ausgewertet werden.

Plan	|E|	|V|	t1 [s]	t2 [s]	Faktor
grundriss-2.xml	559	775	1,2	1,5	1,2
wmitest.xml	249	283	0,2	0,8	3,8
wmitest-1.xml	1531	1756	0,8	0,4	0,5
wmineu-00.xml	7282	9092	54,8	95,1	1,7
wmineu-06.xml	16027	20291	343,6	645,2	1,9
wmineu-13.xml	1406	1587	1,4	0,8	0,6

Abbildung 50: Zeitmessung für die Anwendung der Operationen

In Abbildung 50 sind die Laufzeiten der Operationen bei unterschiedlichen Punktmengenrepräsentationen zu sehen. Der Wert *t1* bezeichnet die Laufzeit bei Verwendung des k-d Baums und *t2* die Laufzeit bei Verwendung der Delaunay Triangulierung. Für jeden der Pläne wurden sämtliche Operationen in einer sinnvollen Reihenfolge fünf Mal ausgeführt. Die abgebildeten Laufzeiten sind die Mittelwerte über die fünf einzelnen Laufzeiten

Der in CGAL implementierte k-d Baum ist nicht für dynamisches Einfügen und Löschen von Punkten konzipiert. Die Operationen am ungerichteten Graphen könnten mit der Implementierung einer dynamischen Variante eines k-d Baum sicherlich noch beschleunigt werden. Ein solcher dynamischer k-d Baum wird von Procopiuc et al. in [PAAV03] beschrieben.

Verbesserungen der Laufzeit sind sicher auch beim Suchalgorithmus von kantennahen Knoten möglich. Durch die Überprüfung aller Knoten im Begrenzungsrechteck werden vor allem bei längeren diagonalen Kanten wesentlich mehr Knoten untersucht als notwendig. Eine mögliche Alternative wäre beispielsweise ein Algorithmus von Sergei Bespamyatnikh, der in [Bes03] beschrieben ist.

Generell problematisch für die vorgestellten Algorithmen ist die Tatsache, dass sich bei der Ausführung unter Umständen die Geometrie ändern kann. Werden Knoten auf nahe liegende Kanten gezwungen, kann es zu Verschiebungen von Kanten kommen. Dadurch kann es passieren, dass andere Knoten in die ε-Umgebung der verschobenen Kante fallen. In solchen Fällen müsste die Operation erneut ausgeführt werden, um sicherzustellen, dass sich keine Knoten innerhalb des Suchabstandes von Kanten befinden. In der Praxis zeigt sich, dass solche Fälle nur dann auftreten, wenn die maximale relevante Entfernung ε bezüglich der Geometrie nicht korrekt gewählt wird.

Die maximale Entfernung ε sollte stets so gewählt werden, dass die kleinsten gewünschten geometrischen Elemente des Plans erhalten bleiben. Ein Wert, der beispielsweise der halben Länge der kürzesten gewollten Kante des Plans entspricht, hat in der Praxis stets stabile Ergebnisse gebracht.

Weiters ist es problematisch, eine möglichst universelle Operation für die Transformation der Plansymbole zu finden. Die vorgestellte Operation kommt zwar mit einer großen Menge an Symbolen zurecht, kann aber bei weitem nicht alle möglichen Symbole korrekt in Spezialkanten umwandeln.

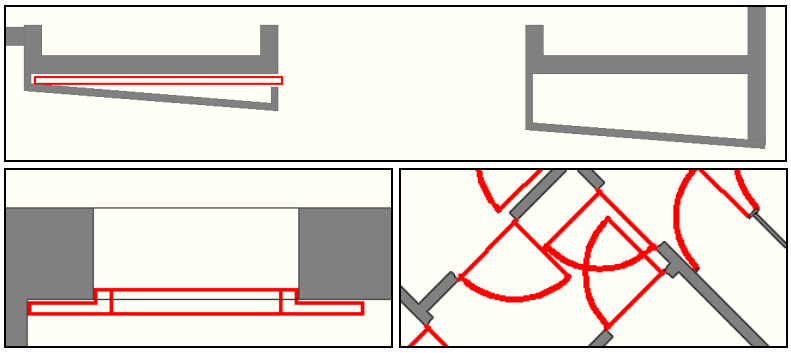


Abbildung 51: Probleme mit Türsymbolen in CAD-Plänen

Zwei Beispiele für Symbole, die von der vorgestellten Operation nicht umgewandelt werden können, sind in Abbildung 51 (Bilder oben und unten links) dargestellt.

Der vorgestellte Ansatz ist ebenfalls problematisch, wenn sich Türsymbole überschneiden. Da Symbole als Zusammenhangskomponenten erkannt werden, würden zwei oder mehr Symbole als eine Zusammenhangskomponente erkannt, und somit nicht korrekt in Spezialkanten umgewandelt werden (Abbildung 51: Bild unten rechts).

3.4 Analyse der Flächeninformationen

Als nächstes gilt es die bereinigten und mit Spezialkanten semantisch angereicherten Plandaten weiter zu untersuchen. Die von den Kanten des Graphen aufgespannten Flächen müssen erkannt und klassifiziert werden. Ziel ist es, eine abstraktere Repräsentation der Plandaten zu erreichen.

Im Kapitel über semantische Gebäudemodelle wurden bereits zwei Verfahren zur Flächenanalyse beschrieben:

- Algorithmus zum Ermitteln von Begrenzungslinien im BMG (mit der Erweiterung zur Behandlung von verschachtelten Flächen)
- Ermitteln der Basiszyklen im AUG (Verschachtelte Flächen werden in weiterer Folge bei der Bildung der Bereiche berücksichtigt)

Für diese Arbeit wurde die Analyse der Flächen wiederum mit einem Modul der CGAL Geometriebibliothek umgesetzt. Die in diesem Modul verwendeten Datenstrukturen und Algorithmen werden im folgenden Abschnitt beschrieben.

Zur Bildung und Klassifikation der Flächen kommen wie schon bei der Bereinigung der Plandaten verschiedene Operationen zum Einsatz. Jede dieser Operationen wird in einem eigenen Kapitel noch genauer betrachtet.

3.4.1 2D Anordnungen in CGAL

Die folgende Beschreibung der Datenstrukturen und Algorithmen, welche in Summe das Modul für 2D Anordnungen in CGAL ausmachen, wurde aus [FHH+00] und [WFZH07] zusammengefasst.

Die Klasse `CGAL::Arrangement_2` repräsentiert die planare Einbettung einer Menge von, in ihrem Inneren disjunkten, schwach x-monotonen Kurven. Schwach x-monoton bedeutet, dass die Kurve von jeder beliebigen vertikalen Geraden maximal einmal geschnitten wird, außer die Kurve ist selbst vertikal ausgerichtet. Für diese Arbeit sind Liniensegmente der einzige vorkommende Kurventyp. Das die oben genannte Bedingung für diesen Kurventyp zutrifft, ist unmittelbar klar.

Die Repräsentation der topologischen Zusammenhänge erfolgt getrennt von der Repräsentation geometrischer Primitive. Die konkreten Repräsentationen werden über zwei Template Parameter der Klasse `CGAL::Arrangement_2` festgelegt.

Der erste Parameter `Traits` definiert die Schnittstellen zwischen der abstrakten Topologischen Repräsentation und den konkreten geometrischen Operationen.

Die Beschaffenheit eines Typs, der als `Traits` Parameter zulässig ist, wird nach dem Konzept-Modell Prinzip festgelegt. Dies ist vergleichbar mit der klassischen, objektorientierten Herangehensweise, in der Schnittstellenklassen festlegen, welche Methoden eine Klasse implementieren muss, damit sie gewissen Anforderungen genügt. Analog dazu definieren Konzepte ebenfalls Methoden und Typen, die eine konkrete Klasse (ein Modell) zur Verfügung stellen muss, um für bestimmte Zwecke eingesetzt werden zu können.

Die konkreten Anforderungen für einen als `Traits` Parameter übergebenen Typ werden im Konzept `ArrangementBasicTraits_2` festgelegt.

Gefordert sind zunächst die Objekttypen `X_monotone_curve_2` und `Point_2`. Diese entsprechen der bereits bezeichneten, speziellen Klasse von Kurven und einem Typ für die Repräsentation von Punkten. Außerdem müssen von jedem Modell dieses Konzepts bestimmte Prädikate ausgewertet werden können. Zwei einfache Beispiele für solche Prädikate sind:

- *point-status*: Für eine gegebene Kurve *C* und einen gegebenen Punkt *P* sagt dieses Prädikat aus, ob der Punkt über, unter oder auf der Kurve liegt
- *compare-to-right*: Legt eine Ordnungsrelation für zwei Kurven C_1 und C_2 fest, die denselben linken Endpunkt haben

Im zweiten Template Parameter `Dcel` wird die Datenstruktur übergeben, welche zur Speicherung der topologischen Zusammenhänge der Anordnung dient.

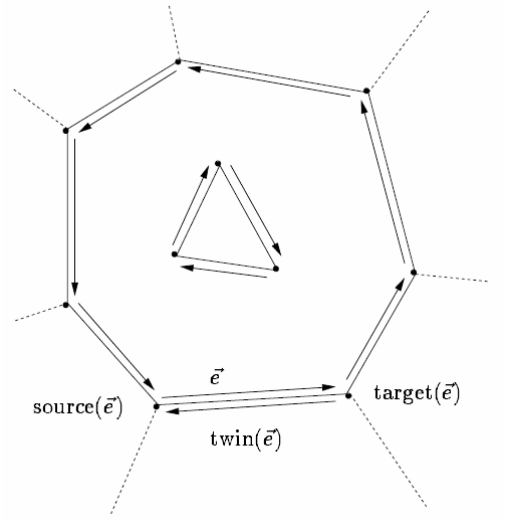


Abbildung 52: Doppelt verkettete Liste von Kanten (aus [FHH+00])

Die Repräsentation der topologischen Zusammenhänge erfolgt mit Hilfe einer auf doppelt verketteten Listen von Kanten (doubly connected edge list, DCEL) basierenden Datenstruktur (Abbildung 52). Diese Datenstruktur erlaubt es, die Speicherung der Flächenanordnung effizient aufrecht zu erhalten.

In einer DCEL wird jede Kurve von zwei Halbkanten repräsentiert. Eine Halbkante zeigt vom Quellpunkt (*source*) zum Zielpunkt (*target*) und die Zweite (*twin* der Ersten) vom Zielpunkt zum Quellpunkt.

Halbkanten dienen also dazu, Punkte zu verbinden und Flächen voneinander zu trennen. Jede Halbkante speichert einen Zeiger auf die links liegende Fläche. Weiters werden Halbkanten in doppelt verketteten Listen abgespeichert. Diese Ketten bilden jeweils die gegen den Uhrzeigersinn gerichtete, Umrandung einer Fläche.

Außerdem kann es sein, dass für eine nicht-triviale Fläche eine Liste von Löchern gespeichert wird. Dies ist dann der Fall, wenn innerhalb einer Fläche zumindest eine weitere disjunkte Kette von im Uhrzeigersinn gerichteten Halbkanten existiert.

3.4.2 Operationen

Im Folgenden sind die Operationen beschrieben, die aus den Kanten des ungerichteten Graphen eine abstrakte Beschreibung des Plans generieren.

3.4.2.1 Flächen aus Kanten erzeugen

Die erste Operation sorgt dafür, dass aus den Kanten des ungerichteten Graphen Flächen generiert werden.

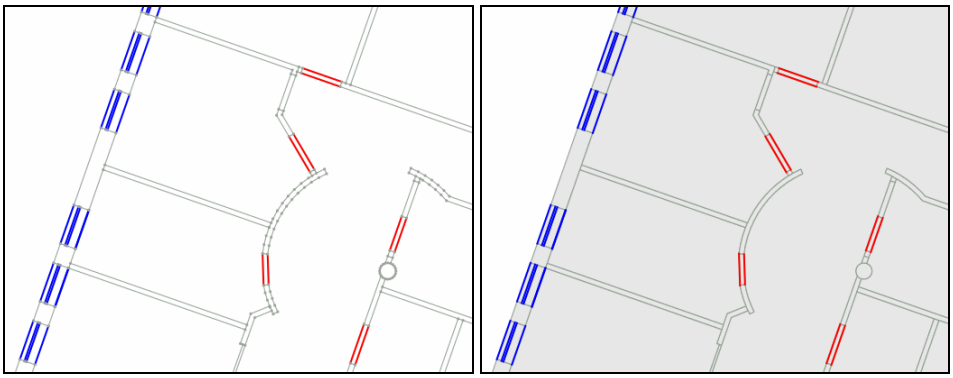


Abbildung 53: Erzeugen von Flächen aus Kanten des ungerichteten Graphen

Zu diesem Zweck werden die Kanten in beliebiger Reihenfolge in eine Instanz der Klasse `CGAL::Arrangement_2` eingefügt. Dies geschieht mit Hilfe der freien Funktion `CGAL::insert` welche die Anordnung und das einzufügende Liniensegment übergeben wird. Die Darstellung in der Applikation ändert sich dahingehend, dass Flächen grau eingefärbt werden (Abbildung 53).

Beim Einfügen einer Kante des ungerichteten Graphen in die Anordnung entstehen jeweils zwei Halbkanten. Damit diese mit den gleichen Attributen versehen sind, wie die ursprünglichen, ungerichteten Kanten, kommt ein spezieller Mechanismus zum Einsatz.

Die Klasse `CGAL::Arr_observer` wurde nach dem Entwurfsmuster Beobachter (englisch: Observer) umgesetzt. Der Zweck des Entwurfsmusters Beobachter wird in [GHJV95] folgendermaßen beschrieben:

> *Definiere eine 1-zu-n Abhängigkeit zwischen Objekten, so dass die Änderungen des Zustands eines Objekts dazu führt, dass alle abhängigen Objekte benachrichtigt und automatisch aktualisiert werden.*

In diesem Fall wird der Anwender der Klasse `CGAL::Arr_observer` über sämtliche Änderungen in der Anordnung benachrichtigt. Diese Benachrichtigung wird verwendet, um einer in die Anordnung eingefügten Kante die Attribute der ursprünglichen Kante zuzuweisen.

```
1:    struct InsertionObserver : public Exact::ArrangementObserver
2:    {
3:      virtual void after_create_edge( Halfedge_handle e )
4:      {
5:        UndirectedGraph::EdgeHandle edge_to_be_inserted =
6:          undirected_graph_.findEdge(
7:            undirected_graph_.findNearestVertex(
8:              Exact::dpoint( e->source()->point() ), 0. ),
9:            undirected_graph_.findNearestVertex(
10:             Exact::dpoint( e->target()->point() ), 0. ) );
11:       InfoTypes::EdgeInfo edge_info;
12:       edge_info = edge_to_be_inserted.info();
13:       e->set_data( edge_info );
14:       e->twin()->set_data( edge_info );
15:     }
16:   }
```

Abbildung 54: Implementierung des Beobachters zur Erhaltung von Attributen

Die Implementierung des Beobachters (Abbildung 54) erfolgt über Ableitung einer Klasse von `CGAL::Arr_observer`. Der Typ `Exact::ArrangementObserver` (Zeile 1) entspricht einer Ausprägung der Klasse `CGAL::Arr_observer`.

Die eigentliche Benachrichtigung erfolgt über die polymorphe Methode `after_create_edge`. Sie wird immer dann aufgerufen, wenn eine neue Kante in die Anordnung eingefügt wird (Zeile 3).

Um die Attribute aus dem ungerichteten Graph zu übernehmen, wird die entsprechende Kante zunächst im Ausgangsgraphen gesucht (Zeilen 5-10). Die Suche erfolgt über Start- und Endpunkt der Kante.

Schlussendlich werden beiden Halbkanten die Attribute der gefundenen Kante zugewiesen (Zeilen 11-14).

3.4.2.2 Flächen durch Spezialkanten klassifizieren

Im nächsten Schritt gilt es, den aus den Kanten gebildeten Flächen, eine abstrakte Bedeutung zuzuordnen. Dies erfolgt ausschließlich über Spezialkanten. Spezialkanten sind Kanten, denen über ein Attribut eine bestimmte Bedeutung zugeordnet wurde.

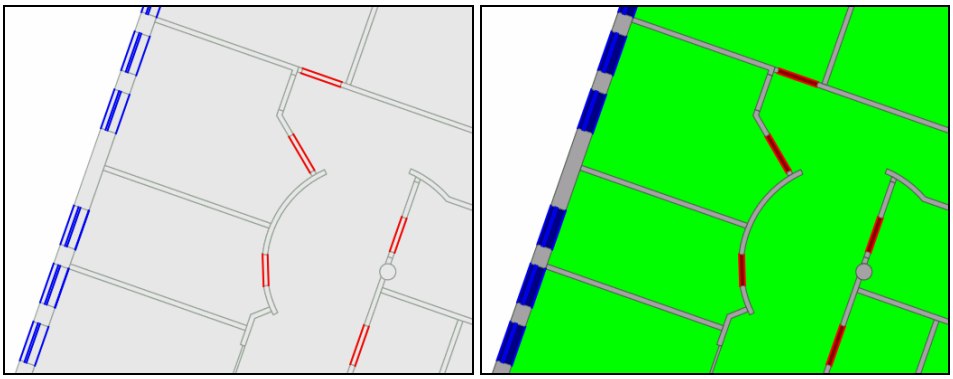


Abbildung 55: Klassifikation der Flächen durch Spezialkanten der Anordnung

Momentan sind Spezialkanten für die Typen Tür (rote Kanten) und Fenster (blaue Kanten) vorgesehen. Nach der Klassifikation werden die Flächen entsprechend des Typs eingefärbt

dargestellt. Fensterflächen werden blau, Türflachen rot, Wandflächen grau und Flächen die Räumen entsprechen grün eingefärbt (Abbildung 55).

Die Klassifikation erfolgt nach einfachen Regeln:

- Hat eine Fläche vier begrenzende Kanten handelt es sich um einen Kandidaten für eine Tür- oder Fensterfläche. Die Klassifikation als Tür- oder Fensterfläche erfolgt nur dann, wenn zwei der vier Kanten der Flächenbegrenzung Spezialkanten des entsprechenden Typs sind.
- Ist eine Fläche durch mehr als vier Kanten begrenzt und ist zumindest eine der Kanten eine Spezialkante, wird die Fläche als Raum klassifiziert.
- Alle anderen Flächen werden als Wand bzw. Hindernis klassifiziert.

3.4.2.3 Flächen vereinfachen

Oft sind Flächen nach der Klassifikation unnötig unterteilt. Um diese unnötigen Unterteilungen zu entfernen, werden einfach Kanten aus der Anordnung entfernt, welche auf beiden Seiten an Flächen des gleichen Typs angrenzen.

Eine Ausnahme bilden dabei einfache Wandkanten, die zwei Raumflächen teilen. Oft sind Räume in CAD-Plänen absichtlich durch einfache Kanten in mehrere Bereiche aufgeteilt. Wandkanten, die auf beiden Seiten an Raumflächen angrenzen, werden also nicht aus der Anordnung entfernt.

