



Ing. Roman Zeleznik BSc

**Efficient Implementation of Virtual
Physics Experiments for Web Based
Interactive Courses at the MIT**

MASTER'S THESIS

to achieve the university degree of

Master of Science

Master's degree programme: Telematics

submitted to

Graz University of Technology

Supervisor

Univ.-Doz. Dipl.-Ing. Dr.techn. Christian Guetl

Institute for Information Systems and Computer Media - IICM

Graz, October 2015



Ing. Roman Zeleznik BSc

**Effiziente Implementierung von
Virtuellen Physik-Experimenten für
Interaktive Online Kurse am MIT**

MASTERARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

Masterstudium Telematik

eingereicht an der

Technischen Universität Graz

Betreuer

Univ.-Doz. Dipl.-Ing. Dr.techn. Christian Guetl

Institut für Informationssysteme und Computer Medien - IICM

AFFIDAVIT

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present diploma thesis.

Graz, _____

Date

Signature

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Diplomarbeit identisch.

Graz, am _____

Datum

Unterschrift

Abstract

Teaching freshman MIT students physics is a challenging task as every MIT student has to pass these courses no matter what topics they are studying. Hence, most of the students are not very interested. Additionally, the rest of their study is very time consuming. These are the main reasons why students do not like these courses. Trying to motivate them to learn physics, the courses were ported to so called e-learning courses. Evaluations in the past have shown that this new form of education can increase motivation as well as simplify learning. Especially interactive simulations help students understand the taught topics. Therefore a new e-learning platform called edX was created, an online platform for many courses. Two of them are the basic physics courses 8.01x and 8.02x. These courses contained seven simulations which gave the students the possibility of working whenever they want and as often as they want. Unfortunately an evaluation of a past physics course has shown that students did not, or only in a very limited manner, make use of the simulations. Therefore the wish was raised to make the simulations interact with edX to give the course creator the possibility of asking students questions that can only be answered by using the simulations. These questions can then be used for homework as well as exams. The big benefit of this system is, that students get instant feedback to their work, if there were errors and where these errors occurred.

In this master thesis an interface between the seven existing simulations and the edX platform is created. The main problem here was, that the existing evaluations were written in Java and compiled to JavaScript. At the time the master thesis was written, edX support of stand alone JavaScript applications was still under construction and furthermore had problems with the Java written simulations. To overcome these problems, a HTML host page for the simulations was used as a buffer between edX and the simulations. With this solution, a stable and safe interface between edX and the simulations was created.

Furthermore, three of the existing TEAL simulations were extended which can now be used for answering course questions. These simulations were the *Point Charges Extended Simulation*, *Gausses Law Extended Simulation* and *Amperes Law Extended Simulation*. The main idea for these new simulations was that students have to find hidden elements with a moving element. This would give the simulations a game character and should increase students motivation.

For evaluating the work of this thesis the Point Charges Extended Simulation was used. Five test sets were created which can be used by participants for training and learning. After that an exam set had to be solved. The analysis of the evaluation shows, that the participants could learn the physics behind the simulation very well and were really motivated during the test and exam sets. The majority of the participants would enjoy to use this system at their home university.

Kurzfassung

Studenten am Massachusetts Institute of Technology (MIT) müssen in den ersten Semestern ihres Studiums zwei Physik Grundlagenkurse bestehen. Diese Studenten für diese Kurse zu interessieren und zum Lernen zu motivieren ist eine große Herausforderung. Da ein Studium an dieser Universität sehr zeitaufwendig ist sind viele Studenten nicht besonders interessiert an solchen Grundlagenfächern. Forschungen haben gezeigt, dass online Kurse, speziell solche die interaktive Simulationen beinhalten, sich positiv auf das Lernverhalten und Verständnis von Studenten auswirken können. Dies führte zu der Entwicklung von Onlinekursen sowie der Online Lernplattform edX. Diese Plattform bietet eine große Anzahl verschiedener Kurse an. Zwei dieser Kurse sind die Physik Grundlagenkurse 8.01x und 8.02x welche in TEAL Klassenzimmern abgehalten werden. Um Studenten einen einfacheren und flexibleren Zugang zu Physikexperimenten zu bieten wurden sieben Simulationen für die Onlineplattform edX portiert, wofür JavaScript verwendet wurde. Diese neuen Simulationen gaben den Studenten die Möglichkeit wann und so oft sie wollten mit ihnen zu lernen. Ziel war es mithilfe dieser Simulationen die Studenten für den Inhalt der Kurse zu interessieren und zum Lernen zu motivieren. Eine Evaluierung eines vorangegangenen Physikkurses hat jedoch gezeigt, dass Studenten die online Simulation nur sehr selten oder auch gar nicht verwendeten. Dies führte zu der Überlegung die Simulationen interaktiv zu gestalten um zukünftig Fragen stellen zu können welche von den Studenten mithilfe der Simulationen beantwortet werden müssen.

In dieser Diplomarbeit wurde ein Interface zwischen den bestehenden sieben Simulationen und der edX Plattform erstellt. Ein großes Problem dabei war, dass die bestehenden Simulationen in Java geschrieben waren und dann nach JavaScript kompiliert wurden. Zu der Zeit, als diese Arbeit geschrieben wurde befand sich die Unterstützung von JavaScript Anwendungen in edX noch in einer Beta-Phase und hatte große Probleme mit den Java Simulationen. Aus diesem Grund wurde die HTML-Host Datei der JavaScript Simulation als Puffer verwendet um so eine stabile und sichere Kommunikation zwischen edX und den Simulationen gewährleisten zu können.

Weiters werden 3 der bestehenden TEAL Simulationen erweitert um sie zukünftig zur Beantwortung von Fragen in Onlinekursen verwenden zu können. Diese Simulation waren die *Point Charges Extended Simulation*, die *Gausses Law Extended Simulation* und die *Amperes Law Extended Simulation*. Die Idee hinter den neuen Simulationen war, dass Studenten versteckte Elemente finden müssen. Das verleiht den Simulationen einen Spiel-Charakter und soll die Studenten weiter motivieren mit ihnen zu arbeiten.

Zu Evaluierung am Ende dieser Arbeit wurde die Point Charges Extended Simulation verwendet. Dafür wurde 5 Trainings-Sequenzen zum Üben erstellt gefolgt von einer Prüfung-Sequenz. Die Auswertung der Prüfung ergab, dass die Probanden waren nach dem trainieren mit den Übungs-Sequenzen sehr gut auf die Prüfung vorbereitet und konnten viele Punkte erreichen. Die Auswertung ergab auch, dass die Probanden sehr motiviert waren und die meisten würden sich dieses System für Kurse an ihrer Universität wünschen.

Acknowledgement

I want to thank so many people for making this master thesis as incredible as it is. I have to thank all of you for your support and guidance through my work. During the time I worked on this master thesis I had the opportunity to work with some of the best scientists at one of the best universities.

First I want to thank every one from the Graz University of Technology. Especially Christian Güttl for making all this possible and for the continuous help and support through all the six month in Boston and the time back in Graz. Thanks also to Johanna for all the help getting the visa and also the technical help through my work.

Thank you very much!

And I want to thank all the people from Boston, especially the CECI team. My thanks go to John Belcher for all the help from the west side of the Atlantic, for showing me a whole new form of education and a new direction in which I want to steer my future work and for introducing me to so many wonderful people. Thanks to Phil for all the support and help with all the technical issues. Thanks to Kirky and Maria for the administrative support. At this point a special thanks to Kirky for all the help before my trip, answering thousands of mails and for the help getting my visa! Thanks to Luciano for having lunch with me so many times. And thanks to Kimberle for bringing Kyla into the office.

Furthermore I want to thank Saif for the work on the edX course with me. And of course thanks to Ivan for all your help and the bike. Thanks also to Mark for all the wonderful stories about the MIT, Boston and the US and for the many recommendations for our weekends and vacation.

Thanks all of you!

I want to thank the Marshall Plan foundation for the financial support which made this journey possible!

Finally I want to thank my love and my life Oana. Words can not describe how much I love you!

Contents

1. Introduction	1
1.1. Goals and Objectives	3
1.2. Structure of the work	3
2. Background and related work	5
2.1. E-learning	5
2.2. Massive Open Online Courses (MOOCs)	7
2.3. Simulations in education	10
2.3.1. Simulations vs games	11
2.4. Games in education	12
2.4.1. Gamification	12
2.4.2. Games vs gamification	14
2.5. TEAL and TEALsim	14
2.6. EdX - An online learning platform	16
2.7. Evaluation of a prior 8.02x course	17
2.8. The <i>STAR</i> framework	18
2.9. Summary	19
3. Technologies	21
3.1. EdX and the mix of HTML, Python, JavaScript and LaTeX	22
3.2. Main structure of an edX online course	22
3.2.1. A basic setup of an edX course	23
3.2.2. The setup of the courses 8.02x	25
3.2.3. The segmentation of the edX course 8.02x in separate files	26
3.3. GWT - Extending edX with a new programming language	28
3.3.1. The GWT compiler	28
3.3.2. Advantages of GWT	29
3.4. The jsinput sample problem	30
3.5. Summary	31
4. Problem definition	33
4.1. The goal of this master thesis	34
4.1.1. Adapting the simulations for answering questions	35

Contents

4.1.2.	Getting an initial state from edX to the simulation . . .	36
4.1.3.	Returning an answer to edX	36
4.1.4.	Reloading a previous state	36
4.2.	Summary	37
5.	Design and development of the Interface between TEAL Simulations and edX	39
5.1.	The first extended simulation	40
5.2.	Grading a simulation	42
5.2.1.	EdX - <i>customresponse</i>	43
5.2.2.	EdX - <i>jsinput</i>	43
5.2.3.	Grading using python	46
5.3.	Starting a simulation with initial values	47
5.3.1.	Creating the initial values with python	49
5.3.2.	Iframes and postmessages	49
5.4.	Development of a test course using jsinput	52
5.5.	Loading, saving and sending values	52
5.5.1.	One variable in one file used by two programming languages	54
5.5.2.	One variable in two files used by the same programming language	54
5.5.3.	One variable in two files used by two programming languages	55
5.5.4.	Reloading a previous state - The way from XML to JavaScript to Java	56
5.6.	The synchronization problem	61
5.7.	Summary	62
6.	The extended TEAL simulations	65
6.1.	The first extended TEAL simulation: Point Charges Extended	66
6.1.1.	Requirements and Design	66
6.1.2.	The extended simulation	67
6.1.3.	Different states of the running simulation	69
6.1.4.	The prototype of Point Charges Extended	71
6.2.	The second TEAL simulation: Gausses Law Extended	72
6.2.1.	Requirements and Design	72
6.2.2.	The extended simulation	74
6.2.3.	Different states of the running simulation	76