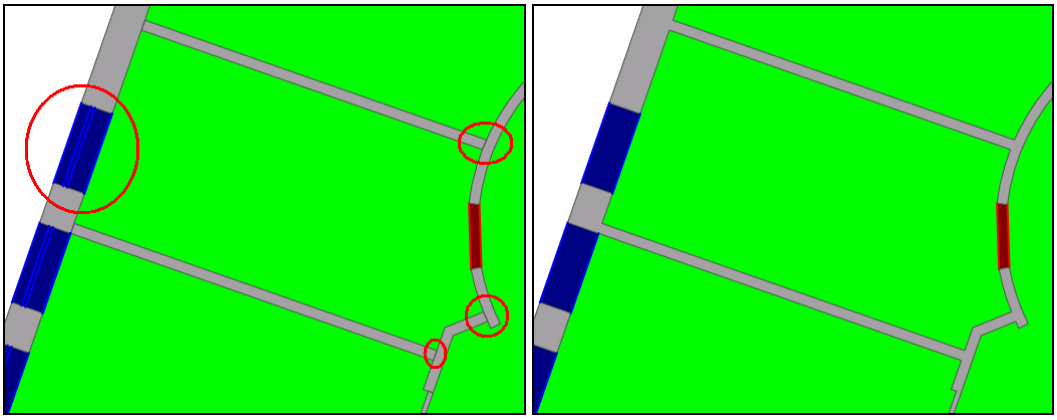


Abbildung 56: Vereinfachung der Flächen durch Entfernen unnötiger Kanten

Im CAD-Plan aus Abbildung 56 sind Fenstersymbole so beschaffen, dass diese als zwei getrennte, aneinander anliegende Fensterflächen erkannt werden. Auch die unnötige Unterteilung von Wandflächen ist gut erkennbar.

3.5 Diskussion

Mit den vorgestellten Operationen kann aus einem CAD-Plan ein semantisches Modell der repräsentierten Räumlichkeiten erzeugt werden. Dieses Modell besteht aus Flächen, denen jeweils einer der Typen Wand, Tür, Fenster oder Raum zugewiesen wurde.

Die vorgestellten Operationen sind keinesfalls vollständig. In CAD-Plänen sind noch viele Informationen enthalten, die bis jetzt noch gar nicht berücksichtigt wurden. Bezeichnungsblöcke oder Labels für Räume könnten beispielsweise verwendet werden, um Raumflächen bereits mit dem korrekten Namen zu versehen. Dasselbe gilt auch für Bezeichnungen von Türen und Fenstern. Das Problem bei Bezeichnungen ist, dass diese vom Zeichner des Plans oft nicht einmal innerhalb eines Plans konsistent umgesetzt sind. So wird beispielsweise ein Teil der Räume in einem Plan mit Labels und die übrigen Räume mit Blockreferenzen bezeichnet.

Weiters wurden vertikale Portale wie Treppen und Aufzüge bis jetzt noch nicht berücksichtigt. Diese könnten ebenfalls als Flächen repräsentiert werden. Für eine Treppe würde die Fläche dem Grundriss des Stiegenhauses entsprechen und für Aufzüge dem Grundriss des Aufzugsschachtes.

Voraussetzung für das Bilden solcher Flächen wäre zunächst eine Zuordnung bestimmter Schichten des Plans zu einem Typ für vertikale Portale (ähnlich wie dies bereits für Türen und Fenster geschieht). Die Operation für die Klassifikation von Flächen müsste ebenfalls erweitert werden. Denkbar wäre die Behandlung eines neuen Spezialkantentyps für die Begrenzung von Grundrissen vertikaler Portale.

Für das nächste Kapitel stellt sich die Frage, wie die klassifizierten Flächen nun hierarchisch in einer relationalen Datenbank abgelegt werden können. Hierarchie bezieht sich dabei auf die Hierarchie des Gebäudes im Sinne der einzelnen Stockwerke.

Abbildung 57: Aus einem CAD-Plan generiertes Modell eines Stockwerks

4 Repräsentation des Gebäudemodells in einer relationalen Datenbank

Ziel dieses Abschnitts der Arbeit ist es, die aus den CAD-Plänen gewonnenen Informationen in eine relationale Datenbank einzupflegen. Auf diese Art entsteht, Schritt für Schritt, ein abstraktes semantisches Modell des Gebäudes.

4.1 Relationale Datenbanken

Eine der ersten Arbeiten zum Thema relationale Datenmodelle wurde von Edgar Codd Anfang der 70er Jahre veröffentlicht ([Cod70]). Aufbauend auf diesen Ideen entstand bei IBM System R, eines der ersten relationalen Datenbank Managementsysteme (Relational Database Management System, RDBMS).

Die Abfragesprache für dieses System wurde SEQUEL (Structured English Query Language, strukturierte, englische Abfragesprache) genannt. Aus dieser Sprache ging in weiterer Folge auch die mittlerweile sehr weit verbreitete Abfragesprache SQL (Structured Query Language, strukturierte Abfragesprache) für relationale Datenbanken hervor.

Relationale Datenbanken sind grundsätzlich in *Tabellen* organisiert. Jede Tabelle ist in eine Menge von *Spalten* gegliedert. Für jede Spalte wird ein Datentyp festgelegt, dem die eingetragenen Werte entsprechen müssen. Eine Zeile der Tabelle mit konkreten Werten wird auch als *Datensatz* bezeichnet.

Für jede Tabelle muss ein eindeutiger *Primärschlüssel* definiert werden. Dieser Primärschlüssel dient zur Identifikation eines Datensatzes innerhalb der Tabelle. Primärschlüssel können entweder über eine, oder über mehrere Spalten einer Tabelle gebildet werden. Eindeutigkeit dieses Schlüssels innerhalb der Tabelle muss jedenfalls gegeben sein. Oft enthält eine Tabelle eine zusätzliche Spalte, die als Primärschlüssel dient. In einer solchen Spalte wird meist ein ganzzahliger Index gespeichert, der für jeden neuen Eintrag automatisch erhöht wird.

Neben Primärschlüsseln können Spalten einer Tabelle auch als *Fremdschlüssel* deklariert werden. Damit wird festgelegt, dass die Werte in einer bestimmten Spalte jeweils Datensätze einer anderen Tabelle referenzieren. Dieser Mechanismus wird verwendet um 1:n und n:n Beziehungen abzubilden. Sind Tabellen über Fremdschlüssel miteinander verknüpft, stellt das RDBMS die Integrität dieser Referenzen sicher. Wird ein Datensatz von einem anderen Datensatz über einen Fremdschlüssel referenziert, kann dieser referenzierte Datensatz nicht gelöscht werden (referentielle Integrität).

4.2 Eine einfache Beispieldatenbank

Wie bereits erwähnt, ist SQL eine Sprache mit der konkrete Abfragen bezüglich einer relationalen Datenbank durchgeführt werden können. Anhand einer einfachen Beispieldatenbank sollen Syntax und Möglichkeiten in SQL erläutert werden.

Eine Möglichkeit für die Darstellung eines Datenbankschemas ist das sogenannte Gegenstands-Beziehungs Diagramm (englisch „Entity-Relationship Diagram“).

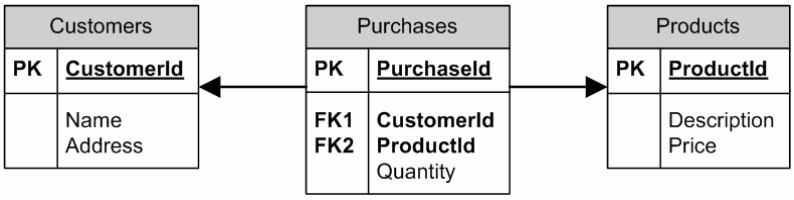


Abbildung 58: Gegenstands-Beziehungs Diagramm einer einfachen Datenbank

In Abbildung 58 ist ein Gegenstands-Beziehungs Diagramm für eine Datenbank dargestellt. Diese Beispieldatenbank bildet einen einfachen Geschäftsprozess ab. Spalten, die Primärschlüssel sind, werden mit dem Kürzel PK (Primary Key) und Spalten, die Fremdschlüssel sind, mit dem Kürzel FK (Foreign Key) gekennzeichnet. Die Abhängigkeiten zwischen Tabellen, die sich aus Fremdschlüsseln ergeben, sind im Diagramm als Pfeile dargestellt. Ist die Bezeichnung einer Spalte fett gedruckt, darf der Wert in dieser Spalte einer Tabelle nicht undefiniert („NULL“) sein.

Die Tabelle `Customers` enthält dabei die Daten zu allen Kunden eines Geschäfts. Sämtliche vertriebenen Produkte sind in der Tabelle `Products` abgespeichert. Über die Tabelle `Purchases` werden sämtliche Einkäufe der Kunden registriert.

Aus dem Diagramm sind Abhängigkeiten zwischen den Tabellen ersichtlich, die sich aus der Vergabe von Fremdschlüsseln ergeben. Die Tabelle `Purchases` ist mit zwei Fremdschlüsseln versehen. Es wird jeweils ein Kunde (`CustomerId`) mit einem gekauften Produkt (`ProductId`) in Verbindung gebracht. Die einzige Zusatzinformation, welche durch die Tabelle hinzufügt wird, ist die Anzahl der Einheiten, die der Kunde gekauft hat.

Natürlich muss in der Tabelle `Purchases` mit `PurchaseId` ein neuer Primärschlüssel eingeführt werden, da sonst keine eindeutige Identifizierung eines Einkaufs möglich wäre (ein Kunde könnte ja mehrmals die gleiche Menge des gleichen Produkts bestellen).

```
1:    INSERT INTO "Products" ( "Description", "Price" )
2:      VALUES ( 'Mouse', 30.0 );
```

Abbildung 59: SQL Anweisung zum Einfügen eines Datensatzes in eine Tabelle

Mit Hilfe einer `INSERT INTO` Anweisung kann ein neuer Datensatz in eine Tabelle eingefügt werden. Im Fall von Abbildung 59 wird der Produkttabelle (`Products`) ein neues Produkt mit Bezeichnung und Preis hinzugefügt. Die auszufüllenden Spalten können nach der Zieltabelle angegeben werden. Die einzutragenden Werte werden mit dem Schlüsselwort `VALUES` festgelegt (Zeile 2).

```
1:    SELECT *
2:      FROM "Products"
3:      WHERE "Price" > 50.0;
```

Abbildung 60: SQL Anweisung zum Abfragen der Datensätze einer Tabelle

Die SELECT Anweisung kann verwendet werden, um Datensätze einer Tabelle abzufragen. Mit dem WHERE Schlüsselwort können zusätzlich noch Kriterien für die gesuchten Datensätze übergeben werden. Im Beispiel aus Abbildung 60 werden alle Produkte ausgewählt, deren Preis größer als 50.0 ist (Zeile 3).

```
1:    UPDATE "Products"
2:      SET "Price" = 15.0
3:      WHERE "Description" = 'Mouse';
```

Abbildung 61: SQL Anweisung zum Erneuern eines Wertes in einer Tabelle

Mit der UPDATE Anweisung können Werte in Tabellen verändert werden. Im Beispiel aus Abbildung 61 wird der Preis von allen Produkten, die mit der Beschreibung „Mouse“ versehen sind auf 15.0 gesetzt. Das Einschränken auf Produkte mit einer bestimmten Beschreibung erfolgt wiederum mit dem Schlüsselwort WHERE (siehe Zeile 3).

```
1:    DELETE FROM "Products"
2:      WHERE "Price" < 20.0;
```

Abbildung 62: SQL Anweisung zum Löschen eines Datensatzes aus einer Tabelle

Die Anweisung DELETE FROM wird verwendet um Datensätze aus einer Tabelle zu löschen. Im Beispiel aus Abbildung 62 werden alle Produkte aus der Produkttabelle entfernt, deren Preis kleiner als 20.0 ist. Die Auswahl der zu löschenden Datensätze erfolgt wieder mit der Anweisung WHERE (Zeile 2).

Die meisten RDBMSe ermöglichen es auch, die Verwaltung der Datenbank selbst mit Hilfe von SQL Anweisungen durchzuführen. So gibt es beispielsweise Anweisungen zum Erstellen von neuen Datenbanken und Tabellen. Auch die Vergabe von Primär- und Fremdschlüsseln kann mit speziellen Anweisungen durchgeführt werden. Moderne RDBMSe beinhalten meist grafische Benutzeroberflächen, mit denen solche administrativen Aufgaben bequem erledigt werden können. Da die Administration und das Aufsetzen von Datenbanken nicht im Fokus dieser Arbeit liegt, soll an dieser Stelle nicht weiter auf diese Möglichkeiten eingegangen werden.

4.2.1 Geometrische Erweiterungen

Für einige Datenbanksysteme gibt es Erweiterungen, die es ermöglichen Geometriedaten direkt zu verarbeiten. Dies hat prinzipiell zwei Implikationen auf das RDBMS:

- geometrische Merkmale wie Polygone und Linienzüge können als Typen für Spalten von Tabellen verwendet werden
- Abfragen unterstützen spezielle Befehle zur Verarbeitung von geometrischen Merkmalen

Die Erweiterung PostGIS für das RDBMS PostgreSQL ist ein Beispiel für eine solche Erweiterung. Da beide Softwarepakete frei zur Verfügung stehen, wurden sie zur Umsetzung der Datenbankfunktionalität für diese Arbeit ausgewählt.

Das Ziel von PostGIS ist es, den PostgreSQL Datenbankserver dahingehend zu erweitern, dass er als RDBMS für Geoinformationssysteme verwendet werden kann. Zu diesem Zweck wird das Ziel verfolgt, den offenen Standard „Simple Features Specification for SQL“ (Spezifikation einfacher Merkmale in SQL, SFS) des OGC zu implementieren. Die Implementierung eines Teils dieses Standards wurde bereits erfolgreich zertifiziert. Im SFS sollen wohldefinierte und gebräuchliche Mechanismen definiert werden, mit denen Applikationen auf geographische und geometrische Merkmale in relationalen Datenbanken zugreifen können.

4.2.1.1 Repräsentation geometrischer Merkmale

In der SFS werden prinzipiell zwei verschiedene Repräsentationen für geometrische Merkmale angeboten:

- Well-Known Text (WKT, wohlbekannte Textkodierung): Bezeichnet eine einfache, menschenlesbare, textbasierte Repräsentation von geometrischen Merkmalen.
- Well-Known Binary (WKB, wohlbekannte Binärkodierung): Bezeichnet eine binär kodierte Repräsentation von geometrischen Merkmalen.

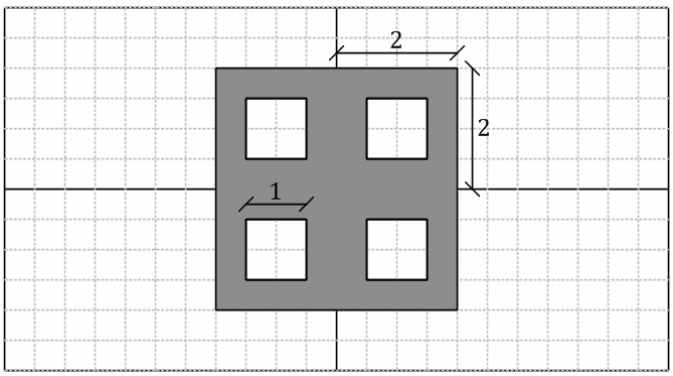


Abbildung 63: Polygon mit Löchern als Beispiel für die Textkodierung

Als Beispiel wird das in Abbildung 63 dargestellte Polygon in der WKT-Kodierung dargestellt:

```
1:    POLYGON(
2:      ( -2.0 +2.0, -2.0 -2.0, +2.0 -2.0, +2.0 +2.0, -2.0 +2.0 ),
3:      ( -1.5 +1.5, -0.5 +1.5, -0.5 +0.5, -1.5 +0.5, -1.5 +1.5 ),
4:      ( +0.5 +1.5, +1.5 +1.5, +1.5 +0.5, +0.5 +0.5, +0.5 +1.5 ),
5:      ( +0.5 -0.5, +1.5 -0.5, +1.5 -1.5, +0.5 -1.5, +0.5 -0.5 ),
6:      ( -1.5 -0.5, -0.5 -0.5, -0.5 -1.5, -1.5 -1.5, -1.5 -0.5 )
7:    )
```

Wie üblich wird ein Polygon durch eine äußere Begrenzungslinie und optionalen Begrenzungslinien für die Löcher im Inneren definiert. Die Punktreihenfolge für die äußere Begrenzungslinie ist gegen den Uhrzeigersinn festgelegt. Die Punkte der

Begrenzungslinien von Löchern werden im Uhrzeigersinn festgelegt.

Die Binärkodierung erfolgt im Wesentlichen analog zur Textkodierung. Jedem Datensatz wird zunächst ein Kopf vorangestellt, in dem die Bytereihenfolge und der Typ des repräsentierten Merkmals festgelegt werden. Für Polygone folgt danach die Anzahl der Begrenzungslinien. Weiters folgen für jede Begrenzungslinie die Anzahl der enthaltenen Punkte und deren Koordinaten in binärer Repräsentation nach IEEE 784 kodiert.

4.2.1.2 Beispielabfrage über ein geometrisches Merkmal

PostGIS bietet eine Vielzahl an Funktionen, die auf geometrische Objekte angewendet werden können. Im Folgenden soll das Konzept anhand eines kurzen Beispiels erläutert werden.

```
1:    SELECT
2:      ST_Area( ST_GeomFromText(
3:        'POLYGON('
4:        '  ( -2.0 +2.0, -2.0 -2.0, +2.0 -2.0, +2.0 +2.0, -2.0 +2.0 ),'
5:        '  ( -1.5 +1.5, -0.5 +1.5, -0.5 +0.5, -1.5 +0.5, -1.5 +1.5 ),'
6:        '  ( +0.5 +1.5, +1.5 +1.5, +1.5 +0.5, +0.5 +0.5, +0.5 +1.5 ),'
7:        '  ( +0.5 -0.5, +1.5 -0.5, +1.5 -1.5, +0.5 -1.5, +0.5 -0.5 ),'
8:        '  ( -1.5 -0.5, -0.5 -0.5, -0.5 -1.5, -1.5 -1.5, -1.5 -0.5 )'
9:        ')' ) );
```

Abbildung 64: Berechnung der Fläche des Polygons in einer SQL Abfrage

Im Beispiel in Abbildung 64 wird die Fläche des zuvor abgebildeten Polygons in einer SQL Abfrage berechnet. Zunächst muss das geometrische Objekt mit der Funktion `ST_GeomFromText` aus der WKT-Kodierung erzeugt werden. Die Funktion `ST_Area` berechnet dann die Fläche des übergebenen geometrischen Objekts. Wird die Abfrage ausgeführt, liefert sie den erwarteten Wert für die Fläche des Polygons (den Wert 12).

4.2.2 Anbindung an objektorientierte Softwarearchitektur

Sollen die Daten einer relationalen Datenbank in einer objektorientierten Softwarearchitektur verwendet werden, muss dazwischen eine Abstraktionsschicht eingeführt werden. Relationale und objektorientierte Datenmodellierung können sich in der Umsetzung sehr deutlich voneinander unterscheiden. Unterschiede in der Modellierung und die Probleme die sich daraus ergeben, werden in der englischen Fachliteratur auch als „Object Relational Impedance Mismatch“ bezeichnet.