6.3. The third extended TEAL simulation:	
Amperes Law Extended	78
6.3.1. Requirements and Design	78
6.3.2. The extended simulation	80
6.3.3. Different states of the running simulation	81
6.4. Findings	83
6.4.1. Placing hidden elements	83
6.4.2. The accidental gamification	84
6.5. Summary	86
7. Evaluation	87
7.1. Procedure	87
7.1.1. Pre- and Post-Questionnaires	88
7.1.2. Test setup	89
7.1.3. The five training sets	89
7.1.4. The exam set	90
7.2. Participants	90
7.3. Results	92
7.3.1. Pre-Questionnaire	92
7.3.2. Experiment	92
7.3.3. Post-Questionnaire	97
7.4. Discussion	99
8. Future Work	101
8.1. Near future - The rest of the seven TEAL simulations	101
8.2. Ultimate goal - A complete new framework for simulations	108
8.3. Improvements on jsinput	108
9. Summary	111
List of Figures	115
Bibliography	121
Appendix	125
Appendix	125
A. Code parts	127
A.1. Limit the movement	127
A.2. Place elements on a grid	129

Contents

A.3. Place and remove dummies	131
B. Used Software and Hardware	135
C. Setup of a local edX test server	137
C.1. Setup of the edX test server - Version 2013	137
C.2. Setup of the edX test server - Version 2014	138
D. Questionnaires	141
D.1. Pre-Questionnaire	141
D.2. Post-Questionnaire	144

1. Introduction

At the Massachusetts Institute of Technology (MIT) every undergraduate student has to take two semesters of basic physics courses, no matter which field of studying the students are going to take. These courses are tough and require a lot of work, and hence are not very favored by the students. Since 2005 the physics courses are held in so called *Technology-Enabled Active Learning* (TEAL) classes. TEAL is a method for giving the students, among other things, the possibility to work on their own with different physics experiments. This changed the traditional and passive way of teaching to an interactive way where students can test their understanding. There were different reasons for the TEAL project (Dori & Belcher, 2005). One was to decrease the number of students failing the courses. Another was to make studying easier and hopefully to make students remember the learned material for the rest of their life. It was also made to motivate the students to learn physics. During one of the first meetings, Prof. Belcher explained that he wanted the students to not hate the courses any more. And if they just dislike it instead of hating it, he would be happy.

During the last decades technological improvements on the level of personal computers have changed the way students learn. Taking advantage of these developments the *TEAL simulation framework* (TEALsim) was created (Belcher et al., 2006). It is a stand alone software including several virtual TEAL simulations which were used in TEAL classes but could be used by every student at home also.

The last evolutionary step was to enhance the physics courses by the e-learning platform edX for teaching the physics courses. These courses included the whole course material as well as seven ported TEAL simulations. Furthermore, online questionnaires with instant feedback and online homeworks were part of these edX courses. The mid-semester survey of 8.02x has shown that students really liked the edX course (Rayyan & Chu, 2014). Most important for them was the immediately feedback they received for online questions. Past works have also shown that interactive simulations can improve students learning (Clayton & Theodora-Ismene, 2005). That

1. Introduction

was also the reason why the TEAL simulations were ported to edX. The mid-semester survey (Rayyan & Chu, 2014) has also shown that 47% of the students did not use them and only 4% thought they were extremely helpful. Prof. Belcher wanted to extend the existing seven TEAL simulations with the possibility of asking students questions during the online course which can only be answered by using those simulations. This would on the one hand make the students using them, on the other hand it would give them the fast feedback they liked most from the edX courses.

There still existed a small number of online courses¹ which did something similar to the ideas of this thesis, for example the courses 7.QBWx, 8.EFTx and others. They also contained interactive experiments which were used to answer questions and solve tasks. In contrast to these courses, where one person created the edX part as well as the JavaScript part, the physics courses where this work is focused on were created by two different people. One was responsible for the course content, which was written in XML and python, the other for the simulations, which were written in Java and JavaScript. Hence the goal of this work was not only to extend the existing simulations, but also to create an interface between them and the course content to keep the strict separation.

Another objective of this work was the development of a concept of how to change the existing simulations for giving the lecturers the possibility of asking useful questions which increase students motivation for using the simulations. Once all requirements were defined the actual work on the implementation began. Therefore *jsinput* was used², an edX function made for interacting with JavaScript implementations like the seven ported TEAL simulations. An aspect that had to be dealt with was that the existing simulations were written in Java and compiled with the *Google Web Toolkit* (GWT) to JavaScript Code. This made a direct use of *jsinput* with the TEAL simulations impossible. Once the programming work and implementation was done the new extended simulations were presented to the physics education staff which consisted of brilliant, well experienced and very smart MIT lecturers. At this point it became obvious that this project was not just to create a working interactive simulation with a stable interface. Each decision was questioned, every tiny detail had to be correct and everything had to be perfect to fulfill the high standards of the MIT.

¹I. Ceraj created these courses and therefore a comprehensive framework. More information to this framework can be found in chapter 2.8.

²J.M Claus from edX recommended the using of the edX function *jsinput*.

1.1. Goals and Objectives

In order to solve the problems described above two main objectives were defined.

Objective One: Extend one of the seven existing TEAL simulations which were ported to edX for enabling the course creator the possibility of asking students questions which have to be answered by using them. This includes on the one side to setup a virtual edX server locally for creating and testing an prototype course and furthermore understanding and editing an existing course. On the other hand understanding the implementation of the existing simulations.

After that, a concept of how to extend these simulations has to be developed followed by formulating useful course questions for this simulations. This was an interesting process which included many different people from education as well as programmers. This work had to combine the wishes of lecturers with the possibilities the programmers had and take into account the available time period.

Objective Two: Create an interface between the extended TEAL simulations and the course platform which embeds them. This was a very important aspect as the main course was created by educational stuff but the simulations were created by programmers. That means the educational stuff had less knowledge about programming while the programmers were not familiar with education. While this circumstance was very reasonable for creating the best physics course for a great education, it also led to the requirement of an interface between the simulations and the online course which ensures none of the two creators has to work in the others field.

1.2. Structure of the work

This master thesis is divided into two main parts. The first section describes the background and basics of this work. It encompasses the chapters 2, 4 and 3. Part two is about the implementation and the results, containing chapters 5 and 6 as well as an evaluation in chapter 7 and an outlook in chapter 8.

Chapter 2 explains the concepts *e-learning*, *MOOCs* and *simulation* and their application in education. Furthermore it describes the *Technology-enabled active learning* teaching format and the conversion to TEALsim with the

1. Introduction

seven ported simulations, which should be extended in this work. After that an overview of edX, an online e-learning platform for which the physics courses were created, is given. That includes also an overview of the structure of such an online course and the different programming languages which were used. Finally an introduction to GWT, the *Google Web Toolkit*, is given and how an edX course can be extended using it.

Chapter 4 illustrates the problem definition. Introductory the relation of TEALsim and edX is outlined. After that the individual points which should be accomplished within this work are defined.

In chapter 3 the related work to this master thesis is summarized. It shows the innovation of this work and consequently why there was so less related work which was appropriate for this project. Afterwards two concepts which were meant to be suitable for this work are explained and why they turned out to be incompatible with the requirements of this project.

Chapter 5 refers to the whole implementation of this project. It shows the underlying technologies, which problems came up and how they were solved. For better understanding a test course is shown which was created for testing the new mechanisms. At the end a flow chart is presented which explains the whole structure of the extended simulations and its interaction with edX.

Chapter 6 shows the final results of this work, the three converted simulations. It describes how the existing simulations were extended, which changes had to be done, where the problems were and finally the implementation and underlying mechanism of the new simulations. As the three new simulations are all different an own sub-chapter was created for each of them which shows the ideas and possibilities of how they could be used in future courses.

In chapter 7 an evaluation of the new simulations is presented and shows the benefits of them as well as the drawbacks. The evaluation was done by various people in order to evaluate different aspects of the work. The participants include physics experts and computer scientists but also staff and students from the *Graz University of Technology* as well as the *University of Graz*.

In the last chapter 8 a short outlook is given on how the new simulations can be used in online courses. Furthermore this chapter describes how this work will be continued and where it may lead to and which modifications to edX and its functions would have to be done.

2. Background and related work

Electronic learning (e-learning) and *Massive Open Online Courses* (MOOCs) are both new learning methods, especially MOOCs (Hernández et al., 2007; Rizzardini et al., 2013). They became well known during the last years but have also changed enormously during that time. Used techniques are very different and hence results and experiences too. Therefore a general description of these two terms is given in this chapter as well as an overview of how they can be extended by simulations and games. Furthermore an evaluation of the physics course 8.02x is presented to describe the benefits and disadvantages of the simulations' application focusing on this course. This physics course is held in a so called TEAL class room but is also enhanced by online materials like homeworks, questioners and simulations using the MOOCs platform edX. Part of this work will be to create an interface between the online simulations of the physics course and the edX platform.

2.1. E-learning

This section summarizes the work of Zhang & Nunamaker (2003). With the increasing use of computers and the spread of network technologies and the Internet, teaching is continuously moving from the traditional classroom learning to electronic-learning (e-learning). They define e-learning as every form of learning where the course material is delivered electronically via the Internet. Therefore different systems can be used:

- E-mails
- Newsgroups
- Knowledge boards
- Online databases
- Websites

For this work the last listed concept, *Websites*, is most interesting. In the near past, especially since the rise of JavaScript, websites became more and

2. Background and related work

more the preferred way of e-learning. This progress was supported by the upcome of HTML5 which is supported by every modern browser as of today (W3schools website, 2014).

The benefits of e-learning are numerous (Zhang & Nunamaker, 2003): It is completely time and location flexible. Through e-learning people from all over the world are able to access the same courses, discuss with each other or receive support from experts from different universities. E-learning is a learner-centric teaching method. It gives the learners the possibility to structure and organize their own way and time of learning instead of receiving passively information in a classroom during fixed lessons. Moreover, students have unlimited access to learning materials so they can reuse it as much as they need to understand a specific topic or also skip parts they maybe already know. E-learning is more self-directed and self-paced. Furthermore, discussions or asking questions through online forums for example can encourage learners to ask questions they would not ask in a classroom because of fear or shame. Another advantage of e-learning are cost and time saving. Up to 40% of in-person corporate learning are spend for traveling. Zhang & Nunamaker (2003) also found out, that there is no significant difference in effectiveness between e-learning and traditional learning.