Mittlerweile haben sich bereits einige Herangehensweisen zur Abbildung relationaler Datenmodelle in objektorientierter Software etabliert. Für diese Arbeit wurden konkret zwei Konzepte verwendet um die Abbildung umzusetzen bzw. zu erleichtern.

4.2.2.1 Single Table Inheritance

Dieses Konzept bezieht sich auf den Entwurf der Datenbank selbst. Dabei werden Objekte unterschiedlichen Typs in ein und dieselbe Tabelle der Datenbank gespeichert. Die Tabelle

enthält dann Spalten für die Übermenge aller Objektattribute. Der eigentliche Typ des Objekts wird in einer zusätzlichen Spalte abgelegt. Dies widerspricht natürlich einigen Regeln der relationalen Datenmodellierung, erleichtert die Anbindung an objektorientierte Software allerdings erheblich.

4.2.2.2 Data Access Object

Mit dieser Herangehensweise lässt sich das Problem des Zugriffs auf die Daten der relationalen Datenbank elegant lösen. Für jede Tabelle der Datenbank wird dabei im objektorientierten Design eine Klasse vorgesehen. Instanzen dieser Klasse beziehen sich dann jeweils auf eine Zeile der modellierten Tabelle. Dieses Datenobjekt kann dann nicht nur die eigentlichen Werte aus der Datenbank zwischenspeichern, sondern kennt auch den Primärschlüssel des repräsentierten Eintrags. Auf diese Art können die entsprechenden Daten in der Datenbank jederzeit aktualisiert oder auch gelöscht werden.

4.3 Relationales Schema für das Gebäudemodell

Es gilt ein Schema für die relationale Datenbank zu finden, das mächtig genug ist, ein semantisches Gebäudemodell abzuspeichern. Im Idealfall sollte dieses Schema auch flexibel für Erweiterungen des Gebäudemodells sein.

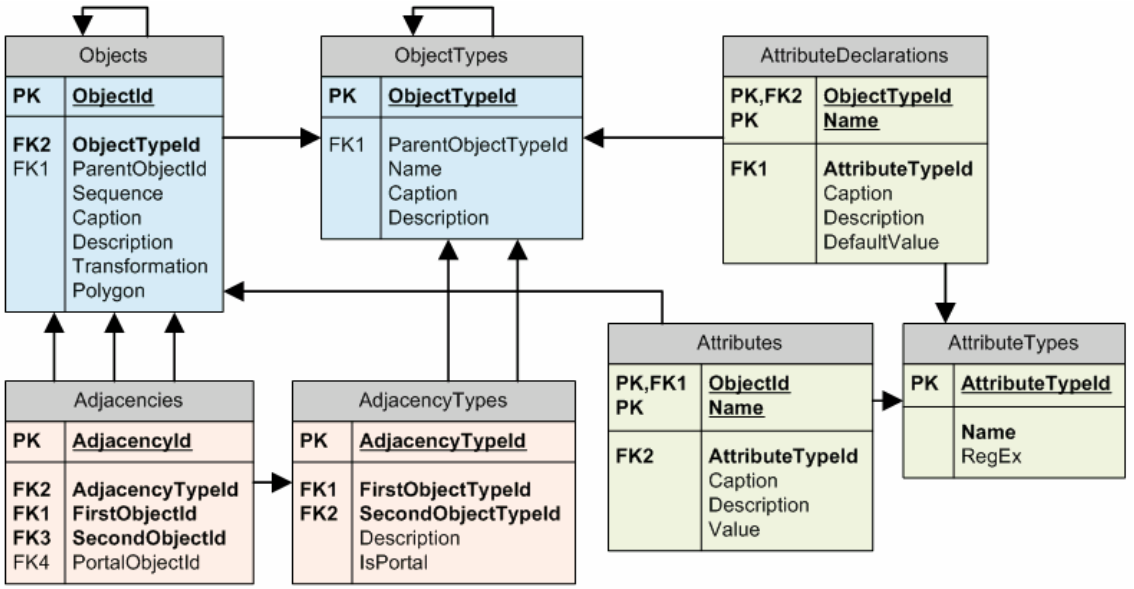


Abbildung 65: Relationales Datenbankschema für das semantische Gebäudemodell

In Abbildung 65 ist das relationale Schema für das semantische Gebäudemodell abgebildet. Die Darstellung erfolgt anhand eines Gegenstands-Beziehungs Diagramms. Im Wesentlichen gliedert sich die Modellierung in drei Teile:

- Objekte (Tabellen `ObjectTypes` und `Objects`)
- Attribute (Tabellen `AttributeDeclarations`, `AttributeTypes` und `Attributes`)
- Nachbarschaften (Tabellen `AdjacencyTypes` und `Adjacencies`).

Für die Objekte werden zunächst die unterschiedlichen Typen in der Tabelle `ObjectTypes` abgelegt. Ein Typ wird durch einen Namen (Spalte `Name`), einen kurzen und einen langen Beschreibungstext (Spalten `Caption` und `Description`) und optional durch einen Elterntyp (Spalte `ParentObjectTypeId`) definiert. Mit Hilfe des Elterntyps kann eine hierarchische Gliederung der eigentlichen Objekte vorgegeben werden. Würde man einen Typ für Raum festlegen, wäre der entsprechende Elterntyp beispielsweise der Typ für ein Stockwerk.

Ein konkretes Objekt wird zunächst durch einen Typ (Spalte `ObjectTypeId`), ein optionales Elternobjekt (Spalte `ParentObjectId`) und die bereits bekannten Beschreibungstexte (Spalten `Caption` und `Description`) definiert. Zusätzlich erhält ein Objekt mit der Spalte `Polygon` auch eine geometrische Entsprechung. Für ein Objekt

das einem Raum entspricht, wird in `Polygon` das entsprechende Raumpolygon abgespeichert. In der Spalte `Transformation` kann zusätzlich eine Transformation, relativ zu einem eventuell vorhandenen Elternobjekt, angegeben werden. Mit dieser Spalte könnte man beispielsweise ein Gebäude im Weltkoordinatensystem georeferenzieren.

Für jeden Objekttyp können schließlich eine Reihe von Attributen deklariert werden (Tabelle `AttributeDeclarations`). Mögliche Attributtypen müssen zunächst in der Tabelle `AttributeTypes` festgelegt werden. Konkrete Attribute werden schließlich in der Tabelle `Attributes` abgelegt. Die einzelnen Spalten dieser Tabellen sollten selbsterklärend sein.

Mögliche Nachbarschaften zwischen Objekten müssen zunächst in der Tabelle `AdjacencyTypes` festgelegt werden. Ein Nachbarschaftstyp wird durch die Typen der benachbarten Objekte (Spalten `FirstObjectTypeId` und `SecondObjectTypeId`) und einem Portalschalter (Spalte `IsPortal`) definiert. Mit dem Portalschalter kann festgelegt werden, ob man sich über diese Nachbarschaft von einem Objekt zum nächsten Objekt bewegen kann. Dies ist für die Anwendung des Gebäudemodells zur Wegfindung wichtig. Die konkreten Nachbarschaften in der Tabelle `Adjacencies` werden von den benachbarten Objekten (`FirstObjectId` und `SecondObjectId`) und einem optionalen Portalobjekt bestimmt. Je nach Art der Modellierung, können zum Beispiel Türobjekte als Portalobjekte verwendet werden. Die Nachbarschaft selbst wird dann zwischen zwei Raumobjekten mit dem entsprechenden Türobjekt als Portal bestimmt.

Das vorgestellte relationale Modell ist sehr allgemein gehalten. Für die konkrete Modellierung eines Gebäudes müssen zuerst konkrete Objekttypen definiert werden.

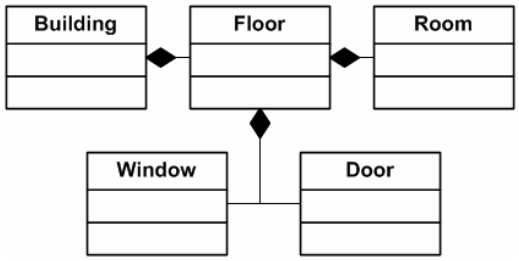


Abbildung 66: Konkrete Objekttypen eines minimalen Gebäudemodells

In Abbildung 66 sind (in informeller UML Notation) die konkreten Typen eines minimalen Gebäudemodells abgebildet. Die Kompositionspfeile legen dabei implizit die hierarchische Gliederung über die Spalte `ParentObjectTypeId` der Tabelle `ObjectTypes` fest.

4.4 Operationen

Die im Folgenden beschriebenen Operationen dienen dazu, die Datenbank mit den aus den CAD-Plänen erfassten Informationen zu füllen. Da diese Daten in der 2D Anordnung der Flächen bereits strukturiert und leicht zugänglich vorliegen, sind diese Operationen sehr einfach umzusetzen.

4.4.1 Importieren von flächenbasierten Objekten

Bei der Erfassung wurden Flächen bereits als Wände, Räume, Türen oder Fenster klassifiziert. Der Import dieser Flächen in die Datenbank wird in Einzelschritten für jeden Typ vorgenommen. Mit Hilfe eines Flächeniterators wird über alle in der Anordnung gespeicherten Flächen iteriert. Entspricht der Typ der aktuellen Fläche dem Typ der importiert werden soll, wird in der Datenbank ein entsprechendes Objekt angelegt. Die geometrische Entsprechung des Objekts wird aus der Repräsentation in der Anordnung in die WKB-Repräsentation umgewandelt.

Flächen mit Löchern werden vor dem Import in die Datenbank noch einer Vereinfachung unterzogen. Oft kommt es vor, dass eine Fläche mehrere aneinander anliegende Löcher besitzt. Diese Löcher werden vor dem Import in die Datenbank zu einem Loch zusammengefasst. Die geometrische Repräsentation der Fläche in der Datenbank kann so möglichst einfach gehalten werden.

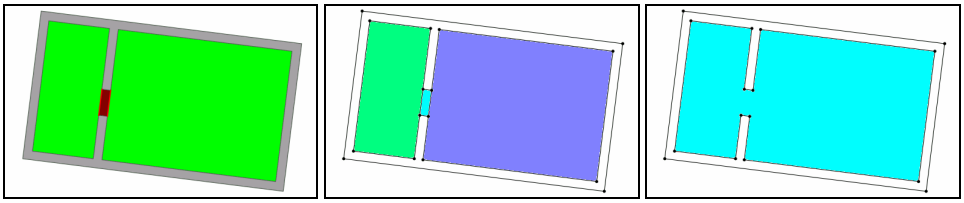


Abbildung 67: Vereinfachung von aneinander anliegenden Löchern

In Abbildung 67 ist ein problematischer Fall zu sehen. Die einzige Wandfläche der Anordnung (graues Polygon im linken Bild) hat jeweils Löcher für die beiden grünen Raumflächen und die rote Türfläche (Flächen im mittleren Bild). Vor dem Import in die Datenbank wird das Polygon für die Wand neu gebildet. Zu diesem Zweck wird aus dem Polygon ein neuer ungerichteter Graph erzeugt. Dabei werden einfach jene Kanten weggelassen, die benachbarte Löcher voneinander trennen (rechtes Bild). Nachdem der ungerichtete Graph erneut einer Flächenanalyse unterzogen wurde, bleibt lediglich ein Loch über (blaue Fläche im rechten Bild).

4.4.2 Importieren von Nachbarschaftsinformationen

Beim vorangegangenen Datenbankimport wird zu jeder Fläche der Anordnung auch die Identifikationsnummer des entsprechenden Objekts (Spalte `ObjectId` in der Tabelle `Objects`) gespeichert. Die Nachbarschaften der Anordnung selbst können mit Hilfe von Halbkanteniteratoren komfortabel ausgewertet werden. Je nach konfigurierten Nachbarschaftstypen (Tabelle `AdjacencyTypes`) können die Nachbarschaften entsprechend in die Tabelle `Adjacencies` eingetragen werden.

4.5 Diskussion

Das vorgestellte relationale Modell der Datenbank ist sehr allgemein und flexibel. Der Nachteil dabei ist, dass die Integrität der gespeicherten Daten nicht vom RDBMS alleine

sichergestellt werden kann. So muss die Software, die auf die Datenbank zugreift dafür sorgen, dass die hierarchische Gliederung von Objekten der, mit den Objekttypen impliziten Gliederung, entspricht.

Weiters ist die hierarchische Gliederung der Objekte in einem Punkt eingeschränkt: Die Beziehung zwischen Objekttyp und Elterntyp ist stets eine n:1 Beziehung. So können unterschiedliche Objekttypen denselben Elterntyp haben, aber ein Objekttyp kann immer nur einen Elterntyp haben. Diese Einschränkung hat sich in der Praxis bis jetzt noch nicht als problematisch herausgestellt.

Im vorgestellten Modell werden alle Objekte des Gebäudes als planare Flächen repräsentiert. Auf diese Art lässt sich mit Sicherheit nur eine Abstraktion eines realen Gebäudes repräsentieren. Zwar könnten schräge Flächen durch Objektattribute spezifiziert werden, eine perfekte geometrische Repräsentation mit allen Details wird mit dem vorgestellten Modell aber nur schwer umsetzbar sein.

5 Übergabe des Gebäudemodells zur Visualisierung und Ergebnisse

Wir gehen nun davon aus, dass in der Datenbank ein hierarchisch gegliedertes, semantisches Flächenmodell eines Gebäudes abgelegt wurde. Für den ersten Teil dieses Kapitels („Übergabe zur Visualisierung“) stellt sich die Frage, wie diese Daten transformiert werden müssen, um eine dreidimensionale Darstellung für eine Visualisierung zu erreichen.

Der zweite Teil des Kapitels („Endergebnis in Form eines größeren Beispielmodells“) widmet sich einem umfangreicheren Beispiel für ein Gebäudemodell. Hier soll der komplette Arbeitsablauf von der Erfassung des Modells aus den CAD-Plänen bis zur dreidimensionalen Darstellung dokumentiert werden.

Im letzten Teil des Kapitels („Verifikation der Methoden anhand eines weiteren Modells“) soll die allgemeine Anwendbarkeit der vorgestellten Methoden an einem weiteren Beispielplan gezeigt werden. Es handelt sich dabei um ein Stockwerk des Gebäudes Inffeldgasse 16 des Campus der TU Graz.

5.1 Übergabe zur Visualisierung

Zunächst stellt sich die Frage, welches Framework zur Visualisierung eingesetzt werden soll. Es gibt eine Vielzahl von frei verfügbaren Bibliotheken zur Visualisierung von dreidimensionalen Szenen. Diese gliedern sich im Großen und Ganzen in Szenengraphenbibliotheken (z.B. [Opeb], [Coi] und [Opea]) und Bibliotheken zur Spielentwicklung (z.B. [OGR] und [Irr]). Die Darstellungsroutinen in Spieleengines basieren mittlerweile meist ebenfalls auf Szenengraphen.

Der Unterschied zwischen Szenengraphenbibliotheken und Bibliotheken zur Spielentwicklung ist wohl am ehesten beim Grad der Abstraktion von Darstellungsformen zu suchen. Bibliotheken zur Spielentwicklung ermöglichen meist direkten Zugriff auf die Funktionen der zur Darstellung verwendeten Grafikhardware. Generell verfolgen Spieleengines meist auch nicht den Ansatz, sämtliche Funktionalität in Form des Szenengraphen umzusetzen. Man könnte also von „schlankeren“ Implementierungen bei den Szenengraphen sprechen.

Für diese Arbeit wurde zur Umsetzung der Visualisierung die Bibliothek Irrlicht ([Irr]) ausgewählt. Diese ist ein Vertreter der Bibliotheken zur Spielentwicklung. Interessant an dieser Engine ist, dass für die Grafikausgabe verschiedene Module zur Verfügung stehen. Neben Backends für die größten Schnittstellen zur Grafikprogrammierung OpenGL und Direct3D, sind auch zwei Softwarerenderer enthalten. Die Unterschiede beim Ansprechen dieser Programmierschnittstellen sind für den Benutzer der Engine nicht sichtbar, da diese in entsprechenden Interfaces gekapselt sind.

Das Konzept zur Visualisierung sieht vor, dass stufenlos zwischen einer vereinfachten Gebäudeansicht und einer detaillierten Ansicht eines einzelnen Stockwerks gewechselt werden kann. Details wie Türen, Fenster und Wände sollen in der Gebäudeansicht

ausgeblendet werden, um die Darstellung visuell nicht zu überfrachten. Der fließende Übergang zwischen Gebäudeansicht und Stockwerkansicht soll ähnlich einer Explosionszeichnung umgesetzt werden. Alle Stockwerke bis auf ein gewähltes Stockwerk sollen beim Zoomen in Richtung Bildschirmrand verschoben werden, bis schließlich nur noch das gewählte Stockwerk zu sehen ist.

5.1.1 Transformation in einen Szenengraphen

Szenengraphen sind eine Möglichkeit zur Repräsentation von dreidimensionalen Szenen. Der Inhalt der Szene wird dabei durch einen Graphen festgelegt. Zur konkreten Beschreibung stehen verschiedene Knotentypen zur Verfügung. Die wichtigsten sind Knoten zur Beschreibung von Kameraperspektiven, Lichtquellen und geometrischen Körpern.

Eine wichtige Rolle spielt die hierarchische Gliederung im Szenengraphen. Meist sind mit den Knoten implizit auch geometrische Transformationen verknüpft. Werden einem Knoten weitere Knoten als Kinder untergeordnet, so werden diese ebenfalls nach der Vorgabe des Elternknotens transformiert. So lassen sich sehr einfach komplexe Objekte aus vielen kleinen Teilobjekten zusammenbauen. Auch die Animation einzelner Teile eines Objekts lassen sich auf diese Art sehr leicht umsetzen.

Die hierarchische Gliederung des Szenengraphen kann man sich auch in anderer Hinsicht zu Nutze machen. Sind Teile der Szene aufgrund der aktuellen Kameraperspektive nicht sichtbar, sollten diese auch nicht gezeichnet werden. Zur Überprüfung der Sichtbarkeit eines Objekts wird der entsprechende Knoten mit einem Begrenzungskörper versehen. In den meisten Fällen handelt es sich dabei um achsenparallele Quader, welche die Geometrie des Objekts vollständig umschließen. Mit einfachen, geometrischen Berechnungen kann sehr schnell festgestellt werden, ob ein Begrenzungskörper sichtbar ist, oder nicht. Ist der Begrenzungskörper nicht sichtbar, muss auch das Objekt nicht gezeichnet werden. Wird dieser Ansatz in der Hierarchie des Szenengraphen konsequent umgesetzt, kann die Sichtbarkeit komplexer Körper sehr schnell ermittelt werden. Zu diesem Zweck wird der Begrenzungskörper eines Knotens automatisch aus den Begrenzungskörpern der Kinderknoten gebildet.

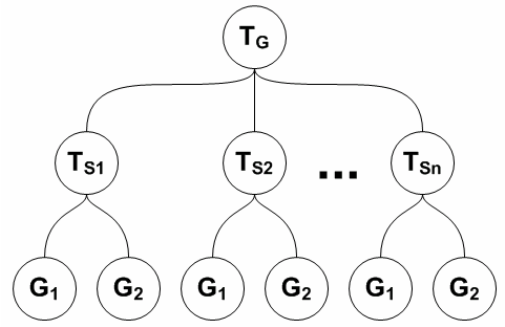


Abbildung 68: Szenengraph für die Visualisierung

Die Gliederung des Szenengraphen ergibt sich direkt aus dem Konzept für die Visualisierung (Abbildung 68). Die Wurzel ist ein Transformationsknoten (T_G), der das Gebäude ins Zentrum der Szene rückt. Darunter folgt für jedes Stockwerk jeweils ein weiterer Transformationsknoten (T_{S1}-T_{Sn}). Mit Hilfe dieser Transformationsknoten wird das explosionsartige Verschwinden der Stockwerke beim Zoomen realisiert. Schließlich folgt die Geometrie zu einem Stockwerk in zwei Ausführungen. Die vereinfachte Geometrie (G_1) wird für die Gebäudeansicht verwendet und die detaillierte Geometrie (G_2) für die Darstellung in der Stockwerksansicht. Die Geometrieknoten werden je nach Zoomstufe angezeigt oder versteckt.

5.1.2 Erzeugung von 3D-Modellen für die Objekte

Für die Darstellung müssen aus den Objekten im Gebäudemodell automatisch 3D-Modelle generiert werden können. Die Geometriedaten dieser Modelle werden dann den entsprechenden Knoten G_1 und G_2 des Szenengraphen zugewiesen.

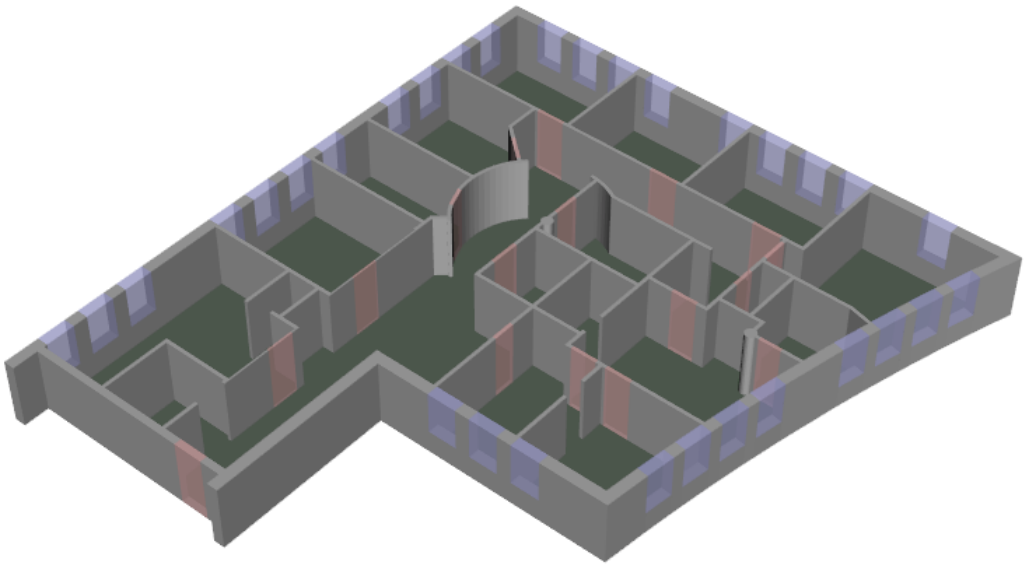


Abbildung 69: Beispiel für ein detailliertes 3D-Modell in der Stockwerkansicht

In Abbildung 69 ist ein Beispiel für ein detailliertes 3D-Modell in der Stockwerkansicht abgebildet. Sämtliche Geometrie wird über Extrusion der Polygone aus der Datenbank

erzeugt. Parameter für die Extrusion sind jeweils untere und obere Höhenbegrenzungen. Wände und Türen können in einem Schritt extrudiert werden. Die schematische Darstellung der Fenster wird in zwei Extrusionsschritten erreicht. Zunächst wird der Fenstersims bis zu einer gewissen Höhe wie eine Wand extrudiert. Darüber erfolgt in einem zweiten Schritt die Extrusion des halbtransparenten Fensters.

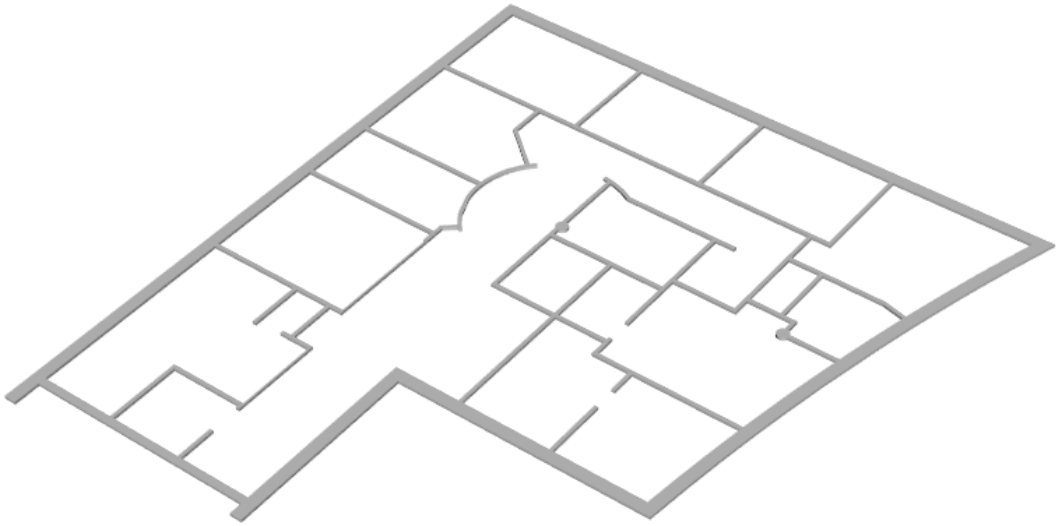


Abbildung 70: Beispiel für ein Umrissmodell in der Gebäudeansicht

Für das 3D-Modell in der Gebäudeansicht werden alle Objekte eines Stockwerks einfach flach extrudiert (siehe Abbildung 70).

Zum Rendering müssen die Geometriedaten als indizierte Dreieckslisten übergeben werden. Das heißt, es muss zunächst eine Liste von Eckpunkten generiert werden, welche dann von einer Dreiecksliste indiziert wird. Diese Dreiecksliste enthält jeweils drei Indizes für die Eckpunkte eines darzustellenden Dreiecks.

Zur Triangulierung der Polygone als Deckflächen der 3D-Modelle kommt wieder die Delaunay Triangulierung zum Einsatz. Das Problem bei der Triangulierung von Polygonen ist, dass gewisse Kanten auf jeden Fall Teil der Triangulierung sein müssen.

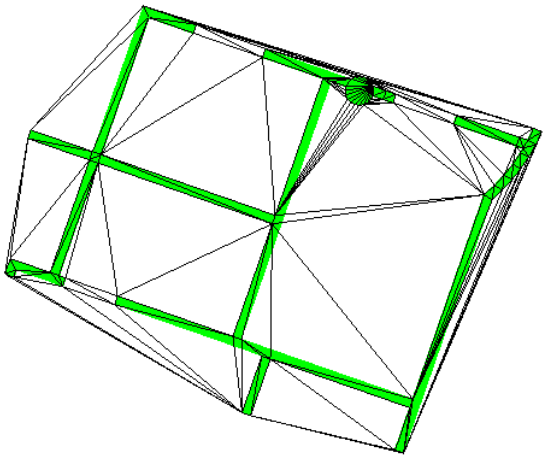


Abbildung 71: Fehlende Randkanten bei der Triangulierung nach Delaunay

Wird dies nicht sichergestellt, kann es sein, dass die Ränder des Polygons nicht korrekt repräsentiert sind (siehe Abbildung 71). Zur korrekten Triangulierung von Polygonen kommt eine Constrained Delaunay Triangulierung zum Einsatz. Dem Algorithmus wird dabei eine Liste jener Kanten übergeben, die auf jeden Fall Teil der Triangulierung sein sollen. Die resultierende Triangulierung erfüllt meistens die Umkreisbedingung nicht und ist daher streng genommen auch keine Delaunay Triangulierung. Die eingesetzte Geometriebibliothek CGAL enthält mit der Klasse `CGAL::Constrained_Delaunay_triangulation_2` eine Implementierung der notwendigen Algorithmen.

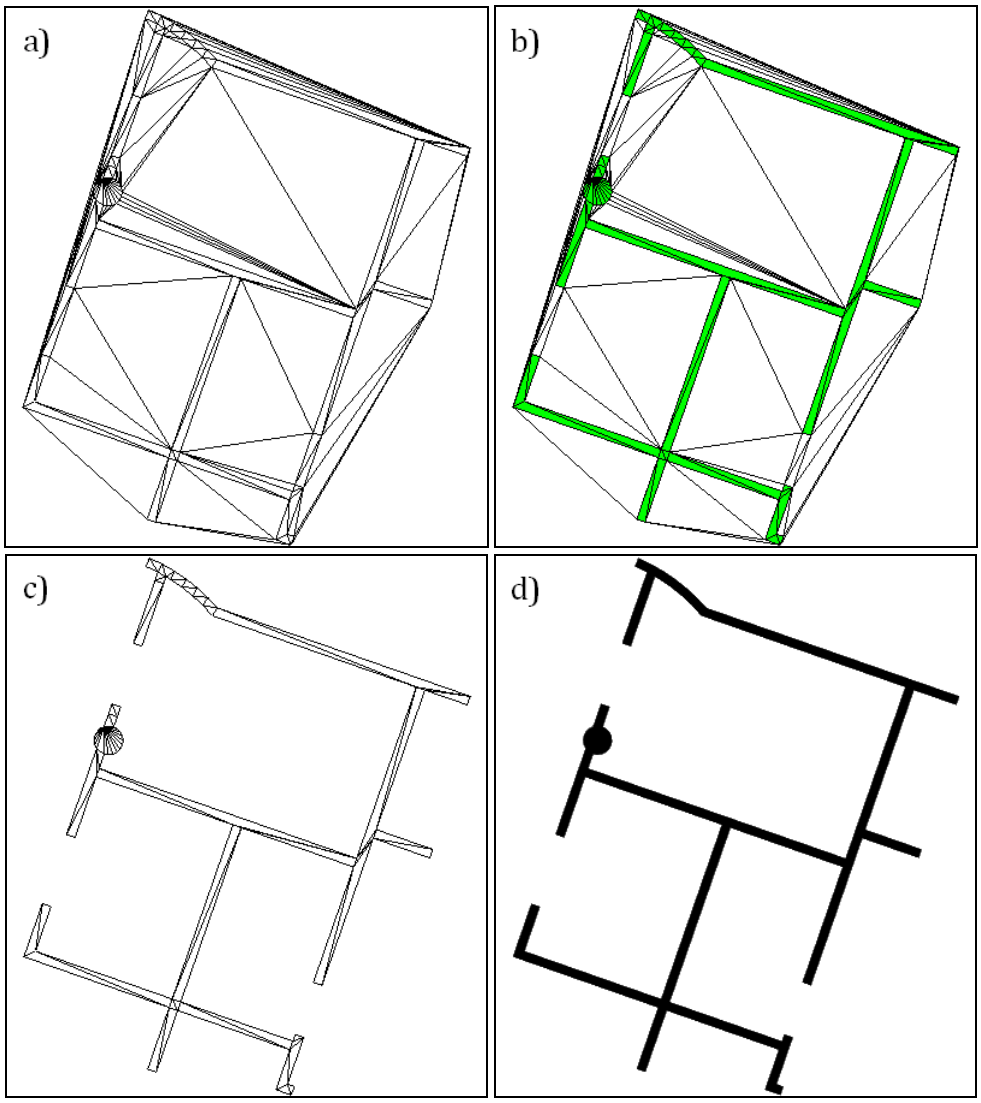


Abbildung 72: Triangulieren von Polygonen mit Constrained Delaunay

In Abbildung 72 ist in vier Bildern die schrittweise Triangulierung eines Polygons zu sehen. Die Triangulierung der Punktmenge mit den einschränkenden Kanten ist in Bild a) abgebildet. In Bild b) wurde diese Triangulierung mit dem Zielpolygon hinterlegt. In Bild c) wurden jene Dreiecke verworfen, deren Schwerpunkte außerhalb des Polygons liegen. Die Überprüfung der Schwerpunkte wurde mit der Methode `has_on_unbounded_side` der Klasse `CGAL::Polygon_2` umgesetzt. Das Ergebnispolygon in Flächendarstellung ist schließlich in Bild d) zu sehen.

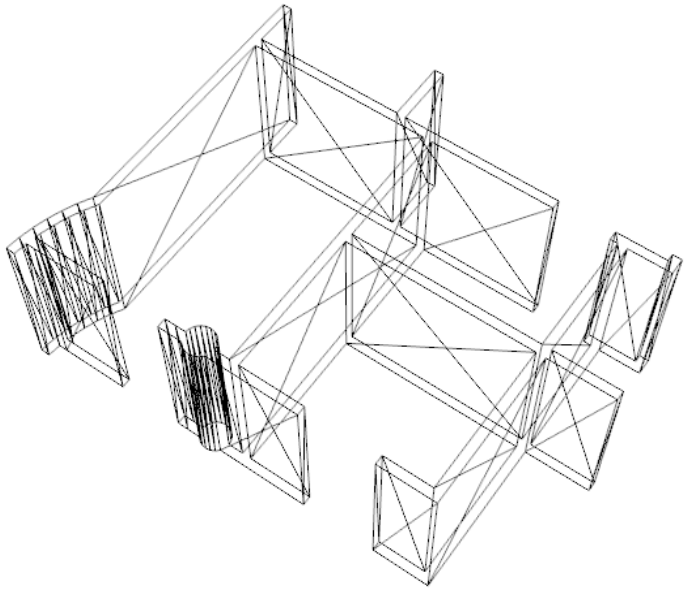


Abbildung 73: Triangulierung der Seitenflächen durch Einfügen von Dreiecken

Die Triangulierung der Seitenflächen erfolgt durch manuelles Erzeugen von Dreiecken. Dazu wird einfach über die Begrenzungslinien des Polygons iteriert und zwei Dreiecke für jede Kante erzeugt (siehe Abbildung 73).

5.1.3 Benutzerschnittstelle für die Navigation im Modell

Die Navigation innerhalb des Modells erfolgt mit Hilfe der Maus. Durch einen Klick mit der rechten Maustaste kann die Ansicht um den angeklickten Ort im Modell zentriert werden (bezieht sich immer auf das ausgewählte Stockwerk). Die Ansicht kann durch Mausgesten bei gedrückter, linker Maustaste gedreht werden. Der Neigungswinkel wird dabei sinnvoll begrenzt (vergleiche [HD08]). Das Modell kann mit dem Mausrad stufenlos gezoomt werden. Eine Besonderheit beim Zoomen ist, dass die Ansicht zusätzlich leicht in die Richtung verschoben wird, die mit dem Mauszeiger angezeigt wird. Bei gedrückter STRG-Taste kann mit dem Mausrad das aktuelle Stockwerk gewechselt werden.

5.2 Endergebnis in Form eines größeren Beispielmodells

Im Folgenden soll nun ein größeres Beispielmodell als Endergebnis der Arbeit präsentiert werden. Das Modell wurde aus ausgewählten Stockwerkplänen des Bauprojekts Wien Mitte erstellt. Der Abschnitt gliedert sich in Erfassung, Visualisierung und Dokumentation ausgewählter Datenbankabfragen über das erfasste Modell.

5.2.1 Erfassung des Beispielmodells

In diesem Abschnitt ist die Vorgehensweise zur Erfassung eines Gebäudemodells beschrieben. Die Schritte müssen für jedes Stockwerk einzeln durchgeführt werden. In einem ersten Schritt werden durch Ausblenden von Schichten alle Informationen des

Plans versteckt, die nicht unmittelbar für das Gebäudemodell verwendet werden können. In der derzeitigen Ausbaustufe des Systems bleiben also Wände, Türen und Fenster übrig.

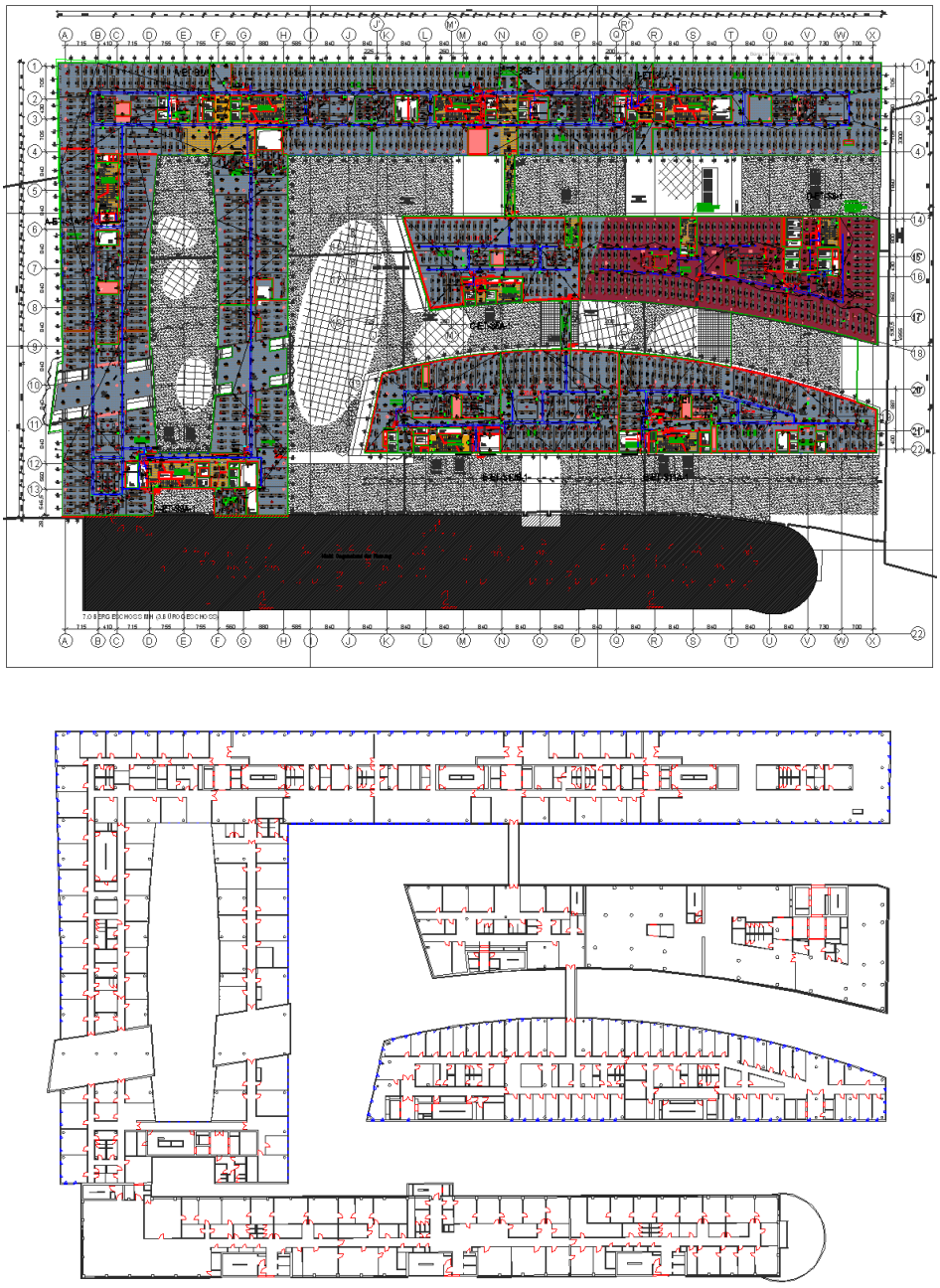


Abbildung 74: Filtern der Plandaten durch Verstecken von Schichten

In Abbildung 74 ist die Auswirkung der Filterung auf einen größeren Plan zu sehen. Im linken Bild sind alle Schichten des Plans eingeblendet. Für das rechte Bild wurden alle nicht relevanten Schichten ausgeblendet. Um die Darstellung zu verbessern wurden die Schichten für Türen und Fenster im rechten Bild farblich hervorgehoben. Mit Hilfe des VBA Exportskripts müssen die Daten der sichtbaren Schichten nun in eine XML Datei nach Plandatenschema exportiert werden.