For transferring information from teachers to learners, Zhang & Nunamaker (2003) describe videos as the most powerful non text based method. Video on demand systems became more and more popular in the last years. This leads to a possible drawback of e-learning. Zhang & Nunamaker (2003) mentioned that text-based learning materials can decrease students motivation and engagement while learning. Furthermore they state the importance of sufficient and considerable learning material. Another drawback of e-learning is that it can be less interactivity as it *“separates students and instructors physically by time and location”* (Zhang & Nunamaker, 2003). And a very important and critical part of e-learning is security. This includes on the one hand the security of knowledge and course material and on the other hand the security of stored data of learners. This can be personal data, grades and course usage data and perhaps even payment informations of students.

E-learning is the topic of a wide area of learning methods. One branch of e-learning are Massive open online courses (MOOCs) (Rizzardini et al., 2013), as described in section 2.2. Furthermore, e-learning courses can be enhanced by simulations and games, as discussed in section 2.3 and 2.4. This work is part of the evolution of the basic physics course 8.02x at the

2.2. Massive Open Online Courses (MOOCs)

MIT in Boston. It was initially hold as a traditional course in a classroom, was then moved to the TEAL classroom 2.5 and finally extended by the MOOCs platform edX offering virtual physics experiments.

2.2. Massive Open Online Courses (MOOCs)

MOOCs) are a relatively new form of education. Rizzardini et al. (2013) describes a MOOC as *“open and free of charge, active involvement and participatory and the contributions are shared for the learning community, and the communication and collaboration tools and recourses are widely distributed”* (Rizzardini et al., 2013). An early example for a MOOC is the course *“Creating Webpages”* from the Galileo University of Guatemala which was offered in 2005 (Hernández et al., 2007).

This section summarizes the work of Rodriguez (2013) on MOOCs. Compared to traditional e-learning models and online education, MOOCs can reach a huge number of learners due to their openness. Examples for three well known MOOCs platforms are edX, udacity and Coursera. Today two different types of MOOCs are known, cMOOCs and xMOOCs, which differ mainly in their pedagogical style. Both types of MOOCs have in common that the main idea behind them is openness. cMOOCs are based on connectivism and networking, hence they are also called connectivist MOOCs. xMOOCs implement the behaviorist approach. They were initially offered by elite US institutions and their x in xMOOCs is adapted from MITx and edX.

cMOOCs *“are deeply immersed in the discourse of openness”* (Rodriguez, 2013). Students obtain access to an online environment where they can present their ideas and approaches for solutions as well as their challenges. This gives other students the possibility of helping with their own knowledge and reflections. The result is a new form of learning, moving away from traditional teacher or tutor orientated teaching in closed groups, to self directed learning within an interactive open online environment. In cMOOCs knowledge is created by three main manners. First, freely available readings and presentations of teachers or tutors. Second, materials of invited experts or other stuff. Third, the contributions shared by the learners.

Related to openness, xMOOCs is more restrictive then cMOOCs as their learning material is mostly shared under limiting copyright licenses. In contrast to the many-to-many relationship of cMOOCs, xMOOCs is a tutor

2. Background and related work

or teacher based learning method, hence xMOOCs fulfill a one-to-many relationship. Due to that, xMOOCs are also described as a learning management system.

Today most well known universities like MIT, Harvard, Stanford University and many more offer MOOCs (Brown, 2013).

Literature outlines the advantages of MOOCs as follows (Gütl et al., 2014; Rizzardini et al., 2013):

- Reach of a wide community of participants.
- High accessibility for participants to courses with no respect to their geographical origin or other personal background.
- Support for self regulated learning including a high number of different learning tools.

The disadvantages were summarized as follows:

- High drop out rates. Only about 10% complete MOOCs successfully.
- Students felt isolated and disconnected.
- Students were not prepared to self organization, independently learning and the use of learning tools.
- The lack of support in learning activities.
- Feedback only in terms of assessment.

Rizzardini et al. (2013) survey in their work two free and open MOOCs offered by the Galileo University in Guatemala provided to Spanish speaking students. Their evaluation of these courses describe two salient findings, the mean of final grades over the students who completed the courses and the drop out rate. The mean of final grades of the mentioned MOOCs were 88% and 81% respectively, again for students who finished the courses. This compares with high dropout rates of 91,5% and 96,56% respectively. The work of Gütl et al. (2014) on these courses analysis a questionnaire on students who did not complete the MOOCs. Results are shown in figure 2.1. Detailed information on why students left the MOOCs can be found in figure 2.2. Rizzardini et al. (2013) also describe a problem they had to deal with while offering a MOOC: The requirements for hardware resources to provide a stable learning system especially during access peaks.

2.2. Massive Open Online Courses (MOOCs)

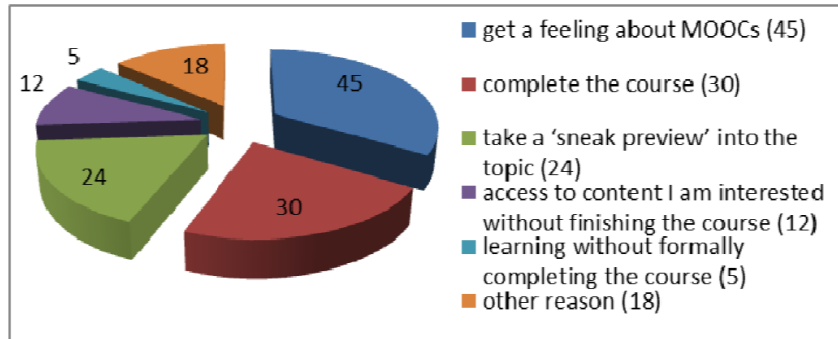


Figure 2.1.: Reasons why users enroll and start the MOOC (Gütl et al., 2014).

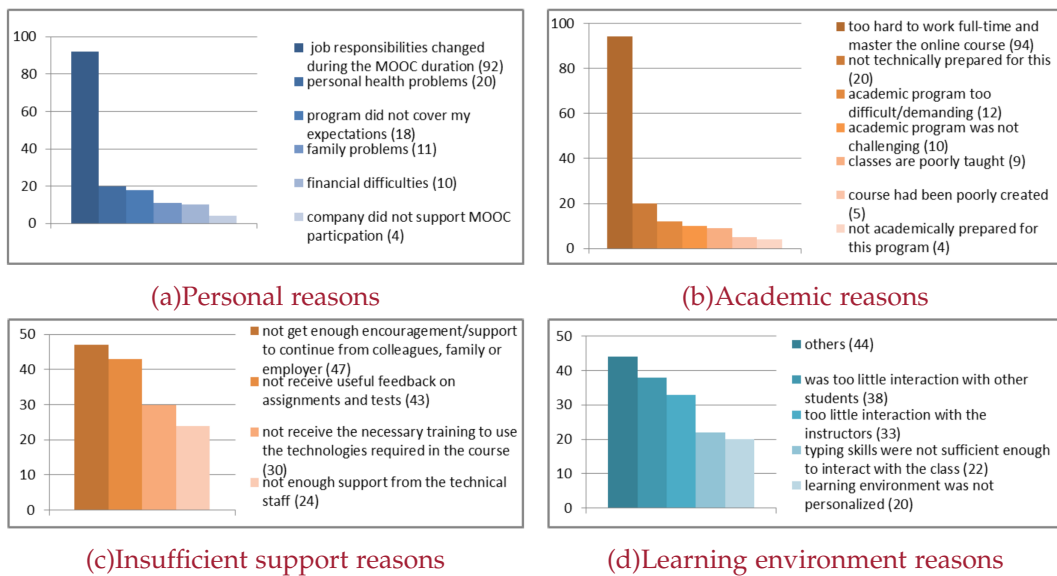


Figure 2.2.: Reasons why users left the MOOC (Gütl et al., 2014).

2. Background and related work

2.3. Simulations in education

As [Gibbons et al. \(1997\)](#) outline in their work, “a simulation simulates something.” That can be on the one hand things that really exist, for example a physics experiment demonstrating the gravity. On the other hand a simulation can be used to explore and present things which nobody knows if and how they exist like a black hole in the universe. Therefore a simulation creates a set of things and builds the relationships between them which can be defined from simple rules to very complex mathematical expressions. After a simulation is started a user can perform changes to the state or values of the simulation and observes the consequences ([Lunce, 2006](#)).

Simulations open up the possibility of changing things, that can not be changed in real life, for example, again, the gravity on earth. They can also be used for experiments which would be too expensive or simply too difficult in real life. Furthermore simulations are predestined for dangerous experiments that would risk human lives ([Lunce, 2006](#)).

Since many decades simulations were used in education to help students understand processes and coherences of various topics as described by [Clayton & Theodora-Ismene \(2005\)](#) in their work. Many students had troubles understanding the relevance of the topics they were taught or were not able to apply the learned material to real life problems. Here simulations can demonstrate applications of learned topics to simplify learning and understanding. Simulations can also help motivate students to learn very complex things or topics in which they were not so interested in. The higher interest leads to a deeper understanding and hence to learning more in less time ([Clayton & Theodora-Ismene, 2005](#)).

There are several advantages of using simulations in education. This paragraph is based on the work of [Rutten et al. \(2012\)](#). They mention in their work that students who attended a course which provided simulations beside traditional teaching could perform significantly better on research tasks. Furthermore, they analyzed the usage of simulations in laboratory activities. There, Simulations can be used for preparation for laboratory experiments which leads to better comprehension, increased interest and higher academic performance. Another idea was the usage of virtual laboratories. They can be used as an effective tool for preparation for students getting familiar with the laboratory setting and their workplace. However, [Rutten et al. \(2012\)](#) also mention that replacing real laboratories with virtual laboratories does not imply increased performance of students. Another study examined the replacement of real experiments with virtual experiments. Again, this

replacement did not automatically lead to better performance of students but using virtual experiments and especially 3D virtual environments for specific applications can lead to increased student's conceptual understanding. Rutten et al. (2012) concluded that simulations can be a good and useful enhancement of traditional education.

2.3.1. Simulations vs games

This section provides more detailed knowledge about the terms *simulation* and *game* and their differences. While working on this thesis, the virtual physics experiments were always declared as simulations. But when they were showed to colleagues almost every one of them said "*That's funny, let me play with it!*". It seemed that after all the changes and the work on the new simulations they were *accidentally gamified*. Accidentally, because this work was never intended to do so. But it was the reason to step into this matter during this master thesis.

The differences between games and simulations are described in this section. Subsequently the use of games in education is discussed in section 2.4 followed by a final description of the term gamification and its application in education in section 2.4.1.