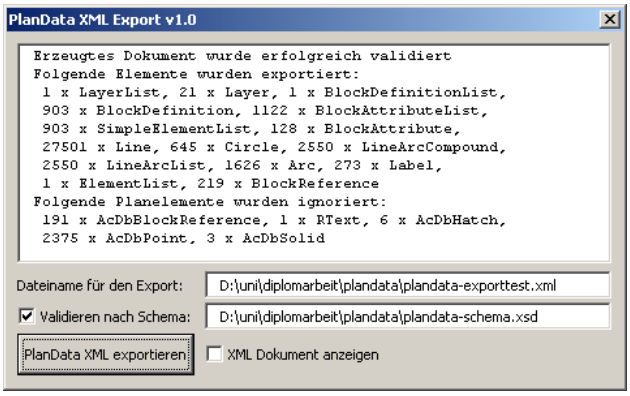


Abbildung 75: Benutzeroberfläche für das VBA Exportskript

Das Exportskript wird mit Hilfe einer kleinen, grafischen Benutzeroberfläche gestartet (Abbildung 75). Die Oberfläche erlaubt unter anderem das Wählen einer Ausgabedatei und das Validieren der exportierten Daten nach einer wählbaren XML Schemadefinition.

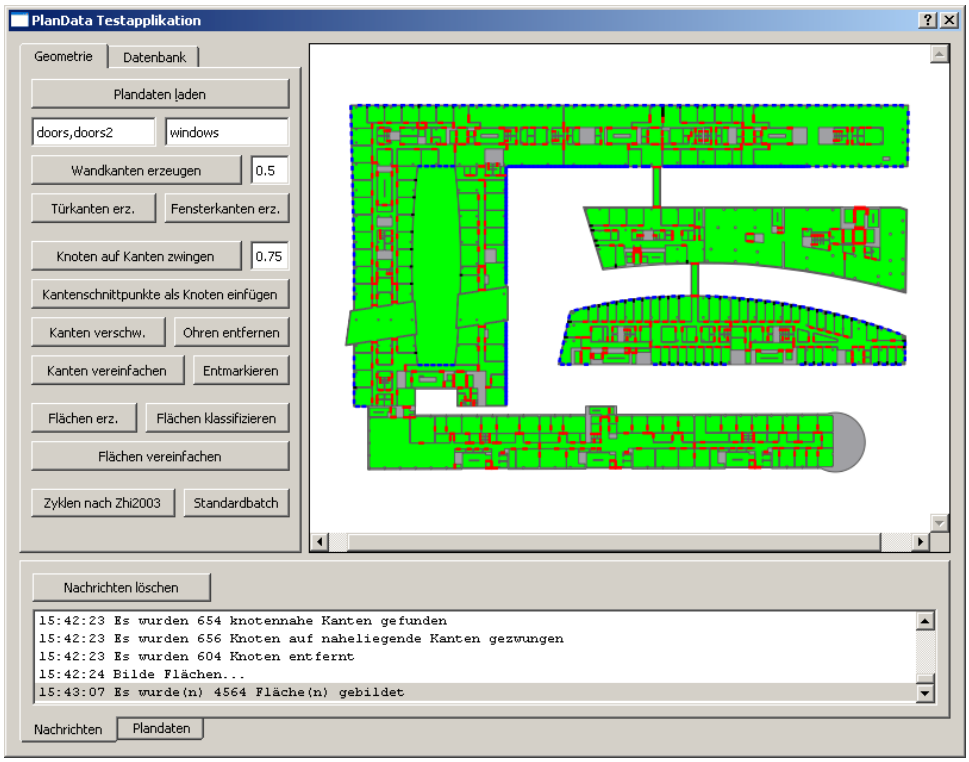


Abbildung 76: Benutzeroberfläche nach Anwendung der Standardoperationen

Mit der Oberfläche zur Anwendung der Operationen (Abbildung 76) wird nun zunächst versucht die Plangeometrie zu bereinigen und die Flächenanalyse durchzuführen. Der Benutzer kann die Operationen zu Testzwecken einzeln anwenden, oder in der nachfolgend beschriebenen Standardreihenfolge automatisch anwenden lassen.

Die Standardreihenfolge ist folgendermaßen definiert:

1. Wandkanten aus Plandaten erzeugen (siehe Kapitel 3.3.2.1: Kanten aus Plandaten erzeugen)
2. Knoten auf nahe liegende Kanten zwingen (siehe Kapitel 3.3.2.2)
3. Kantenschnitte entfernen (siehe Kapitel 3.3.2.4)
4. Kanten verschweißen (siehe Kapitel 3.3.2.5)
5. Türkanten aus Plansymbolen erzeugen (siehe Kapitel 3.3.2.6: Spezialkanten aus Plansymbolen erzeugen)
6. Fensterkanten aus Plansymbolen erzeugen (siehe Kapitel 3.3.2.6: Spezialkanten aus Plansymbolen erzeugen)
7. Kanten vereinfachen (siehe Kapitel 3.3.2.3)
8. Flächen aus Kanten erzeugen (siehe Kapitel 3.4.2.1)
9. Flächen durch Spezialkanten klassifizieren (siehe Kapitel 3.4.2.2)
10. Flächen vereinfachen (siehe Kapitel 3.4.2.3)

Ist eine korrekte Flächenanalyse aufgrund von Fehlern in der Geometrie des CAD-Plans, oder fehlerhaften Transformationen von Plansymbolen nicht möglich, muss der CAD-Plan entsprechend verändert werden. Nach der Korrektur müssen die Plandaten erneut exportiert und in der Benutzeroberfläche zur Anwendung der Operationen bearbeitet werden. Bei einem Plan welcher der Größe und Qualität des in Abbildung 74 abgebildeten Plans entspricht, muss ein in etwa ein halber Manntag (4 Arbeitsstunden) Arbeitszeit für die Korrektur von Fehlern eingerechnet werden. Es handelt sich dabei um einen Erfahrungswert, der je nach Qualität und Größe des Plans natürlich in beide Richtungen abweichen kann.

Verläuft die semantische Analyse des CAD-Plans zufrieden stellend (wird mit der Darstellung in der Benutzeroberfläche zur Anwendung der Operationen überprüft) können die Daten für ein Stockwerk in die Datenbank importiert werden.

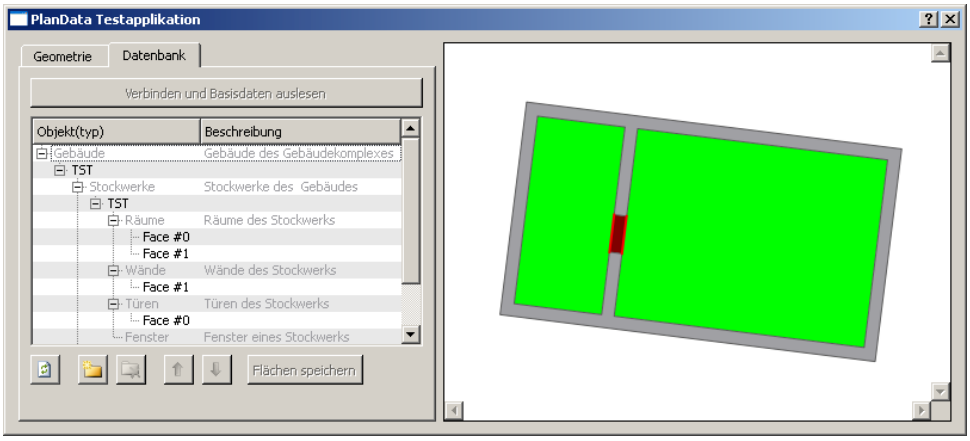


Abbildung 77: Benutzeroberfläche zum Editieren der Datenbank

Das Anlegen der Gebäudestruktur und das Importieren der Daten erfolgt ebenfalls mit der selben Benutzeroberfläche (Abbildung 77).

Nachdem die eben beschriebene Vorgehensweise für jedes Stockwerk des Gebäudes wiederholt wurde, ist das semantische Gebäudemodell vollständig in der Datenbank abgespeichert. Die Applikation zur Visualisierung des Gebäudemodells bezieht alle notwendigen Daten direkt aus der Datenbank.

5.2.2 Visualisierung des Beispielmodells

Es folgen einige Bildschirmfotos, die mit der Applikation zur Visualisierung aufgenommen wurden. In Abbildung 78 ist die Gebäudeansicht zu sehen, in der nur die Umrisse eines Stockwerks visualisiert werden. Zoomt der Benutzer mit dem Mausrad etwas näher zum gewählten Stockwerk, verschwinden die übrigen Stockwerke langsam in Richtung Bildschirmrand (Abbildung 79).

Der Übergang vom Umrissmodell zum detaillierten Modell des Stockwerkes ist durch Skalieren des Umrissmodells in der Höhe und stufenloses Einblenden der Räume umgesetzt. Dies ist an der halbtransparenten Darstellung der Räume ebenfalls aus Abbildung 79 ersichtlich.

In Abbildung 80 ist die Detailansicht eines Stockwerks zu sehen. Objekte wie Türen und Fenster sind bereits deutlich erkennbar. Im letzten Bild (Abbildung 81) wird schließlich ein kleiner Ausschnitt eines Stockwerks gezeigt. Türen, Fenster und weitere Details wie Säulen sind sehr gut sichtbar.