During the research for this thesis about simulations, literature often also used the term *game* for simulations. Often the terms *simulation* and *game* were used in the same context. Interestingly, as shown by Crooltall et al. (1987), the term *game* can be used to refer to the term *simulation* but never in the other way around. He defined the differences between games and simulations. Many features are equal between both of them, but Crooltall et al. (1987) found two main differences:

1. A game does not represent a real world system. Of course a game can be based on real world situations and environments, like ego shooters, but they are not supposed to be realistic. In contrast, a simulation is always created to be as realistic as necessary because it should reflect real world behavior. An example here is the game Chess. It was initially made as a simulation but over the years the reference to real life got lost as the world has changed and nowadays it is definitely a game. A game has it's own world with it's own rules and rights while a simulation always tries to replicate the real world. That means a game is a real system while a simulation is just a meta system.

2. Background and related work

2. An error during a game can lead to real life costs, for example losing money by playing Poker. The costs of losing a game can be very small, maybe only a wounded pride. In contrast, a simulation can not be lost.

With this knowledge Crookall *et al.* realized that one can create a simulation of a game but he can not create a game of a simulation (Crookall *et al.*, 1987).

2.4. Games in education

Pim van de Pavoordt describes in his work (Van de Pavoordt, 2012) the use of games in education, or with other words, “*the gamification of education.*” This paragraph is based on his work. Pim van de Pavoordt shows that games can improve education. Main reasons are on the one hand fun while playing games. On the other hand he described the environment in which young people from today are growing up. The students of today use computers, laptops, smartphones and the Internet since they were very young. They are very familiar with this techniques and thus they feel comfortable with them. Hence games can lead to more fun during learning which leads to higher engaging. That does not mean that games or simulations are the new and ultimate way of teaching, they are rather an extension of traditional methods. For this purpose already existing games can be used as well as new games designed for education. A very famous game created for education is *Supercharged!* by Microsoft and MIT iCampus. It is “*an abstract world of electric charges, electric fields, magnetic fields, charged spheres - all of the basics of a physics textbook, but come to life in 3D*” for teaching physics at MIT (Jenkins *et al.*, 2003). Evaluations have shown that students who were playing *Supercharged!* performed about 20% better on the tests than students who were taught with traditional methods (Van de Pavoordt, 2012). This facts lead to thoughts about how to increase the use of games in education as well as how to teach different topics through games which led to the new expression *gamification*.

2.4.1. Gamification

As mentioned earlier in this chapter, during the work on this thesis several testers of the interactive physics simulations were calling them games. It seems there was an unintentionally *gamification* of the initial physics simulations.

Muntean (2011) describe in their work gamification as the use of game elements in non-game applications for raising motivation and engagement of students. E-learning provides a suitable framework for the implementation of gamification and courses with gamified elements (Kiryakova et al., 2014). Gamification should lead to having fun during the use of gamified applications and furthermore to spend more time with them (Muntean, 2011). Hence, Muntean (2011) expect better results on final tests from students which learned using gamified applications.

Basically school contains game-like elements (Lee & Hammer, 2011) like for example getting points for exams or “level up” to the next school level. Unfortunately, today's school system is not able to motivate students while video games and virtual worlds are very successful increasing gamers motivation and engagement (Iosup & Epema, 2014). Lee and Hammer describe in their work three major areas where gamification can extend traditional education (Lee & Hammer, 2011):

- Cognitive: Games specify only an overall goal not telling the player how to reach it or which way to take to achieve it. Players are intended to experiment and learn how to master different problems and situations. This supports students motivation and engagement.
- Emotional: Games can cause a lot of different emotions, especially positive emotional experiences like for example optimism, pride and joy but also curiosity and frustration. Positive emotions can increase students motivation. Furthermore games can help students to learn how to deal with negative emotions as they include failure and experimentation. *“Gamification offers the promise of resilience in the face of failure, by reframing failure as a necessary part of learning.”* (Lee & Hammer, 2011).
- Social: In games, students can slip in different roles and identities. This can help students to view challenges from different point of views.

Although gamification could be a great approach for increasing students motivation for learning, it can be challenging for teachers and instructors to gamify courses or even parts of courses to reach a successful and suitable system (Iosup & Epema, 2014). For example, Kiryakova et al. found out that tasks which are too hard or too easy to solve could be demotivating students (Kiryakova et al., 2014). *“Gamification is not a universal panacea (Lee & Hammer, 2011)”*. Literature lists several problems as well as recommendations for setting up courses with gamified elements. As written above, the difficulty of the tasks is very important. Therefore Kiryakova et al. recommend to increase the difficulty level from task to task (Kiryakova et al., 2014). Students

2. Background and related work

will have to enhance their effort for every task and apply already achieved knowledge and skills. Furthermore there should exist multiple paths for solving problems. This can lead to the development of different skills as well as forcing students to form their own strategies. Another element of games which could be applied to gamified courses are performance related badges (Iosup & Epema, 2014). Iosup and Epema recommend in their work to use only positive badges in education. A drawback of gamification could be that students will only be motivated to learn if they get external rewards. (Lee & Hammer, 2011).

Iosup and Epema found in their studies that it can be necessary to have an automated management system for bonus points, tracking students results and progress, etc., especially if a high number of students are taking a course (Iosup & Epema, 2014). Kiryakova *et al.* had the idea of using a learning management system (LMS) for gamification (Kiryakova *et al.*, 2014). They describe the advantages of this approach. Modern LMS allow the use of Web 2.0 tools which provide a useful framework for game elements in e-learning courses. Furthermore LMS already include tools for recording students behavior while working on the course to create a detailed report. They can provide forums or blogs for discussions and exchange of ideas.

2.4.2. Games vs gamification

As described in section 2.3.1 the boundaries between games and simulations are not clearly defined. And it is the same with games and gamification. Kiryakova *et al.* (2014) defines games and gamification in their work as follows: “*Gamification is an integration of game elements and game thinking in activities that are not games.*” They list a number of key points a game poses which are also part of gamification, such as challenges, tasks, points and levels, to point just a few of them. There are different concepts which are used in education like games, simulations, gamification, game inspired designs and serious games. All of them have in common that they use game elements to increase students motivation and engagement for learning.

2.5. TEAL and TEALsim

As described in (Dori & Belcher, 2005), teaching freshmen physics is a challenging task. The teaching material should motivate students to learn,

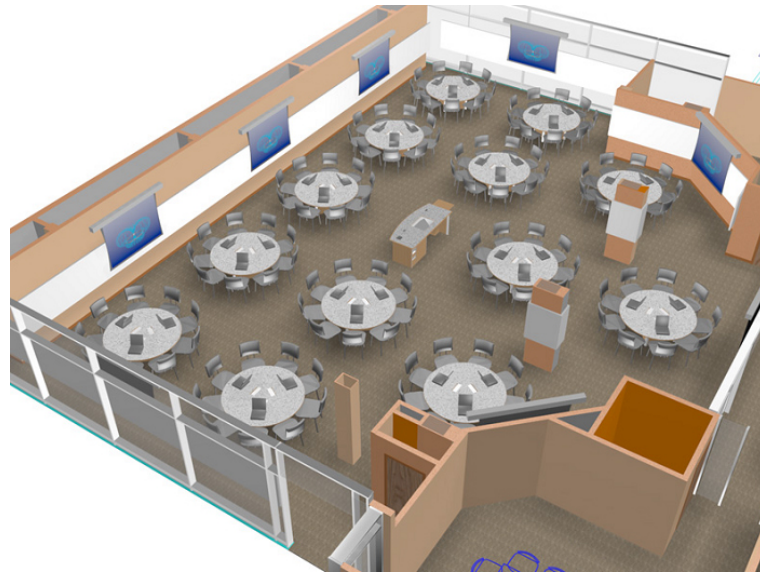


Figure 2.3.: A TEAL class room (Dori & Belcher, 2005).

as well as to understand the things they are learning. *Technology-enabled active learning* (TEAL) is a teaching format that was introduced in 2005. It combines classical lectures with interactive experiments and simulations (MIT iCampus website, 2014). 2014 different versions of introductory physics courses at the MIT where using TEAL (MIT physics department website, 2014).

“The TEAL classes feature:

- 1. Collaborative learning - students working during class in small groups with shared laptop computers*
- 2. Desktop experiments with data acquisition links to laptops*
- 3. Media-rich visualizations and simulations delivered via laptops and the Internet*
- 4. Personal response systems that stimulate interaction between students and lecturers”*

(MIT iCampus website, 2014)

A sketch of a TEAL classroom can be seen in figure 2.3.

2. Background and related work

“The course design is based on the following premises:

- 1. Interaction between teacher and student is an important factor in promoting learning*
- 2. Interaction among students is another*
- 3. Active learning is better than passive learning*
- 4. Hands-on experience with the phenomena under study is crucial”*

(MIT iCampus website, 2014)

Dori & Belcher (2005) showed in their work that the TEAL teaching format improved student’s conceptual understanding of the subject matter. Their research showed that students especially liked hands-on experiments, visualizations as well as interactivity. However, there are two drawbacks of TEAL (Pirker, 2013): First, the high implementation costs of a TEAL classroom which many universities can not afford. Second, TEAL is concentrated on teaching within the TEAL classroom and hence can not fulfill the requirements of e-learning approaches. On the other hand, traditional e-learning approaches can not fulfill the requirements of the TEAL teaching format. Especially interaction between students and between students and teachers is a problem of e-learning approaches (Pirker, 2013).

2.6. EdX - An online learning platform

The information about edX in this chapter is based on the edX webpage (edX website, 2014a). EdX is a non-profit platform that provides free online classes and so called MOOCs (massive open online courses). It was created by the Massachusetts Institute of Technology (MIT) in cooperation with Harvard and was launched in May 2012. EdX is Open Source and the code can be found on GitHub¹.

By today (Oct. 2014) there are 32 edX charter members and 12 edX members offering 157 online courses. Famous examples of members and charter members are Massachusetts Institute of Technology, Harvard University, Berkeley University of California, ETH Zürich, Caltech, University of Toronto and University of Hong Kong.

¹The source code of edX can be found on github at <https://github.com/edx/>

2.7. Evaluation of a prior 8.02x course

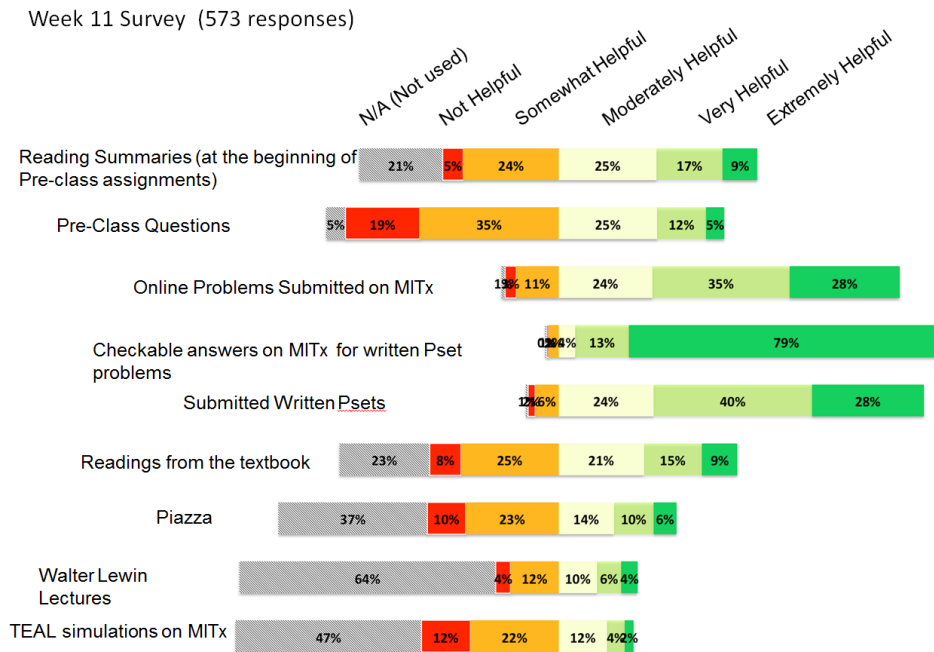


Figure 2.4.: Mid-Semester evaluation of 8.02x, spring 2014 (Rayyan & Chu, 2014)

The online courses are located in a wide variety of subjects including physics, law, history, science, engineering, business, social sciences, computer science, public health and artificial intelligence. The offered courses are free and accessible for everybody, meaning that anybody all over the world is able to join them.

EdX was created with three goals in mind (edX website, 2014a):

1. Open up access to quality education globally
2. Improve on campus education
3. Conduct research to understand more precisely how students learn

2.7. Evaluation of a prior 8.02x course

The outstanding positive feedback of prior terms of 8.01x and 8.02x was the impulse for this master thesis. The wish was to merge the improvement and extension of these courses. Therefore some results of the mid-semester evaluation of the physics course 8.02x 2014 are shown in figure 2.4. Very interesting for this thesis is the result for *Checkable answers on MITx for written Pset problems* as the goal of this work is to extend exactly the referred

2. Background and related work

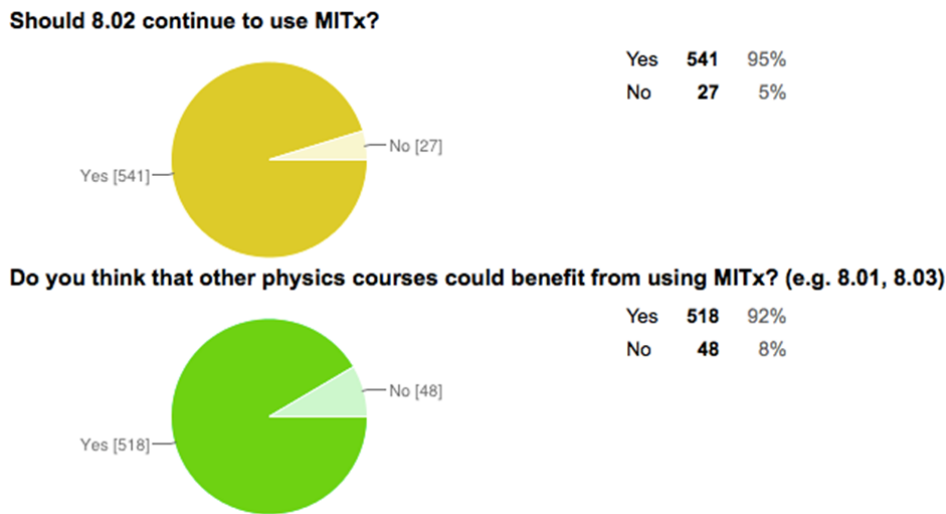


Figure 2.5.: Mid-Semester evaluation of 8.02x, spring 2014 (Rayyan & Chu, 2014)

problem set. Furthermore the result of two specific questions are shown in figure 2.5. The evaluation shows that MOOCs is working great for 8.02x and students really like the way they are taught. A detailed evaluation on the new courses and our work is presented in chapter 7.

2.8. The STAR framework

At the time this thesis was written, there still existed a small number of edX courses which did something very similar to what was done in this work. These courses were all developed from the same person and used the so called STAR framework² which was created especially for these courses. As there did not exist a paper or documentation of STAR at the time this thesis was written all the information about this framework is based on meetings with the developer of STAR as well as the source code and examples which can be found at GitHub³. It would take way too long to describe the whole implementation and functionality of the STAR framework in this section. The reason is that it was created a while ago when there was no functionality for interaction between edX and JavaScript applications implemented or at least only in early beta stage. STAR was mainly written with the Google Web Toolkit (GWT) which compiles Java code to JavaScript

²Star is a framework created by Ivica Ceraj at OEIT - Office of Educational Innovation and Technology.

³The source of the STAR framework can be found at <https://github.com/starteam/starx>.

code. So it struggled with a synchronization problem, as described in section 3.4, among other things. Hence the whole communication between edX and the GWT simulations was implemented in a new framework. The result is a huge library which provides everything for a stable interaction between GWT and edX. But there was a major difference to the work of this thesis. As described in section 4, this work claims a strict separation between the course creator and the simulation creator. Furthermore both did not want to write JavaScript code. At STAR that was different. The whole courses and simulations were written by only one person which additionally had outstanding JavaScript knowledge. Thus STAR courses had no separate interface. The code for interacting between edX and the simulation is rather spread over the course xml file, the GWT simulation and a HTML page connecting them.

2.9. Summary

With the increasing use of computers and the spread of network technologies and the Internet, teaching is continuously moving from the traditional classroom learning to electronic-learning (e-learning) (Zhang & Nunamaker, 2003). Today's students grew up with this technologies. Hence they are very familiar with them and feel comfortable using them (Van de Pavoordt, 2012). Online courses can be enhanced by interactive elements like simulations or games. This can increase the student's motivation and engagement for learning. Simulations can be used for small experiments as well as for whole virtual 3D environments for example to simulate a complete laboratory.

Literature often uses the term *game* for simulations. They are often used in the same context. The term *game* can be used to refer to the term *simulation* but never in the other way around (Crooll et al., 1987). Simulations and games are used to increase student's motivation with having fun while learning but also raise the conceptual understanding (Rutten et al., 2012). Using game elements in education led to the new term *gamification* and furthermore to "*the gamification of education*" (Van de Pavoordt, 2012).

A further development of e-learning are so called MOOCs, Massive Open Online Courses. Different kinds of MOOCs were formed in the last years and all have in common that they are open and free. While MOOCs mostly have a large number of enrollments also a high dropout rate is usual (Gütl et al., 2014).

2. Background and related work

In 2005 the *Technology-enabled active Learning* (TEAL) project was introduced (Dori & Belcher, 2005). It combines classical lectures with interactive experiments and simulations (MIT iCampus website, 2014). The two basic physics course 8.01 and 8.02 at the MIT are held in this TEAL classrooms. A further development was then TEALsim, a standalone environment including several virtual TEAL experiments. The last evolutionary step was to extend the physics course with an online system. Therefore the MOOCs platform edX was used. The online parts include learning materials like texts and videos but also questionnaires, homeworks and 7 ported TEAL simulations.

An evaluation of a prior 8.02 course has shown that students did not or only insufficient make use of the online simulations (Rayyan & Chu, 2014). Therefore these simulations should be extended to make them interact with edX. These should give course creators the possibility of asking students questions during homeworks which have to be answered by using the simulations. This extension of the online simulations is part of this thesis.

3. Technologies

Aim of this chapter is to explain the underlying technologies behind the online platform edX, the physics course 8.02x and the existing virtual physics simulations. It shows the used languages and describes their advantages. Furthermore the use of Java for creating JavaScript simulations using the Google Web Toolkit is described.

The basic physics course 8.02x at the MIT is held in TEAL classrooms using an online course at the MOOCs platform edX next to it (Dori & Belcher, 2005). This online course contains learning material, such as videos, simulations and written explanations but also homework and questionnaires.

EdX courses are written in XML and furthermore a LaTeX to pdf converter is also available for creating course pages (edX website, 2014a; MITxVM webpage, 2014). The course code can contain HTML code, Python code as well as JavaScript code which gives the course creator the possibility to create different and interactive online courses.

The structure of an online course can get quite complex, especially if the course contains many chapters, pages and questionnaires. For a better understanding of this work, the structure of the course 8.02x is explained in section 3.2.

Part of the course 8.02x are seven converted TEAL simulations, which were ported from the stand alone Java application TEALsim to the edX course. Therefore the Google Web Toolkit (GWT) was used. It allows developers to compile Java code to JavaScript code (Adam et al., 2013). A drawback of the ported simulations was, that they were embedded within an course page but running completely on their own with no communication and interactivity with the edX platform. As described in section 1 only a small number of students made use of the online simulations for the physics course (Rayyan & Chu, 2014). Therefore an interface between the simulations and edX should be implemented. This should give course creators the possibility of asking question which can only be answered by using the simulations. For this purpose edX offers the function *jsinput* (edX website, 2014b). It

3. Technologies

implements three functions for a communication between edX code and simulation code.

3.1. EdX and the mix of HTML, Python, JavaScript and LaTeX

The edX platform provides different ways for the creation of online courses, as described in section 3.2 (edX website, 2014b). Generally speaking, when an edX course is opened in a browser everything shown is always HTML code that could include JavaScript code. This does not mean that the whole course is written in HTML and JavaScript. The method used in this work includes XML and Python. XML is used for the whole course content like all the text as well as text-boxes or check-boxes for the online homeworks but also for hints and shown answers to questions. Python is used for all the processes behind the visible course site. For example, for creating random variables, different calculations and computations as well as evaluating given answers. Furthermore LaTeX was used for creating the course pages. Normally, LaTeX is used for writing books, papers or journals but it can be also used for creating edX courses using MIT's latex2edx¹ compiler.

3.2. Main structure of an edX online course

The goal of this thesis is not to describe how to create a whole edX course. However, for understanding how to embed the physics simulations in the edX course it is important to be familiar with the main structure of such a course.