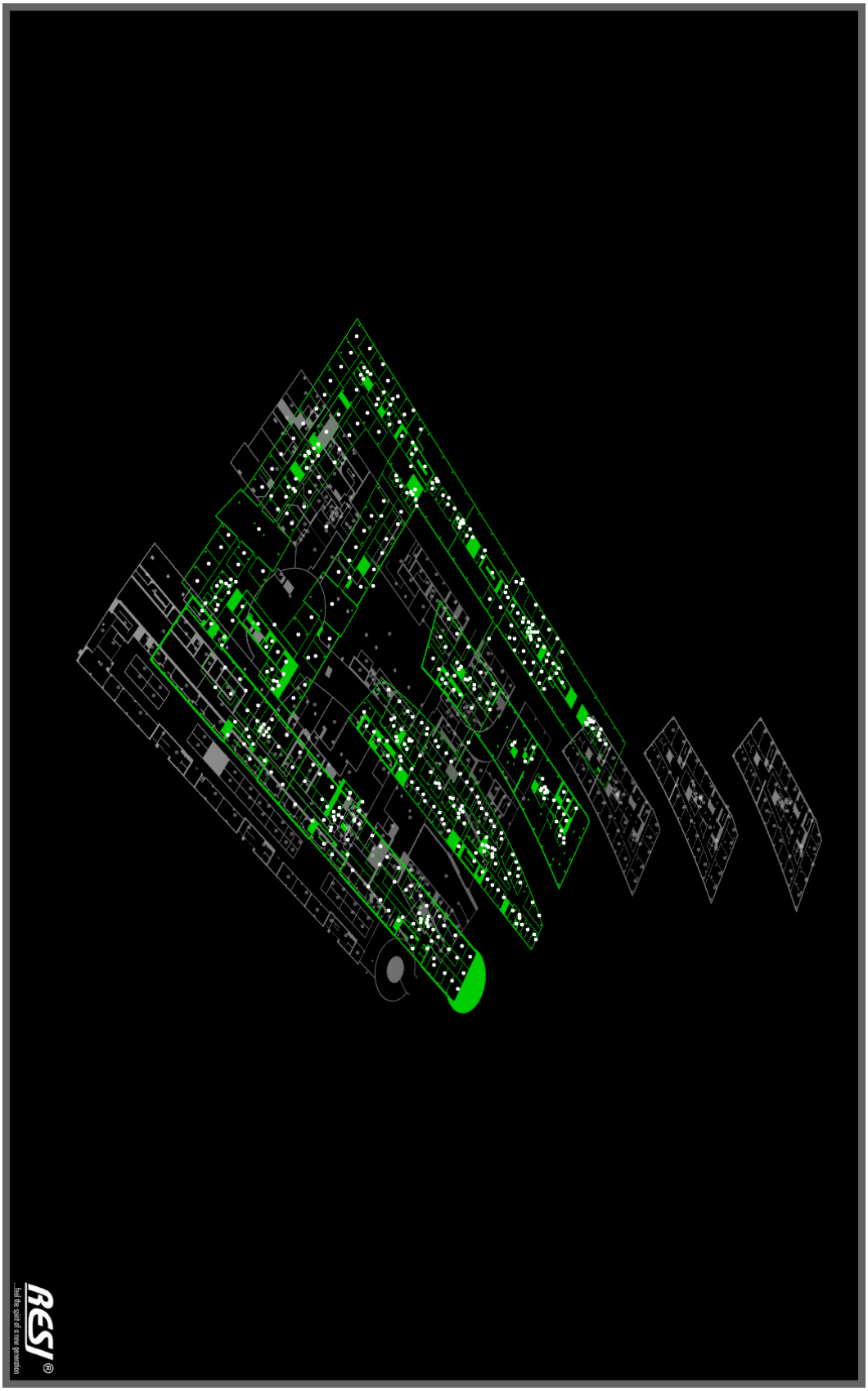


Abbildung 78: Ansicht des gesamten Gebäudes

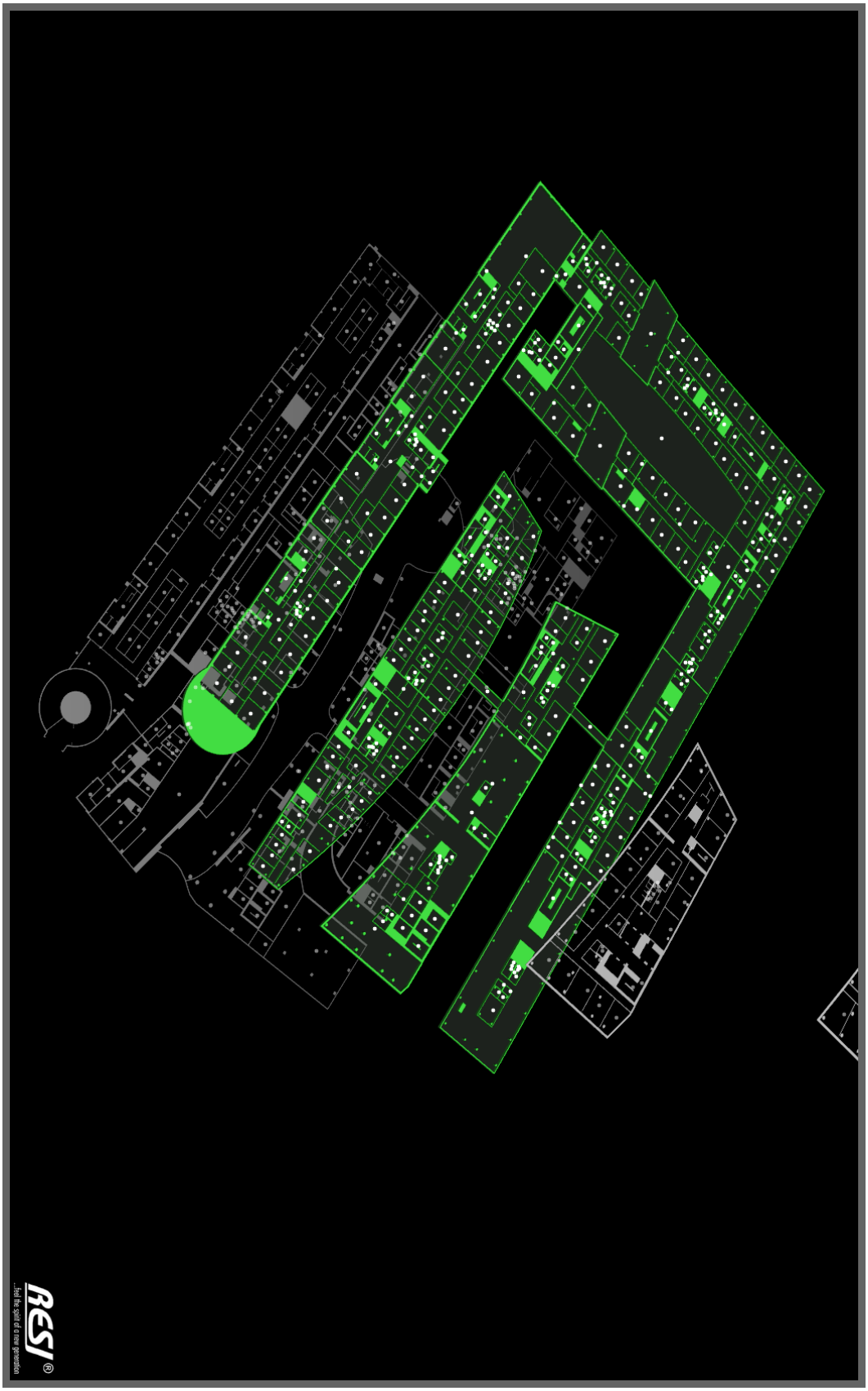


Abbildung 79: Heranzoomen an das gewählte Stockwerk

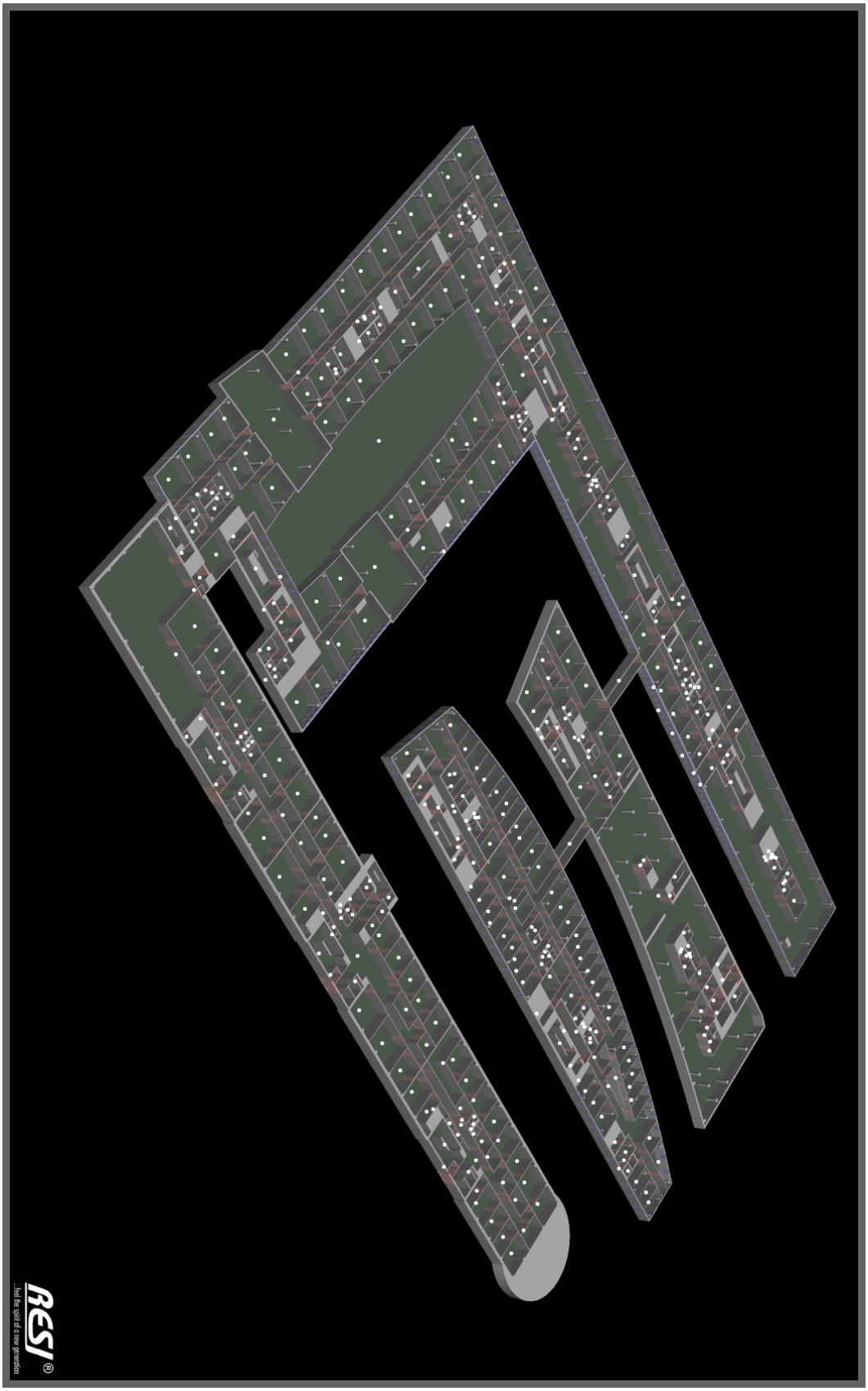


Abbildung 80: Detailansicht zu einem Stockwerk des Gebäudes

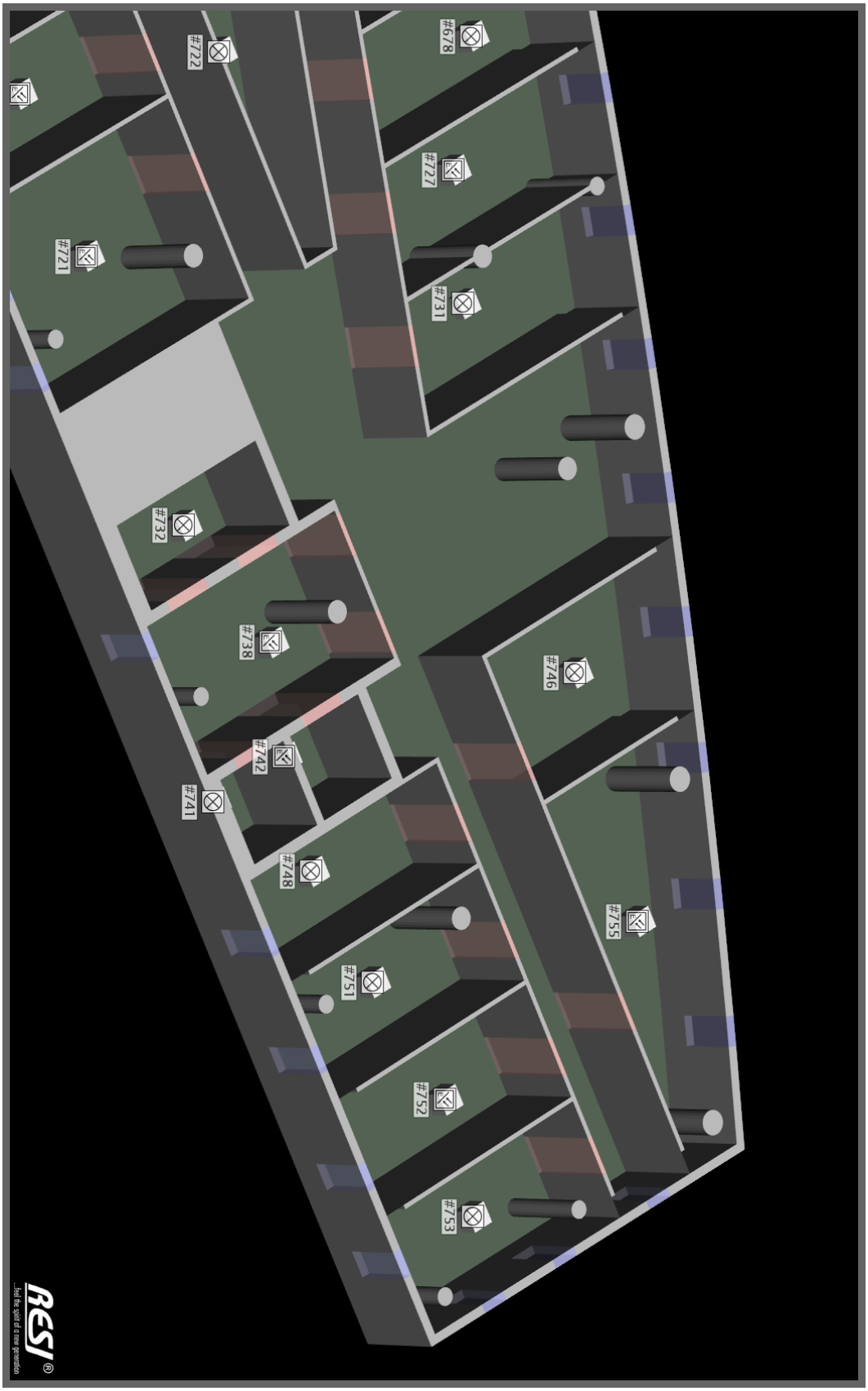


Abbildung 81: Detailansicht einzelner Räume eines Stockwerks

5.2.3 Datenbankabfragen über das Beispielmodell

Um einen kurzen Einblick in die Möglichkeiten der relationalen Datenbank zu bieten, werden nachfolgend noch zwei Beispiele für Datenbankabfragen über das Beispielmodell dokumentiert.

```
1:    SELECT count( * ) AS "DoorCount" FROM
2:    (
3:      SELECT * FROM
4:      (
5:        SELECT "Objects"."ObjectId", "ObjectTypes"."Name" FROM "Objects"
6:          INNER JOIN "ObjectTypes"
7:            ON "Objects"."ObjectTypeId" = "ObjectTypes"."ObjectTypeId"
8:      ) AS "ViewOnObjects"
9:      WHERE
10:       "ViewOnObjects"."Name" = 'Door'
11:   ) AS "DoorObjects"
```

Abbildung 82: SQL Abfrage zum Zählen aller Türen im Gebäudemodell

In Abbildung 82 ist eine Abfrage zum Zählen aller im Gebäudemodell repräsentierten Türen abgebildet. Mit der `SELECT` Anweisung in den Zeilen 5-7 wird zunächst eine temporäre Tabelle gebildet. Die `AS` Anweisung in Zeile 8 bezeichnet diese temporäre Tabelle mit dem Namen `ViewOnObjects`. Wie der Name vermuten lässt, wird in der Tabelle eine Ansicht auf alle Objekte des Gebäudemodells erzeugt. Die Besonderheit der Ansicht ist, dass der Typ eines Objekts durch eine Zeichenkette repräsentiert wird. Diese Typbezeichnung wird mit Hilfe der `INNER JOIN` Anweisung in Zeile 6 über den Fremdschlüssel `ObjectTypeId` aus der Tabelle `ObjectTypes` extrahiert.

Mit der nächst äußeren `SELECT` Anweisung in Zeile 3 werden aus der Ansicht jene Objekte ausgewählt, bei denen die Typbezeichnung dem Wert `Door` entspricht. An dieser Stelle wird auch klar, warum zuvor die Ansicht gebildet wurde: Ohne Hinzufügen der Typbezeichnung hätte für die Abfrage die Datenbankinterne `ObjectTypeId` des Objekts überprüft werden müssen. Das Ergebnis dieser `SELECT` Anweisung wird ebenfalls in eine temporäre Tabelle namens `DoorObjects` gespeichert (Zeile 11).

Die `SELECT` Anweisung in Zeile 1 berechnet mit Hilfe der Aggregatfunktion `count` schließlich die Anzahl der Zeilen in der temporären Tabelle `DoorObjects`. Die Anzahl der Zeilen entspricht natürlich genau der Anzahl, der im Modell repräsentierten Türobjekte.

```
1:    SELECT sum( ST_Area( "RoomObjects"."Polygon" ) ) AS "RoomArea" FROM
2:    (
3:      SELECT * FROM
4:      (
5:        SELECT "Objects"."ObjectId", "Objects"."Polygon",
6:          "ObjectTypes"."Name" FROM "Objects"
7:          INNER JOIN "ObjectTypes"
8:            ON "Objects"."ObjectTypeId" = "ObjectTypes"."ObjectTypeId"
9:      ) AS "ViewOnObjects"
10:     WHERE
11:       "ViewOnObjects"."Name" = 'Room'
12:   ) AS "RoomObjects"
```

Abbildung 83: SQL Abfrage zur Berechnung der Gesamtfläche aller Räume

In Abbildung 83 ist eine Abfrage zur Berechnung der Gesamtfläche aller Räume im Gebäudemodell abgebildet. Diese Abfrage unterscheidet sich von der vorherigen im abgefragten Objekttyp (Zeile 11) und den verwendeten Aggregat- bzw. Berechnungsfunktionen (Zeile 1). Ein weiterer Unterschied ist, dass die Ansicht der Objekte um die Spalte Polygon erweitert wurde (Zeile 5). In dieser Spalte ist das Polygon abgespeichert, das der Grundfläche des Raumobjekts in der Datenbank entspricht. Die bereits in Abschnitt 4.2.1.2 verwendete Funktion `ST_Area` berechnet die Fläche eines Polygons. Mit Hilfe der Aggregatfunktion `sum` wird schließlich die Summe über alle Einzelflächen gebildet.

Es folgt eine Zusammenstellung interessanter Werte bezüglich des Gebäudemodells, die alle mit den oben beschriebenen, oder ähnlichen Abfragen ermittelt wurden:

- Das Gebäudemodell umfasst 5 Stockwerke, 871 Räume, 1098 Türen und 305 Fenster. Die Wände sind in 1241 disjunkten Polygonen repräsentiert.
- Die Gesamtfläche aller repräsentierten Räume beträgt ca. 30.945m^2.
- Insgesamt sind im Modell 3521 Objekte repräsentiert.
- 3358 Objekte sind durch Polygone mit insgesamt 54891 Eckpunkten repräsentiert.
- Die Repräsentation des Gebäudemodells in der relationalen Datenbank benötigt ca. 11MB Speicherplatz.

5.3 Verifikation der Methoden anhand eines weiteren Modells

Die vorgestellten Methoden und Vorgehensweisen zur Gewinnung des Modells wurden im Wesentlichen anhand einer Menge von Plänen entworfen, die sich in der Struktur sehr ähnlich waren. Mit Struktur ist sowohl die allgemeine Qualität der repräsentierten geometrischen Merkmale als auch die Einteilung dieser in die semantischen Schichten gemeint. Um die Eignung der vorgestellten Methoden als allgemeine Vorgehensweise zur Erfassung von Gebäudemodellen aus verschiedensten CAD-Plänen zu verifizieren, wurde ein weiteres Beispielmodell erstellt.

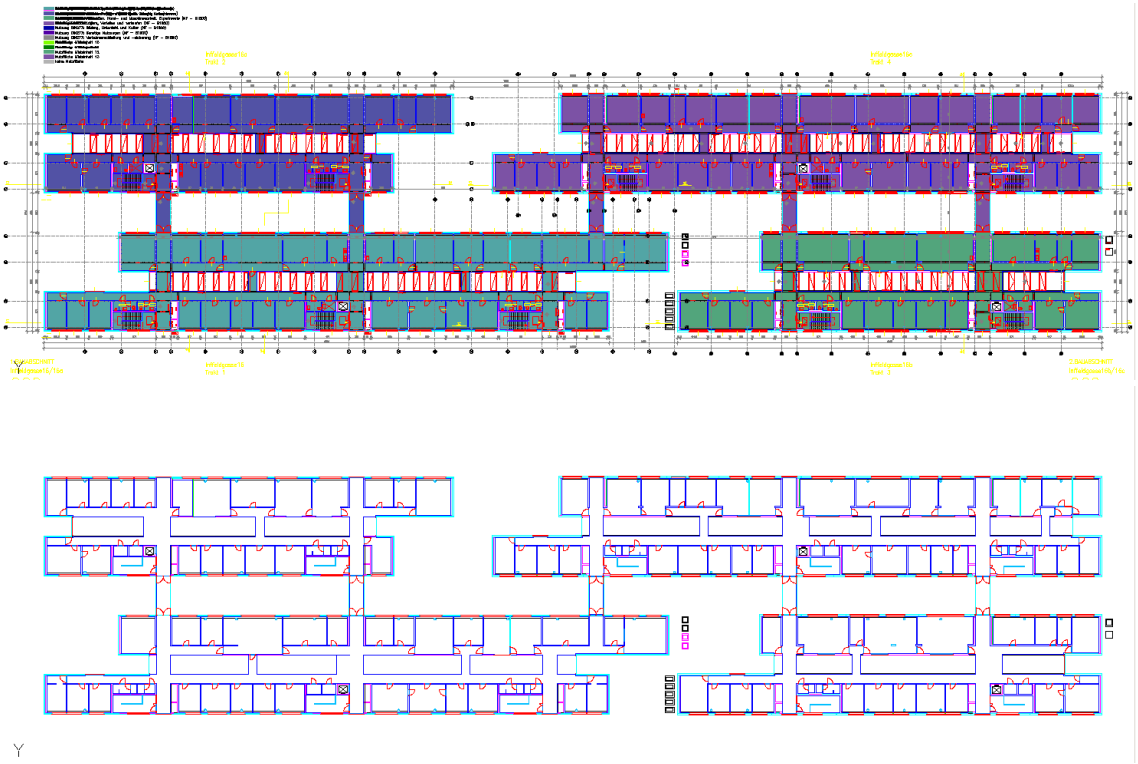


Abbildung 84: CAD-Plan Inffeldgasse 16 nach Verstecken irrelevanter Schichten

Modelliert wurde ein Stockwerk des Gebäudes Inffeldgasse 16 der TU Graz (siehe Abbildung 84). Dieses Gebäude wurde gewählt, weil sich das Modell unter Umständen auch für andere Forschungsbereiche als nützlich erweisen könnte. Das Flächenmodell könnte beispielsweise als Referenzmodell für die Lokalisierung von autonomen Robotern dienen.

Im Folgenden werden zunächst die Probleme erläutert, die bei der Erfassung des Modells aufgetreten sind. Abschließend folgen, analog zum ersten Beispielmodell, einige Daten und Bilder zum erfassten Gebäudemodell.

5.3.1 Probleme bei der Erfassung des Modells

Die konkrete Beschaffenheit von CAD-Plänen ist von Projekt zu Projekt oft sehr unterschiedlich. Bei der Konstruktion werden andere Planelemente eingesetzt und die Zuordnung zu den Schichten erfolgt meist nicht nach einem einheitlichen Schema. Vor allem Plansymbole für Türen und Fenster sind projektabhängig oft grundverschieden. Bei der Erstellung des Gebäudemodells können solche Unterschiede natürlich zu Problemen führen. Für den vorliegenden Fall der Inffeldgasse 16, sollen die Probleme nachfolgend genauer untersucht werden.

Dynamische / parametrierbare Blöcke: In den vorliegenden Plänen wird ein spezieller Typ von Blockdefinitionen verwendet, der bisher für den XML-Exporter nicht berücksichtigt wurde (siehe Kapitel 3.1: „Informationsgehalt von CAD-Plänen“). Es handelt sich dabei um Blockdefinitionen, die bei der Instanzierung parametriert werden können. Beispielsweise wird ein allgemeiner, dynamischer Block für Türen entworfen, der dann für die konkreten Blockreferenzen unterschiedlich parametriert wird. So können Türen unterschiedlicher Größe mit Hilfe einer einzigen Blockdefinition modelliert werden. Dynamische Blöcke können mit einer Funktion der CAD-Applikation automatisch in geometrische Grundelemente umgewandelt werden, und stellen so kein Problem für die weitere Verarbeitung dar.

Verwendung von Ellipsen in Türsymbolen: Für die Darstellung der Türsymbole in den Plänen, wird in manchen Fällen kein Kreisbogen, sondern ein Viertel einer Ellipse verwendet. Ellipsen wurden in der ursprünglichen Form des XML-Exporters ebenfalls nicht als zu exportierende Elemente berücksichtigt (siehe ebenfalls Kapitel 3.1: „Informationsgehalt von CAD-Plänen“).

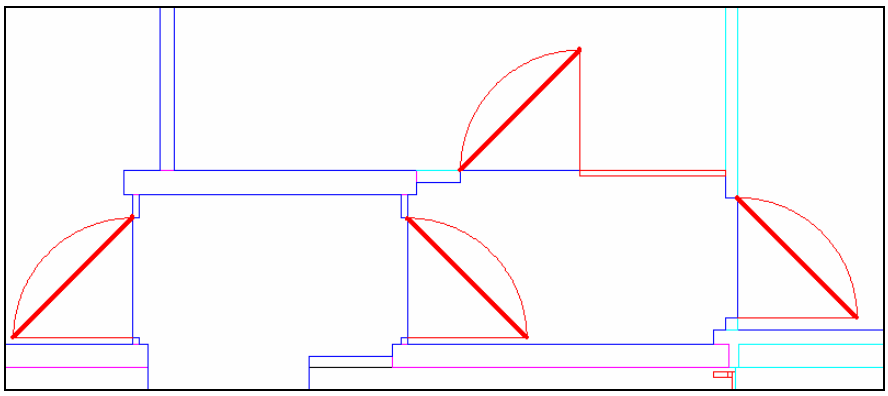


Abbildung 85: Ersetzen von Ellipsen durch einfache Liniensegmente

Für diesen speziellen Fall wurde der XML-Exporter dahingehend erweitert, dass solche „Viertelellipsen“ in einfache Liniensegmente umgewandelt werden, die Start- und Endpunkt verbinden (siehe dicke rote Linien in Abbildung 85). Diese Darstellung ist für die korrekte Transformation der Türsymbole vollkommen ausreichend.