Basically there are three different ways of creating an edX course (MITxVM webpage, 2014):

1. Using edX studio: Studio is a browser based course management system². It provides everything needed for creating an online course including an online editor, calendar, easy drag and drop tools and many more.

¹The latex2edx compiler can be found at https://edge.edx.org/courses/MITx/MIT.latex2edx/2014_Spring/about.

²A detailed description of edX Studio can be found at <https://studio.edx.org>

3.2. Main structure of an edX online course

2. `latex2edx`: With this tool a whole course as well as individual problems can be written in LaTeX. The LaTeX source is then converted to edX XML code. This converter was implemented because of the common use of LaTeX at universities and the well-known highly suitable mathematical content of LaTeX. This method is used by the group at MIT which created the course 8.02x.
3. Using a plain editor: This method is of course the most complicated way to create an edX course but also gives the most capabilities and for programmers it is the most familiar way. Furthermore, it is often used for editing existing courses where other tools reached their limits. This was the method of choice in this master thesis.

Regardless which method is used for creating and testing a course, a test environment is needed. Therefore edX offers a test server³. For this work an Oracle-VirtualBox was used for running a virtual Linux machine together with Vagrant for setting up the development environment. EdX offers a `.box` file which includes the Ubuntu Linux distribution with edX running and everything that is needed therefore. The edX test system can then be reached using a browser calling `http://192.168.42.2`. The information on the virtual edX server is based on (MITxVM webpage, 2014). On this test environment a user account is precreated which can be used for logging into the system and signing up for the created courses. This can be done by using the email `xadmin@mitxvm.local` and password `xadmin`. Of course many different users can be created. For more details see the manual linked above.

3.2.1. A basic setup of an edX course

An example of an edX course can be seen in figure 3.1. It shows the physics course 8.02x with the site for an interactive homework opened.

Every course contains an horizontal top menu with links to general course information and a vertical left menu which includes the course material divided into weeks. In the top menu two sections are particularly interesting for this work:

- Courseware: This tab contains the main online course including Pre-Class assignments, online problems, forms for hand written homework, visualization and other course materials.

³A detailed manual can be found at: <http://people.csail.mit.edu/ichuang/edx/>.

3. Technologies

edX MITx: 8.02x Electricity and Magnetism zeleznik

Courseware Updates Course Info Textbook Download PDFs Calendar Discussion Wiki Progress

Introduction
 Welcome from Prof. Lewin (Video)
 Course Tutorial (Video)
HW0: Learn about Homework (Homework)

Week 1
 Week 2
 Week 3
 Week 4
 Week 5
 Week 6
 Week 7
 Week 8
 Week 9
 Week 10
 Week 11
 Week 12
 Week 13
 Visualization
 TEALsim
 Exit Survey

SYMBOLIC QUESTIONS (6 points possible)

Often you will be asked questions requiring to enter a symbolic expression for the answer. Here are some examples:

(a) What is the magnitude of the gravitational attraction on m caused by a mass M located at distance r ? The distance between the two masses is r . Express your answer in terms of the gravitational constant G , M , m and r .
 Examples of valid answers are $G*M*m/r^2$, G/r^2*M*m , $(G*M*m)/r^2$.

Note that:

- Capitalization is important! For example, if you type $g*M*m/r^2$, the answer is considered wrong.
- Always use $*$ to indicate multiplication. The system will not accept GMm/r^2 . Also, as a good practice, do not use spaces to represent multiplication (On occasions, the answer will be accepted, but most of the time it will generate an error message).
- you can use parentheses in your expression, but make sure you close all of them. An error message will appear if you have mismatched parentheses, for example try: $(G*m*M/r^2$
- as you type the answer, you will be able to see the mathematical expression as you type. Look and make sure the expression looks like the answer that you are trying to enter.

(b) Some equations will contain indices. Use underscore "_". Example: enter the expression $x_0 + \frac{1}{2} a(t_1 - t_0)^2$ below: Examples of valid answers are $x_0+1/2*a*(t_1-t_0)^2$, $x_0+0.5*a*(t_1-t_0)^2$.

(c) Some equations will contain greek letters. According to Gauss's Law, express the Electric flux over a given closed surface area in terms of the charge enclosed q_e , the total charge q_t and ϵ_0 . An example of valid answer is q_e/ϵ_0 .

NATURAL CONSTANTS In the various problem sets, you might need to use *natural constants*. With that expression we mean any of the following variables: ϵ_0 , μ_0 , the speed of light in vacuum c , π , and the gravitational constant G . Try the following string: $\epsilon_0 \mu_0 c \pi G$. This is realized typing $\epsilon_0\mu_0c\pi G$.

Check Save



Figure 3.1.: An example for an edX course with an interactive homework opened.

3.2. Main structure of an edX online course

- Instructor (only visible if logged in as admin): This is an important section for creating a course. Here one can perform a *Reload course from XML files*. This has to be done if the course source XML files were changed.

From the left menu the homework links are particularly interesting for this work. They will point to the TEAL simulations.

3.2.2. The setup of the courses 8.02x

This section is important for understanding how to extend standard edX problems for working with GWT compiled HTML pages. It describes the setup on the pre-class assignment of week one, page 3 of the MITx course 8.02 of spring 2014. As described in chapter 7 the experiments written in GWT will be compiled in one single HTML file. For embedding it, an iframe which is an HTML5 compliant tag is used. Embedded HTML files should be located in the static/html folder. This is the place for non edX created, complete and static HTML files that are included in the dynamically created edX course.

The HTML file is included by hand written source code with an iframe tag:

```
1 <iframe id="teal-iframe"  
2   src="/static/html/gwt-teal/PCharges2.html?size=400"  
3   width="818"  
4   height="475"  
5   scrolling="no"  
6   frameborder="1" />
```

Listing 3.1: Including a HTML page in an edX course.

The benefit of this method is that the course page is dynamically created by edX using XML. As described in section 3.2.3, a whole problem set can be created including questions, check-boxes, etc., only the experiment itself is external HTML code. Here, the main problem and limitation of edX can be seen: It is not possible to interact between edX and the GWT created simulation.

3. Technologies

3.2.3. The segmentation of the edX course 8.02x in separate files

The physics course, for which the simulations are written, is separated in several files. At the beginning the file structure can be confusing. Therefore the hierarchy of a course page is described in this section. An example page of the edX course 8.02x can be seen in figure 3.2. The colored areas are defined by the following file names with matching numbers and coloring.

- 1 **\802r-TTh \course.xml**
This file only holds a link to the next file, *2014_Spring.xml*. It is the base file which prompts edX to create the main course page with the horizontal menu at the top of the page and all the administrative information.
- 2 **\802r-TTh \course \2014_Spring.xml**
With this file the chapters of the course are created. In our example course 8.02x there are always two chapters for one week. One for pre-class assignments and one for the weekly homeworks.
- 3 **\802r-TTh \chapter \pcq \W01pcq.xml**
Here the second horizontal menu, beneath the first one is created. Every pre-class assignment or homework is divided into smaller parts which can be reached over this menu.
- 4 **\802r-TTh \vertical \802r \W01_D1_q1.xml**
This is the file where the actual course data is written. It contains explanations, questions, hints and much more. This is also the place, where the embedded TEAL simulations are included.
- 5 **\802r-TTh \static \html \gwt-teal \PCharges2.html**
This is the HTML file with the simulation. It contains everything that is needed for running the simulation. It is created from GWT and compiled from the Java code. The simulation.html file is fully executable on it's own. That means it is just embedded in the edX course.

3.2. Main structure of an edX online course

The screenshot displays the main structure of an edX online course page. The page is titled "MITX: 8.02r Electricity and Magnetism" and features a navigation bar with tabs for Courseware, Updates, Course Info, Textbook, Sections, Syllabus, Progress, Gradebook, Static Course Page, Instructor, and Staff view. The Courseware tab is active, showing a sidebar with a list of assignments, including "F Feb 7: Electric charge and Electric Field". The main content area is titled "TEALSim Exploration: Point Charges" and contains an interactive simulation. The simulation interface includes a central visualization of electric field lines and vector arrows, a "Parameters" section with sliders for charges Q1 and Q2, and a "Field Visualization" section with checkboxes for "Show Vector Field Grid" and "Field Lines". A "Grass Seeds" button is also present. The page includes a text introduction to the simulation, a "More about this simulation" link, and a quiz question: "Did the Simulation work? choose one answer and click save." with radio buttons for "Yes" and "No". The page footer contains links for About, Jobs, Press, FAQ, and Contact, along with social media icons and the edX logo.

Figure 3.2.: An example for the segmentation in separate files of the edX course 8.02x.

3. Technologies

3.3. GWT - Extending edX with a new programming language

This chapter and all information about GWT are based on (Adam et al., 2013). The Google Web Toolkit (GWT) is a toolkit that allows writing code in Java and compiling it to JavaScript. The main difference to other frameworks is that GWT provides the possibility to write server side as well as client side code in Java. For embedding the virtual experiments in online courses JavaScript is needed, but writing such big applications directly in JavaScript is difficult and complex since it was not developed to implement large projects. Because of that, there exist almost no kind of debugging system for JavaScript. That is a huge advantage GWT, as code could be written in Java using the well known Java environment and test it with a powerful debugging system beneath it. Furthermore, the existing TEALsim experiments are written in Java. Unfortunately, this does not mean that the Java written experiments could be compiled straight forward to JavaScript using GWT. The TEALsim code contains a large number of external libraries that are not supported by GWT. At least, parts of the already existing code could be reused. Another advantage of GWT is that if JavaScript Code is needed that is not available in GWT the Java code can be weaved by JavaScript Code. This is important for example for accessing browser variables. Furthermore, GWT provides a great number of templates and libraries, such as HTML forms and input text boxes.

3.3.1. The GWT compiler

The main function of the GWT compiler is compiling Java code to JavaScript. While doing that it analyzes the written code and optimizes and minimizes it. The created code is then as compact as possible in order to keep data that has to be sent over the Internet as small as possible. The compiler also runs so called generators which are part of GWT and create code at compilation time. A generator replaces specific code, such as templates or other simplifications, and replaces it with new generated and working code. This decreases the amount of code that has to be written by the programmer. There exist a large number of generators which can also be self written. The compiler will call itself the right generator depending on the code.

Another important task of the compiler is to build different versions of JavaScript code for different browsers and therefore also a bootstrap loader.

3.3. GWT - Extending edX with a new programming language

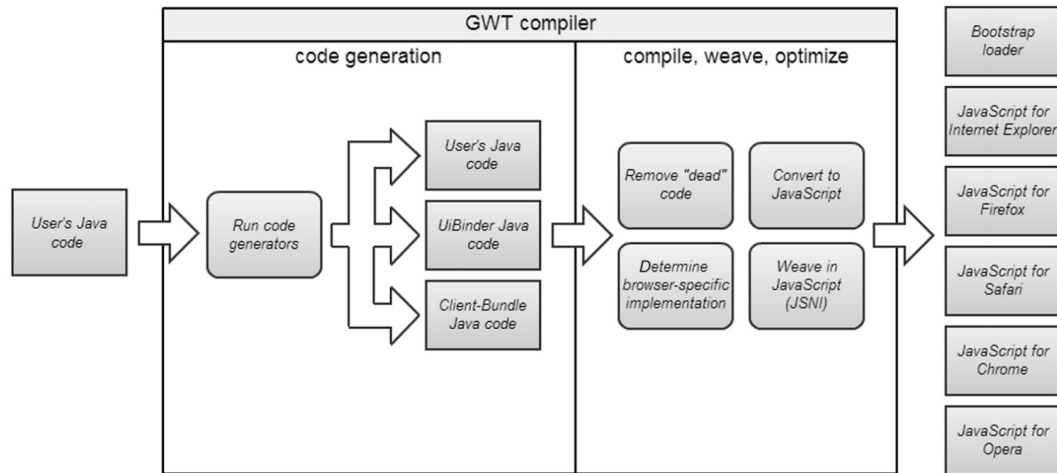


Figure 3.3.: The GWT Compiler in Detail (Adam et al., 2013)

The bootstrap loader will be running on the client side and recall the right version for the client's browser. This method is called *deferred binding* and allows the programmer to take into account differences of how individual browsers are handling specific JavaScript code. An illustration of the GWT compiler can be seen in figure 3.3. How the generated code will be shown to a user can be seen in 3.4. A user enters a URL in the browser's address field. The browser and the server then interact with each other to create the appropriate application for the user and its system.

3.3.2. Advantages of GWT

There are many advantages of GWT especially the fact of being able to use Java, a high level object-oriented programming language, instead of JavaScript. Moreover GWT is perfectly adjusted for communicating between client and server. This is needed to communicate between an iframe and its parent site, which was written using a mix of XML, python, HTML and JavaScript. In conclusion there is no direct access from GWT to edX variables and vice versa.

3. Technologies

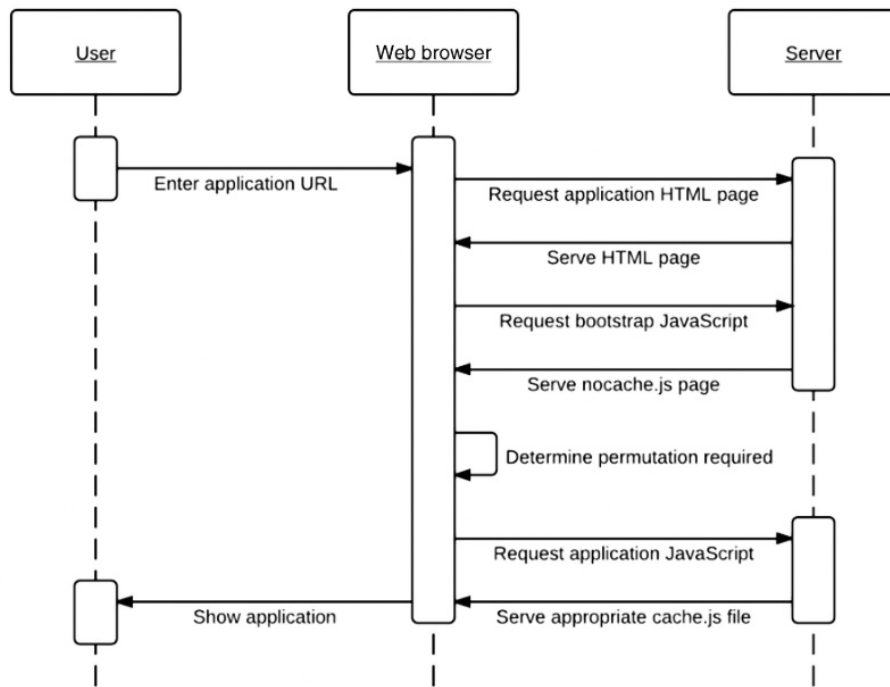


Figure 3.4.: Starting a GWT application (Adam et al., 2013)

3.4. The jsinput sample problem

The information of the edX function *jsinput* in this chapter is based on the edX documentation (edX website, 2014b). *Jsinput* was created to give course creators the possibility of creating interactive JavaScript simulations and to let the students answers be graded and stored by edX. It is an interface between edX and the JavaScript simulation. Therefore, *jsinput* includes three functions: *GetGrade*, *setState* and *getState*. *GetGrade* can be used for returning an answer of a student from the JavaScript simulation to edX and grade it afterwards. *SetState* and *getState* can be used for saving and restoring the current state of the simulation. For more details on this functions see section 5.2.2.

At the time this master thesis was written *jsinput* was still in development and, to the best of our knowledge, there existed no online course which used it. Furthermore there was no up to date and complete documentation, only a more or less stable working sample problem. Though *jsinput* looked quite useful for this project there were two problems. First: The requirement of setting an initial state for the simulations, as described in section 4, could not be realized with *jsinput*. Second: The existing simulations were written

in GWT. This may not look like a problem because GWT compiles the Java code to JavaScript but there was a synchronization problem. In conclusion *jsinput* would try to call functions from the compiled JavaScript code which may not have been loaded yet. The reason for that can be found in the code optimization of GWT. A detailed description of this problem and solution is given in section 5.6. For this project, this meant a plain *jsinput* implementation would not work. A more detailed description of *jsinput* can be found in the edX-manual ([edX website, 2014b](#)) and also in section 5.2.2 of this work.

3.5. Summary

There exist different ways for creating an edX course. The main course code has to be written in XML. The pages of the online course 8.02x are mainly written in LaTeX and converted to XML using the LaTeX to XML converter offered by edX.

8.02x contains also seven ported TEAL simulations which can be used by students for learning and better understanding the course matter. An evaluation of a prior course showed that only a few students did make use of the simulations while learning. This led to the idea of creating an interface between edX and the simulations to give the course creator the possibility of asking questions which can only be answered by using the simulations.

The 7 TEAL simulations were written in Java and compiled with the Google Web Toolkit (GWT) to JavaScript. Using Java had numerous advantages. The most important one was, that the developer of the simulations was using Java for many years and hence had outstanding knowledge about it. Furthermore, Java provides a powerful debugging system.

For interacting with JavaScript applications edX offers a function named *jsinput*. This function provides everything needed for a communication between an edX course page and a JavaScript application. At the beginning of this work it seemed, that this function can be used for implementing the interface between edX and the 7 TEAL simulation. Indeed the compiled TEAL simulations are a JavaScript application but the JavaScript code was build by GWT from the Java code. The problem is that GWT is optimizing the Java code while compiling it to JavaScript code. This led to a synchronization problem between edX and the simulations. Furthermore *jsinput* offered no

3. Technologies

possibility of sending an initial state to a JavaScript application at startup. However this was a main claim for this work. These two points described above are the reason why this work could not be realized only by using *jsinput*.

4. Problem definition

This chapter states the goal of this master thesis. First, a short overview of the existing physics courses at the MIT and the way they are taught is given. Afterwards the new requirements for the extended physics simulations are described. Furthermore the most important points for the following implementation are shown in more detail.

The following introduction is based on information of the websites of edX, TEALsim and MIT (edX website, 2014a; MIT webseite, 2014a; TEALsim website, 2014). During the first semesters of each MIT study every student has to pass one version of the basic physics courses 8.01 and 8.02 (MIT webseite, 2014b) which are held in TEAL classrooms. In spring 2014 the physics department ran an experiment to enhance the on-campus physics course 8.02 by the edX platform (Rayyan & Belcher, 2014). This new course was called 8.02x with the x at the end of the course number indicating the use of edX¹. In the edX system students can find the complete course material, videos of the lessons as well as administrative details. Furthermore, students have to complete online homework, such as answering multiple choice questions or filling in text boxes. Handwritten answers are also submitted through the edX system. 8.02x made use of 7 TEAL simulations which were accessible through the edX platform. These simulations should support students to understand various topics of the course. An evaluation of a foreign course has shown that most students have not used them sufficiently and many even ignored them entirely (Rayyan & Chu, 2014). There were various reasons for this behavior. Many students mentioned their tight schedule as studying at the MIT is very time consuming. Some students simply did not understand the experiments, others were just too lazy to work intensively with them and some were completely not interested (Rayyan & Chu, 2014). This led to the idea of extending the edX course as well as the simulations in such a way that a number of homework and exam questions could only be answered by using the virtual experiments. The

¹The physics online course 8.02x can be found at <http://www.edx.org>. At the time this thesis was written one can register at edX for free. Depending on the accessing time point only an archived version of the course could be available.

4. Problem definition

desired result is that students truly understand the learning material and, as a consequence, have a strong advantage when solving homework and exam problems.

During this work the existing virtual TEAL simulations will be extended to achieve an interaction between them and the edX platform. Sending data between the simulations and edX offers the possibility of asking question within an edX course page which can only be answered by using a simulation. The answer is then evaluated and graded by the edX platform.

4.1. The goal of this master thesis

The main problem of the TEAL simulations used in the edX course was their lack of interactivity between them and edX. The goal of this master thesis is to create an interface between edX and the simulations for giving the course creator the possibility of asking students questions which can only be answered by using the simulations. This led to four requirements. The simulations need to -

1. - be adapted for being able to ask useful questions.
2. - get an initial state from edX.
3. - be able to return an answer to edX.
4. - be able to reload a previous state giving the student the possibility of making a break and continue working at a later point.

Furthermore, the simulations had to be strictly separated from the edX course. The reason behind this requirement was the fact that for 8.01X and 8.02x there was a course creator and a simulation creator and the interface should provide the interaction between them. Ideally that means that the course creator has no knowledge about the simulation's code and the simulation creator no knowledge about the course's code. They should meet each other just once at the beginning of building a new simulation for a course. They discuss how the new simulation and the embedding course page should look like, what they should do and finally which data should be sent between them. And after that every one should be able to write his code without thinking about the other's work.