Vereinfachung der Türsymbole: Die ursprüngliche Beschaffenheit der Türsymbole im Plan hat eine korrekte Transformation in Spezialkanten verhindert.

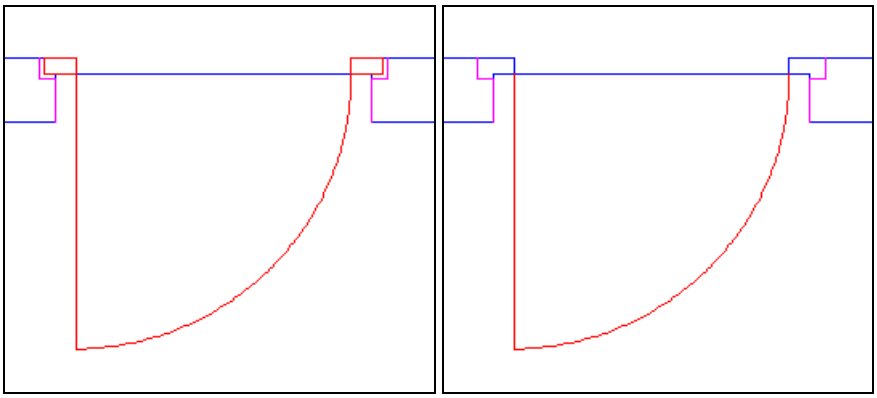


Abbildung 86: Vereinfachung der Türsymbole Inffeldgasse 16

Der Grund dafür ist aus Abbildung 86 ersichtlich. Im linken Bild ist die ursprüngliche Darstellung des Symbols zu sehen, die auch den Türstock beinhaltet. Würden man auf diese Darstellung den in Kapitel 3.3.2.6 („Spezialkanten aus Plansymbolen erzeugen“) beschriebenen Algorithmus anwenden, würden stets vier relevante Knoten vom Grad zwei gefunden werden. Der Algorithmus könnte aus diesen Knoten keine gültigen Spezialkanten für das Symbol ableiten.

Da die Türstöcke im Plan stets durch Linienzüge, oder sehr kurze Einzellinien repräsentiert sind, können diese mit einer eingebauten Funktion der CAD-Applikation sehr leicht entfernt werden. Die Funktion ermöglicht das Auswählen von Elementen im Plan nach bestimmten Kriterien. In diesem Fall sind einerseits die entsprechende Schicht und andererseits der Typ (Linienzug, Einzellinie) bzw. die Länge von Einzellinien als Suchkriterien heranzuziehen. Die ausgewählten Elemente können dann leicht aus dem Plan entfernt werden. Nach der Bereinigung der Symbole von den Türstöcken (siehe auch rechtes Bild in Abbildung 86) ist die Transformation nach der vorgestellten Methode problemlos möglich.

Transformation der Fenstersymbole: Bei den Fenstersymbolen gestaltet sich die Situation bei den vorliegenden Plänen schwieriger.

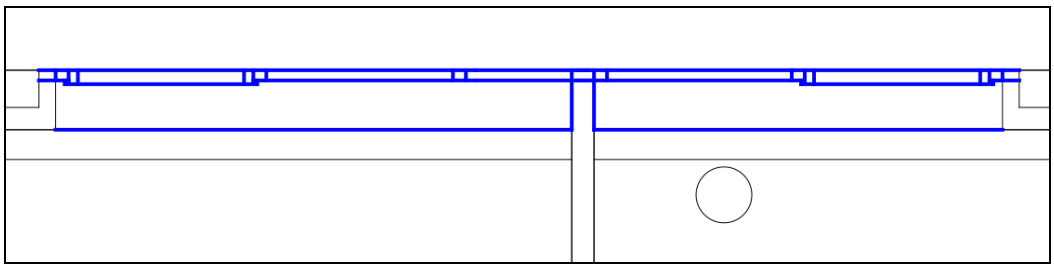


Abbildung 87: Problemtisches Fenstersymbol im Plan Inffeldgasse 16

Analog zu den Problemen mit Türsymbolen werden auch für die Zusammenhangskomponenten der Fenstersymbole oft zu viele kantennahe Knoten gefunden (vergleiche Kapitel 3.3.2.6 „Spezialkanten aus Plansymbolen erzeugen“). Es stellt sich heraus, dass die Transformation erfolgreich durchgeführt werden kann, wenn nur Knoten mit Grad eins für die Spezialkanten herangezogen werden (siehe Abbildung 87).

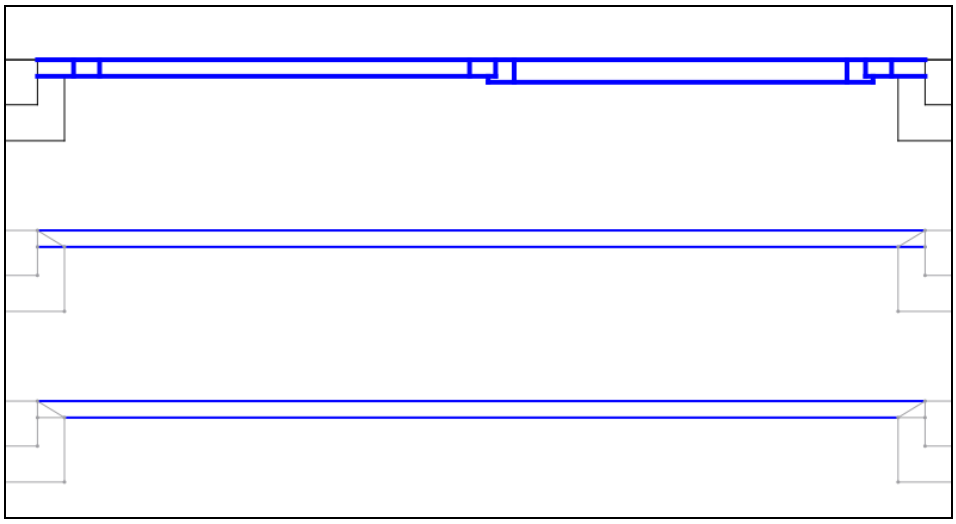


Abbildung 88: Entfernen von zu kurzen Spezialkanten aus dem ungerichteten Graphen

Beim Einfügen der Spezialkanten für die Fenstersymbole werden in manchen Fällen sehr kurze Spezialkanten erzeugt. Dies ergibt sich aus der höheren geometrischen Komplexität der Symbole in den vorliegenden Plänen (siehe Abbildung 88). Um dieses Problem zu beheben, werden Spezialkanten deren Länge unter einem gewissen Schwellwert liegt, wieder in herkömmliche Wandkanten umgewandelt.

Das letzte Problem in Bezug auf die Transformation der Fenstersymbole betrifft die Klassifikation der Flächen (siehe Kapitel 3.4.2.2: „Flächen durch Spezialkanten klassifizieren“). Wiederum aufgrund der erhöhten geometrischen Komplexität der Fenstersymbole, kann es vorkommen, dass eine Fensterfläche an mehr als zwei Wand- und Fensterkanten anschließt. Der Klassifikator wurde so umformuliert, dass eine Fensterfläche an zwei oder mehr Wand- und Fensterkanten anschließen kann.

Der Klassifikator für Raumflächen musste zusätzlich dahingehend geändert werden, dass Fensterkanten alleine keine ausreichende Bedingung für die Klassifikation als Raum mehr darstellen. Ohne diese Änderung wurden einzelne, unterteilte Wandflächen fälschlicherweise als Raumflächen klassifiziert. Nach diesen Modifikationen konnten alle Flächen im Modell korrekt klassifiziert werden.

Modellierung freier Innenflächen: Aufgrund der Architektur des Gebäudes Inffeldgasse 16 ergeben sich einige eingeschlossene Flächen, die nicht als Räume, oder Wände klassifiziert werden dürfen.

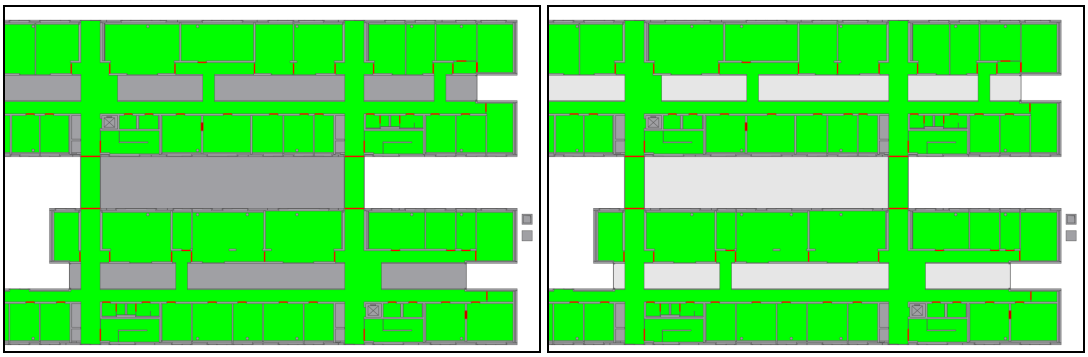


Abbildung 89: Manuelle Klassifikation von Innenflächen

Solche Flächen sind aus Abbildung 89 ersichtlich. Im rechten Bild wurde die Klassifikation der Innenflächen als Wand manuell rückgängig gemacht. Für diesen Vorgang wurde die Benutzeroberfläche zur Anwendung der Operationen dahingehen erweitert, dass einzelne Flächen ausgewählt und deren Klassifikation aufgehoben werden kann. Beim Importieren in die Datenbank werden nicht klassifizierte Flächen ignoriert und sind deshalb auch nicht im endgültigen Modell repräsentiert.

5.3.2 Ergebnis der Modellierung

Die Probleme bei der Modellierung haben gezeigt, dass die Arbeitsweise der vorgestellten Operationen durch Parameter von außen fein steuerbar sein muss, um Pläne unterschiedlicher Herkunft erfassen zu können. Neben unterschiedlicher Parametrierung kann, je nach Ausprägung der Symbole, auch die Implementierung einer neuen Operation erforderlich sein. Der interaktive Schritt zur Klassifikation von freien Innenflächen muss ebenfalls noch automatisiert werden. Um ein flexibel einsetzbares System zu erhalten, sollte die Reihenfolge in der die Operationen angewendet werden konfigurierbar sein (vergleiche Standardreihenfolge der Operationen in Kapitel 5.2.1, "Erfassung des Beispielmodells“).

Abgesehen von den beschriebenen Schwachstellen bei der Anwendung des Systems, konnte ein einwandfreies Modell des Gebäudes erzeugt werden.

Zunächst einige Zahlen zum Modell:

- Das Gebäudemodell umfasst 157 Räume, 172 Türen und 112 Fenster. Die Wände sind in 307 disjunkten Polygonen repräsentiert.
- Die Gesamtfläche aller repräsentierten Räume beträgt ca. 3.373m^2.
- Insgesamt sind im Modell 750 Objekte repräsentiert.

Abschließend folgen einige Bilder von der Visualisierung des Modells.

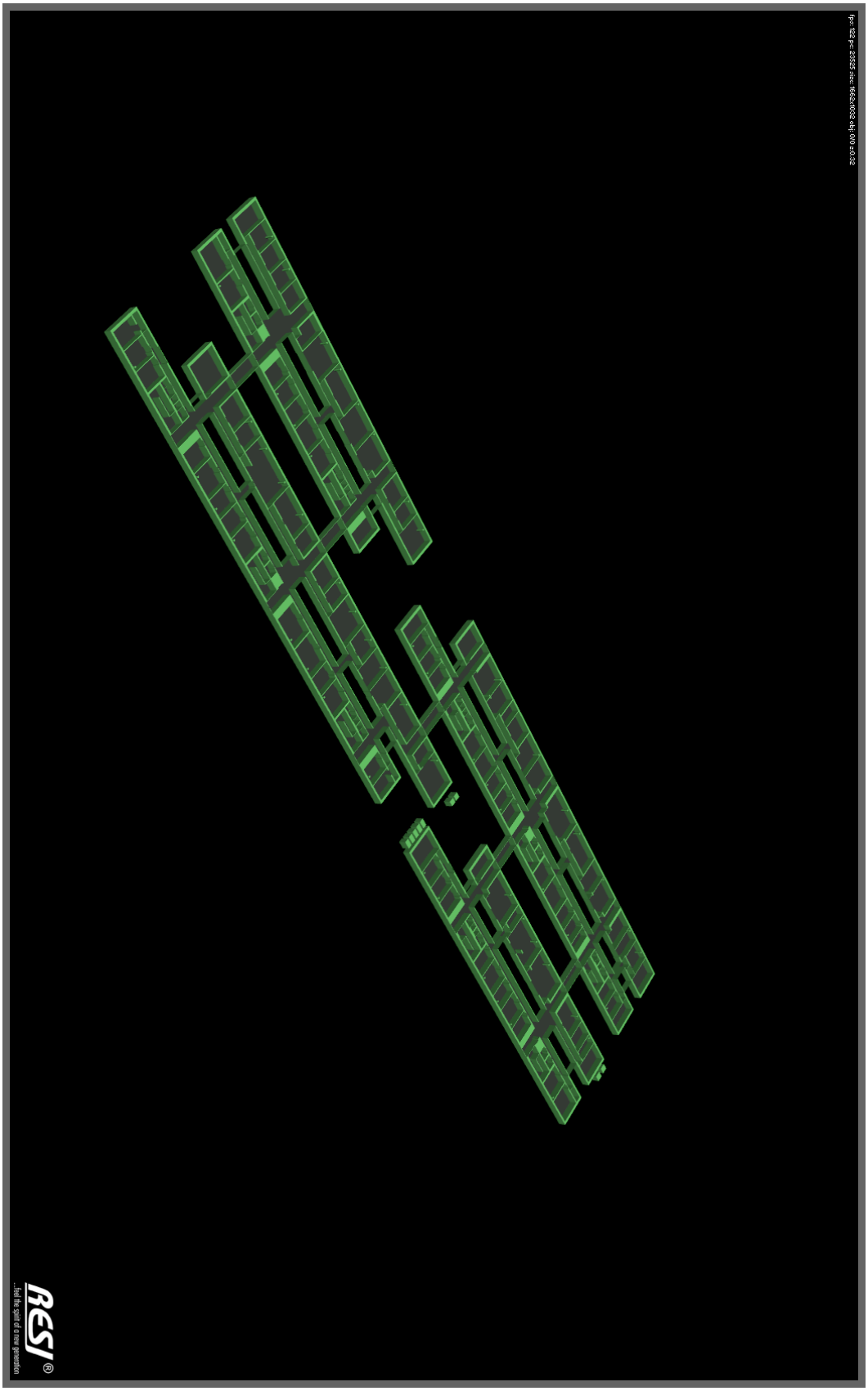


Abbildung 90: Gesamtansicht Inffeldgasse 16, 2. Stock

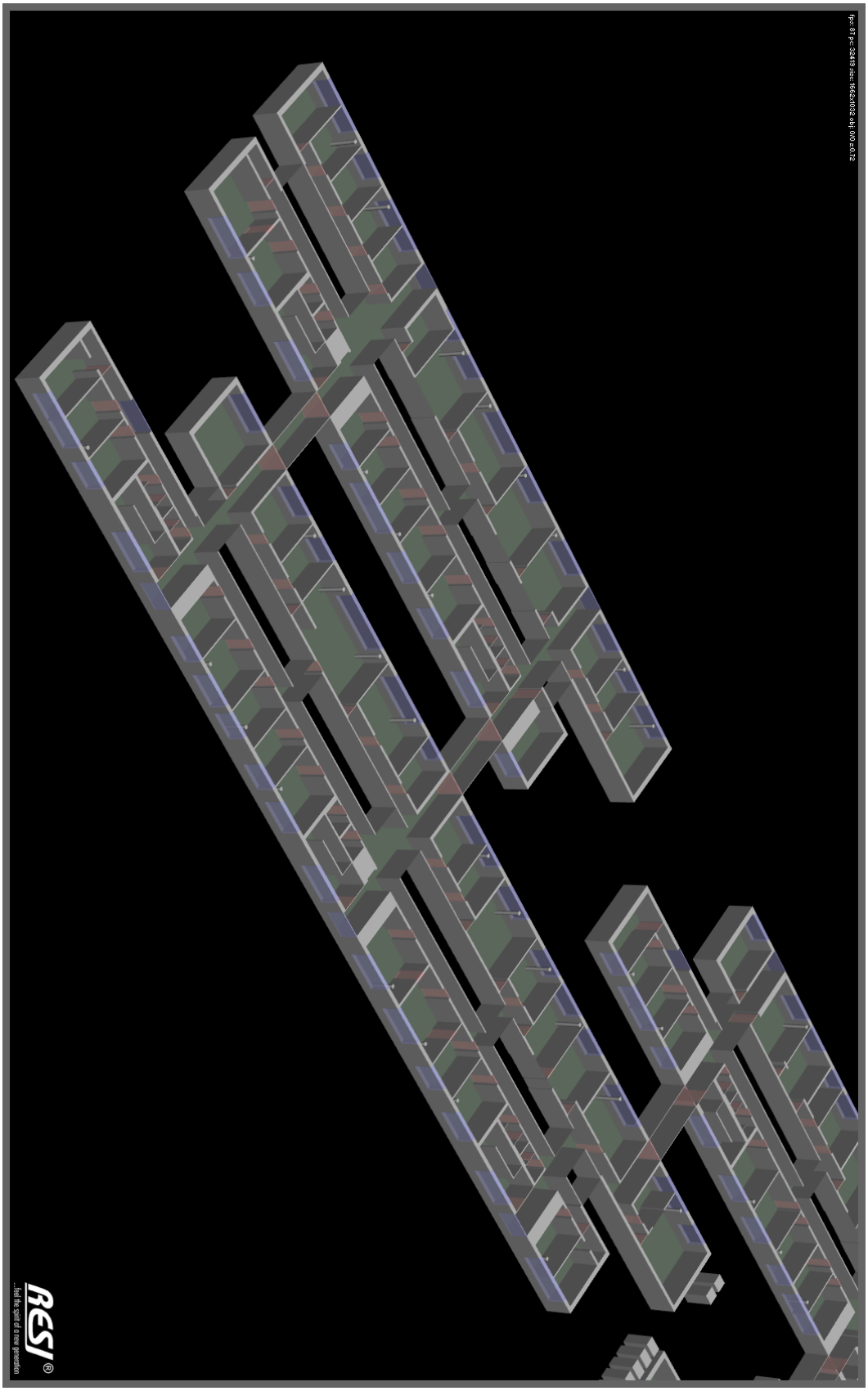


Abbildung 91: Detailansicht Inffeldgasse 16, südwestlicher Teil

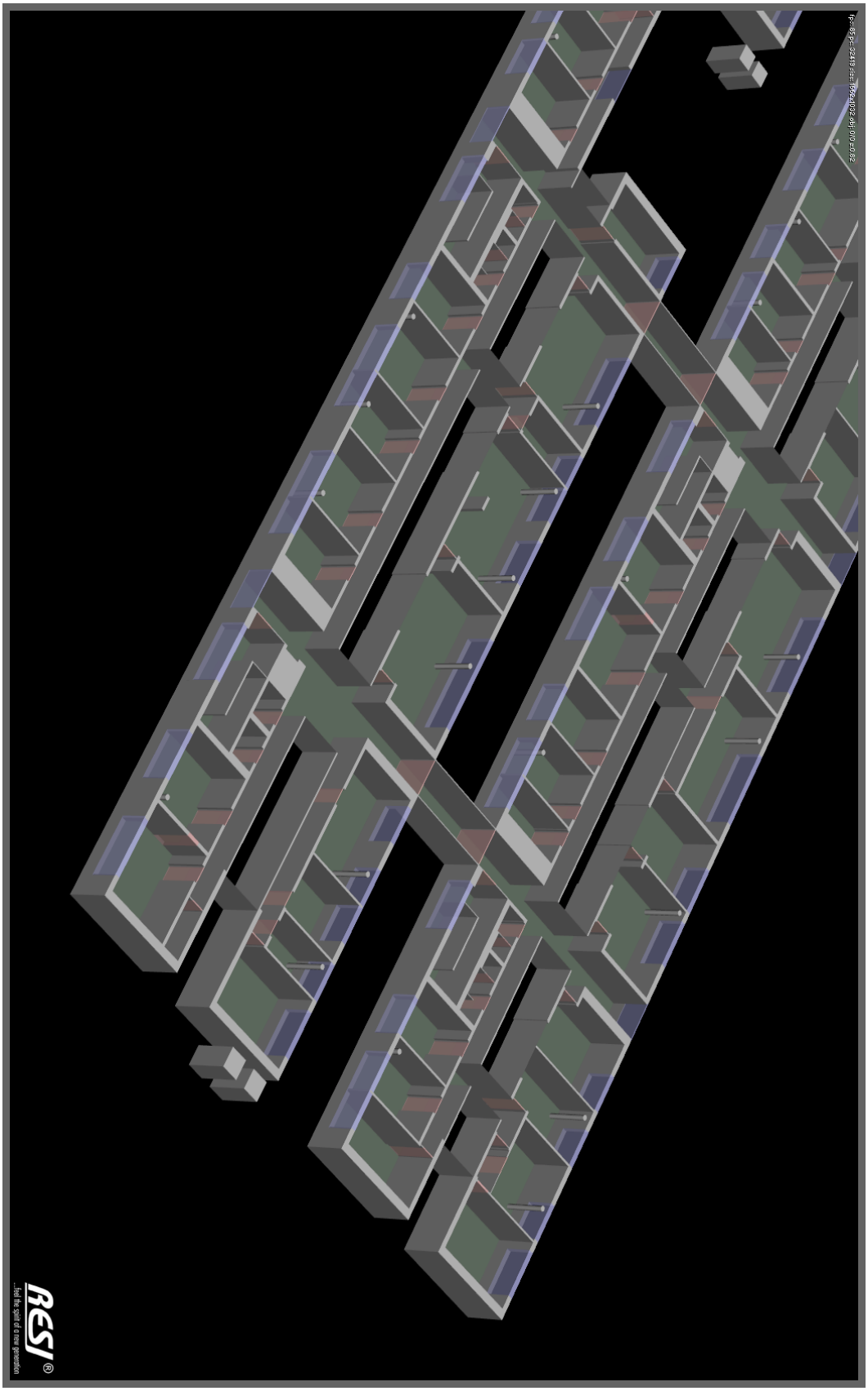


Abbildung 92: Detailansicht Inffeldgasse 16, nordöstlicher Teil

6 Zusammenfassung und Ausblick

Mit Hilfe der in der Arbeit vorgestellten Herangehensweisen, kann aus einer Reihe von CAD-Plänen ein semantisches Modell eines Gebäudes generiert werden.

Die Daten aus den Plänen werden hierzu zunächst bereinigt und mit semantischer Information in Form von Spezialkanten angereichert. Spezialkanten entstehen durch eine Transformation der im Plan eingezeichneten Symbole für Türen und Fenster. Im nächsten Schritt werden die von den geometrischen Elementen des Plans aufgespannten Flächen gebildet. Anhand der zuvor erzeugten Spezialkanten werden diese Flächen dann als Raum, Tür, Fenster oder Wand klassifiziert.

Die klassifizierten Objekte werden anschließend, hierarchisch gegliedert, in einer relationalen Datenbank abgelegt. Es kommt ein Datenbanksystem zum Einsatz, welches das Speichern und Verarbeiten von geometrischen Merkmalen erlaubt. Es ist also möglich Abfragen über die geometrischen Merkmale der repräsentierten Objekte durchzuführen. So kann beispielsweise sehr einfach die Gesamtfläche aller repräsentierten Räume berechnet werden.

Als Beispielanwendung für das Gebäudemodell wurde eine Applikation zur dreidimensionalen Visualisierung des Gebäudes umgesetzt. Sämtliche Daten werden dabei direkt aus der Datenbank bezogen. Für alle repräsentierten Objekte werden 3D-Modelle generiert, die für die Darstellung des Gebäudes verwendet werden. Die Visualisierung unterstützt den stufenlosen Wechsel zwischen einer schematischen Ansicht des gesamten Gebäudes und einer detaillierten Ansicht einzelner Stockwerke.

6.1 Offene Probleme

Eines der schwierigeren Probleme bei der Erfassung des Gebäudemodells ist die Transformation der Plansymbole in Spezialkanten. Die vorgestellte Operation verfolgt einen sehr generellen Ansatz und kommt daher mit vielen verschiedenen Symboltypen zurecht. Manche Symbole können mit dieser Herangehensweise nicht korrekt transformiert werden. In solchen Fällen muss der Benutzer das Symbol im Plan händisch durch eine einfache Kante ersetzen. Ausgehend von dieser Kante kann eine Tür, oder ein Fenster vom System wieder korrekt erkannt werden.

Ein weiteres offenes Problem ist, dass Räume oft nicht lückenlos von Wänden eingeschlossen sind. Je nach Beschaffenheit des Plans kommt dies beispielsweise vor, wenn Dehnfugen zwischen Wänden eingezeichnet sind. Die Operation zum Zwingen von Knoten auf nahe liegende Kanten kann diese Lücken nicht schließen, weil die Abstände meist zu groß sind. Für solche Fälle müsste eine neue Operation geschaffen werden, die Lücken mit speziellen Trennkanten schließt. Die Räume wären dann im Modell korrekt repräsentiert, wobei die Geometrie der Wände unverändert bleibt.

Manchmal werden Flächen, aufgrund der Gebäudestruktur, fälschlicherweise als Räume klassifiziert. Dies ist beispielsweise bei Innenhöfen von ringförmigen Gebäuden der Fall.

Die Klassifikation einer Fläche als Raum müsste in solchen Situationen verhindert werden können. Vorstellbar wäre es, spezielle Blockreferenzen in die Pläne einzuzeichnen, die darunter liegende Flächen aus dem Modell entfernen. Gleichermaßen sollte es auch möglich sein, mit Hilfe solcher Blöcke Räume zu erzwingen, die nicht durch Türen oder Fenster erreichbar sind. Auf diese Art könnten Leitungsschächte im Modell korrekt repräsentiert werden.

Vertikale Portale wie Stiegen und Aufzüge sind im Modell bisher noch nicht repräsentiert. Grundflächen von Stiegenhäusern und Aufzugsschächten müssten zunächst als spezielle Flächen in das Modell aufgenommen werden. Würde man dann zwei übereinander liegende Stockwerke betrachten, könnten die vertikalen Portale über den Schnitt dieser Spezialflächen gefunden werden. Für diese Aufgabenstellung würden sich die geometrischen Funktionen des verwendeten Datenbanksystems anbieten. Übergänge könnten so über eine Reihe von Datenbankabfragen gefunden und gespeichert werden.

CAD-Pläne enthalten eine Vielzahl von Informationen, die bisher noch nicht berücksichtigt wurden. Neben Bezeichnungen für Räume, Türen und Fenster sind dies in erster Linie durch Blockreferenzen repräsentierte Objekte. Je nach Anwendung könnten so beispielsweise Beleuchtungskörper, Bewegungsmelder, Rauchmelder oder Überwachungskameras in das Gebäudemodell aufgenommen werden.

6.2 Ausblick

In weiterer Folge sollte zunächst die Erfassung des Modells für den Benutzer komfortabler gestaltet werden. Die notwendigen Werkzeuge (Plandatenexport, Bereinigung der Plandaten, Analyse der Flächen, Datenbankimport) sollten zu diesem Zweck in einer Erweiterung für das verwendete CAAD System kombiniert werden. Der Grad der Automatisierung sollte erhöht werden, sodass nach entsprechender Konfiguration, auf Knopfdruck ein Modell für das gesamte Gebäude generiert wird.

Die möglichen Anwendungen für das Gebäudemodell sind vielfältig. Die auf dem Modell basierende Visualisierung würde sich hervorragend für Leitstände zur Sicherheits- bzw. Gebäudeleittechnik eignen. Auch Informationssysteme für große Gebäude wie Flughäfen und Einkaufszentren könnten aufbauend auf der vorgestellten Visualisierung umgesetzt werden.

Das Gebäudemodell würde sich auch für die Umsetzung von Steuerungsaufgaben in Gebäuden eignen. In der Sicherheitstechnik gibt es viele Anwendungen für die ein topologisches Modell des Gebäudes sehr hilfreich ist. Wird in einem Bereich Einbruchsalarm ausgelöst, könnten automatisch alle möglichen Fluchtwege beleuchtet werden, damit Überwachungskameras entsprechende Aufnahmen machen können. Das Problem der Wegfindung in Gebäuden kann mit den Nachbarschaftsinformationen im Gebäudemodell gelöst werden. Denkbar wäre der Einsatz als Navigationshilfe in großen Gebäuden.

Literaturverzeichnis

[Ben75] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, 1975.

[Bes03] Sergei Bespamyatnikh. Computing Closest Points for Segments. *Int. J. Comput. Geom. Appl*, 13, 2003.

[BiM] BiMserver. Building Information Modelserver (abgerufen am 27.04.2010): http://www.bimserver.org/.

[Bjö95] Bo-Christer Björk. *Requirements and information structures for building product data models*. PhD thesis, Helsinki University of Technology, 1995.

[BL05] Geiger A. Benner, J. and K. Leinemann. Flexible Generation of Semantic 3D Building Models. In *Next Generation 3D City Models*, Bonn, Germany, June 2005.

[BLK97] Bo-Christer Björk, Kurt Löwnertz, and Arto Kiviniemi. ISO DIS 13567 - The Proposed International Standard for Structuring Layers in Computer Aided Building Design. *ITcon*, 2:32–55, 1997.

[BMN97] Ulrike Bartuschka, Kurt Mehlhorn, and Stefan Näher. A Robust and Efficient Implementation of a Sweep Line Algorithm for the Straight Line Segment Intersection Problem. pages 124–135, 1997.

[Cod70] E. F. Codd. A relational model of data for large shared data banks. *Commun. ACM*, 13(6):377–387, 1970.

[Coi] Coin3D. Graphics Development Tools (abgerufen am 27.04.2010): http://www.coin3d.org/.

[DM99] P. Dosch and G. Masini. Reconstruction of the 3D Structure of a Building from the 2D Drawings of its Floors. page 487, 1999.

[FBF77] Jerome H. Friedman, Jon Louis Bentley, and Raphael Ari Finkel. An Algorithm for Finding Best Matches in Logarithmic Expected Time. *ACM Trans. Math. Softw.*, 3(3):209–226, 1977.

[FHH+00] Eyal Flato, Dan Halperin, Iddo Hanniel, Oren Nechushtan, and Eti Ezra. The Design and Implementation of Planar Maps in CGAL. page 2000, 2000.

[gbX] gbXML. Green Building XML (abgerufen am 27.04.2010): http://www.gbxml.org/.

[GHJV95] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Entwurfsmuster*. Addison-Wesley, 1995.

[HB05] Ian Howell and Bob Batcheler. Building Information Modeling Two Years Later – Huge Potential, Some Success and Several Limitations (abgerufen am 27.04.2010): http://www.laiserin.com/features/bim/newforma_bim.pdf, 2005.

[HB07] Rob Howard and Bo-Christer Björk. Use of standards for CAD layers in building. *Automation in Construction*, 16(3):290 – 297, 2007.

[HD07] Benjamin Hagedorn and Jürgen Döllner. High-level web service for 3D building information visualization and analysis. pages 1–8, 2007.

[HD08] Benjamin Hagedorn and Jürgen Döllner. Sketch-Based Navigation in 3D Virtual Environments. 0 2008. to appear.

[HH01] Iddo Hanniel and Dan Halperin. Two-Dimensional Arrangements in CGAL and Adaptive Point Location for Parametric Curves. pages 171–182, 2001.

[HLZY08] H.C. Huang, S.M. Lo, G.S. Zhi, and R.K.K. Yuen. Graph theory-based approach for automatic recognition of CAD data. *Engineering Applications of Artificial Intelligence*, 21(7):1073 – 1079, 2008.

[Irr] Irrlicht. A free open source 3D engine (abgerufen am 27.04.2010): http://irrlicht.sourceforge.net/.

[KFH+03] Calvin Kam, Martin Fischer, Reijo Hänninen, Auli Karjalainen, and Jarmo Laitinen. The product model and Fourth Dimension project. *ITcon*, 8:137–166, 2003.

[Khe04] Lachmi Khemlani. The IFC Building Model: A Look Under the Hood (abgerufen am 27.04.2010): http://www.aecbytes.com/feature/2004/IFCmodel_pr.html, March 2004.

[KMP+08] Lutz Kettner, Kurt Mehlhorn, Sylvain Pion, Stefan Schirra, and Chee Yap. Classroom examples of robustness problems in geometric computations. *Computational Geometry*, 40(1):61 – 78, 2008.

[Lew96] Rick Lewis. Generating Three-Dimensional Building Models from Two-Dimensional Architectural Plans. Master's thesis, University of California, Berkeley, 1996.

[LS98] R. Lewis and C. Séquin. Generation of 3D Building Models from 2D Architectural Plans. *Computer-Aided Design*, 30:765–779, 1998.

[MN99] Kurt Mehlhorn and Stefan Näher. LEDA: A Platform for Combinatorial and Geometric Computing. 1999.

[Moe06] Herbert Moelle. *Rechnergestützte Planungsprozesse der Entwurfsphasen des Architekten auf Basis semantischer Modelle*. PhD thesis, Technische Universität München, 2006.

[Oeb97] Alfons Oebbeke. IAI / IFC: Der Turmbau zu Babel am Ende des 20. Jahrhunderts (abgerufen am 27.04.2010): http://www.aecweb.de/m/aec-x/f-iai1.htm, 1997.

[OGR] OGRE. Open Source 3D Graphics Engine (abgerufen am 27.04.2010): http://www.ogre3d.org/.

[Opea] OpenSceneGraph. Open source high performance 3D graphics toolkit (abgerufen am 27.04.2010): http://www.openscenegraph.org/projects/osg.

[Opeb] OpenSG. Portable scenegraph system (abgerufen am 29. 04. 2010): http://www.opensg.org/.

[OZMF05] Peter van Oosterom, Siyka Zlatanova, and Elfriede M. Fendel. CityGML: Interoperable Access to 3D City Models. In *Geo-information for Disaster Management*, pages 883–899, 2005.

[PAAV03] Octavian Procopiuc, Pankaj K. Agarwal, Lars Arge, and Jeffrey Scott Vittery. Bkd-tree: A dynamic scalable kd-tree. In *In Proc. International Symposium on Spatial and Temporal Databases*, pages 46–65, 2003.

[PPV95] Alberto Paoluzzi, Valerio Pascucci, and Michele Vicentino. Geometric programming: a programming approach to geometric design. *ACM Trans. Graph.*, 14(3):266–306, 1995.

[PS07] A. Paoluzzi and G. Scorzelli. Pattern-Driven Mapping from Architectural Plans to Solid Models of Buildings. 2007. Israel-Italy Bi-National Conference.

[Rom05] Richard Romberg. *Gebäudemodell-basierte Strukturanalyse im Bauwesen*. PhD thesis, Technische Universität München, 2005.

[SAT08] Ferial Shayeganfar, Amin Anjomshoaa, and A Min Tjoa. A Smart Indoor Navigation Solution Based on Building Information Model and Google Android. pages 1050–1056, 2008.

[SFJS05] G. Schall, F. Fraundorfer, Newman J., and D. Schmalstieg. Construction and Maintenance of Augmented Reality Environments Using a Mixture of Autonomous and Manual Surveying Techniques. In *7th Conference on Optical 3-D Measurement Techniques*, 2005.

[Sig03] Signpost. Signpost 2003 Handheld Augmented Reality (abgerufen am 27.04.2010): http://studierstube.icg.tu-graz.ac.at/handheld_ar/signpost2003.php, 2003.

[SSW+07] Dieter Schmalstieg, Gerhard Schall, Daniel Wagner, Istvan Barakonyi, Gerhard Reitmayr, Joseph Newman, and Florian Ledermann. Managing Complex Augmented Reality Models. *IEEE Computer Graphics and Applications*, 27:48–57, 2007.

[vT04] Christoph Alban van Treeck. *Gebäudemodell-basierte Simulation von Raumluftströmungen*. PhD thesis, Technische Universität München, 2004.

[WBT07] Emily Whiting, Jonathan Battat, and Seth Teller. Topology of Urban Environments. In *Computer-Aided Architectural Design Futures 2007*, 2007.

[WFZH07] Ron Wein, Efi Fogel, Baruch Zukerman, and Dan Halperin. Advanced programming techniques applied to CGAL's arrangement package. *Comput. Geom. Theory Appl.*, 38(1-2):37–63, 2007.

[Wika] Wikipedia1. Wikipedia Artikel über die konvexe Hülle (abgerufen am 27.04.2010): http://en.wikipedia.org/wiki/ArchiCAD.

[Wikb] Wikipedia2. Wikipedia Artikel über die Delaunay Triangulierung (abgerufen am 27.04.2010): http://en.wikipedia.org/wiki/Delaunay_triangulation.

[Wikc] Wikipedia3. Wikipedia Artikel über den k-d Baum (abgerufen am 27.04.2010): http://de.wikipedia.org/wiki/K-d-Baum.

[YWR09] Xuetao Yin, P. Wonka, and A. Razdan. Generating 3D Building Models from Architectural Drawings: A Survey. *Computer Graphics and Applications, IEEE*, 29(1):20–30, Jan.-Feb. 2009.

[ZLF03] G. S. Zhi, S. M. Lo, and Z. Fang. A graph-based algorithm for extracting units and loops from architectural floor plans for a building evacuation model. *Computer-Aided Design*, 35(1):1 – 14, 2003.