The first simulation to be adapted was the point charges experiment which is also used as an example here. The experiment can be seen in figure 4.1. It has two point charges which can be placed anywhere within the dark blue box. After starting the simulation the point charges will start moving

4.1. The goal of this master thesis

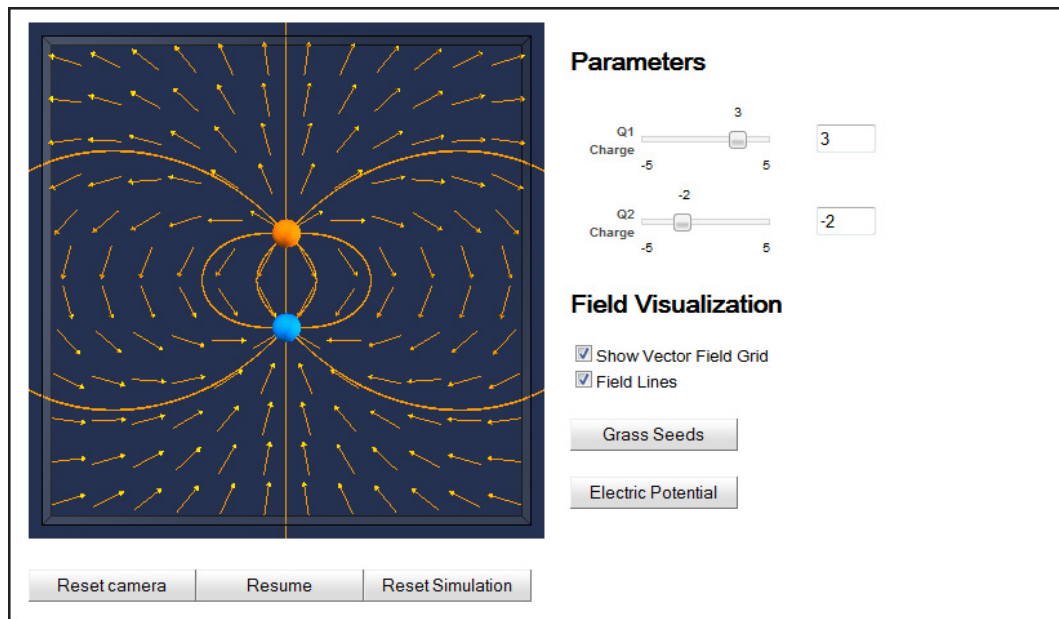


Figure 4.1.: The original point charges simulation.

depending on the influence of their charge on each other. Furthermore, the field lines are shown. The student can adjust the charge of the point charges by using the sliders.

4.1.1. Adapting the simulations for answering questions

The first task that had to be done was to find the questions students could be asked and how the simulations had to be adapted to answer them. The questions should be in a way that they make the students work intensively with the simulations, understand the physical processes, be well prepared for the exam at the end of the term and ideally obtain knowledge for the rest of their life. Observing the example simulation (see figure 4.1), the point charges experiment, it can be seen that there are almost no useful exercises which can be answered by using it. So it had to be adapted. A first idea was to hide one point charge and the student has to find it by moving the second charge. He then places a pointer where he thinks the hidden charge is located and submits his answer.

4. Problem definition

4.1.2. Getting an initial state from edX to the simulation

Taking the example simulation of the previous chapter 4.1.1, the point charges simulation, one problem can be found easily: Cheating. Students will of course work and learn together. If every student gets the same question, which would mean every student gets the same position of the hidden charge, it would make it too easy for them to cheat and the learning effect would be gone. Therefore every student had to get a different position for the hidden charge. Within the code of an edX course page it is possible to create random numbers using python (edX website, 2014b). Moreover, it is selectable if every student gets one random number per experiment, no matter how often he repeats the simulation, or if he gets a new random number on every attempt. Hence the position of the hidden charge should be created within the edX XML code and sent as an initial state to the simulation.

4.1.3. Returning an answer to edX

Once a student submits an answer it has to be evaluated. This has to be done by edX for two reasons. First, as described in the introduction of this chapter, the course creator should have as much latitude at creating a question set as possible. For example, that gives the course creator the possibility of deciding how exact the place of the hidden charge has to be found to be graded as correct. Furthermore, again edX provides some very useful advantages. The most important, edX saves the answers of the students. And even more, it saves all given answers as well as how much time the student has invested to find the answer (edX website, 2014b). That gives the course creator an overview of how hard or easy it was for all the students to find the answer and how he should adjust questioning for the next term. Therefore it was needed to return the answer from the simulation to edX.

4.1.4. Reloading a previous state

It is an important feature of edX courses that students can make a break during a learning task or for exercises to edit their answers till the deadline. This feature was also demanded from the simulations and again edX can help reaching this aim. EdX not only provides a *Check* button, it also provides a *Save* button (edX website, 2014b). Compared to each other these buttons do

mostly the same. Both return the current state of the simulation to edX and save it. The only difference is, that at save the answer is not graded. But as said, both save the answer. So for reloading, the simulation has just to be reloaded with the saved state instead of the randomly created initial state. Further details are provided in chapter 5.

4.2. Summary

This chapter describes the points that should be achieved within this master thesis. The MIT basic physics course 8.02x were held using the edX platform, including seven virtual physics experiments, so called TEAL simulations, for teaching and helping students understand the topics. These simulations were stand alone applications running in an embedded HTML page with no possibility of interacting with edX and no option of asking students questions which could only be answered by using them. The first problem was to find an as effective as possible way of adapting the existing simulations for a suitable questionnaire. With this knowledge an interface between edX and the simulations had to be created with the claim to provide everything needed for a stable communication. Therefore the interface had to support loading an initial state, returning the students answers and saving and reloading the current state of the simulation for giving a student the possibility of resuming his work after a break. Another very important task was to keep the strict separation between course creator and simulation creator as these are two different persons for the physics course who were not familiar with each other's tasks. The work flow is exemplified on the first simulation that was extended during this master thesis, the point charges simulation, which can be seen in figure 4.1.

5. Design and development of the Interface between TEAL Simulations and edX

Aim of this chapter is to show the implementation of the interface between the edX platform and the existing virtual TEAL simulations. At the beginning a new prototype simulation is described which will be used during this chapter to describe the different parts of the implementation, the underlying mechanisms and the upcoming problems. The first point, that will be described is, how to grade an answer from a simulation within edX. After that, the sending of initial values for starting a simulation is explained. The following section describes the loading, sending and saving of a state of a simulation. At the end a synchronization problem, which is a result of using a GWT compiled Java simulation within an edX course, is shown in detail.

The basis for this work was the course 8.02x from 2013. This course contained seven TEAL simulations which were used in so called pre-class assignments. As described in chapter 4.1 these simulations should be extended for asking students questions which have to be answered by using them.

The first simulation which should be extended was the *Point Charges* simulation. The original Point Charges simulation can be seen in figure 3.2. Initially it had two point charges, a positive and a negative one. Originally, the student could move them within the bounding box and observe the field pattern created by the charges.

From the programming point of view, the edX course page including the simulation is a problem-set XML file. This file contains the whole course text for simulation description, background information and everything else. Furthermore it embeds the simulation with an iframe element. The structure of this page is illustrated in 5.1.

5. Design and development of the Interface between TEAL Simulations and edX

As described in chapter 3.3 the simulations are written in Java and compiled with GWT to JavaScript. Therefore GWT creates a HTML host page which then embeds the JavaScript simulation. That means that in the edX course XML page a normal HTML page is embedded no matter what this page contains or how this page is created. EdX treats all HTML pages in the same way. They will be embedded statically and there is no possible interaction between the edX course and the HTML file (edX website, 2014b).

At the beginning, a simple prototype simulation is created to explore and describe the development of the interface. This first extended simulation is not meant to be ready for using it in teaching students within 8.02x. It is used to develop the interface and test the communication with edX.

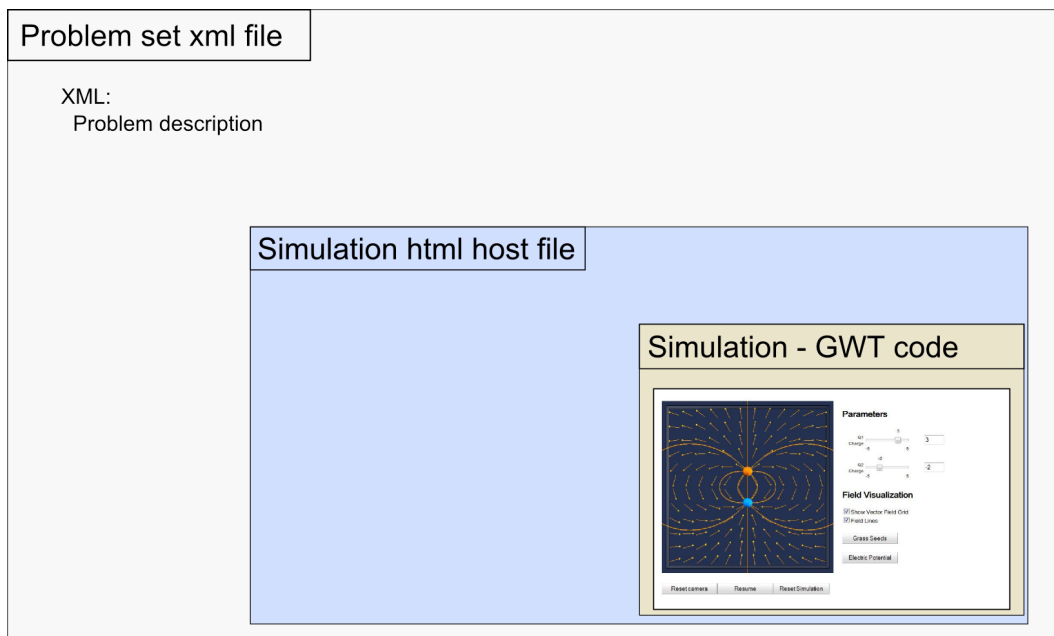


Figure 5.1.: Structure of an edX course page which includes a non interactive simulation.

5.1. The first extended simulation

For better understanding the modified simulations will be called *extended simulations* in this work. Figure 4.1 shows the basis for the first simulation that was extended, the point charges simulation. This simulation should be changed to give students an assignment which has to be solved by using this simulation.

5.1. The first extended simulation

The idea for the new simulation was to place two hidden charges and the student has to find them by moving a visible point charge and observing the field lines. The extended simulation can be seen in figure 5.2. It has two boxes. The inner box describes the area where the hidden charges are placed. Between the inner and the outer box is the area where the moving point charge is placed and can be moved by the user. That means that the student is not able to move the point charge inside the inner box, what would make the task too easy. Instead he has to observe carefully the field lines of the moving charge to predict the places of the two hidden charges. For answering, the student can place so called dummy charges. These are point charges with no effect on the electrical field. They are used for indicating where the student assumes the hidden point charges are placed.

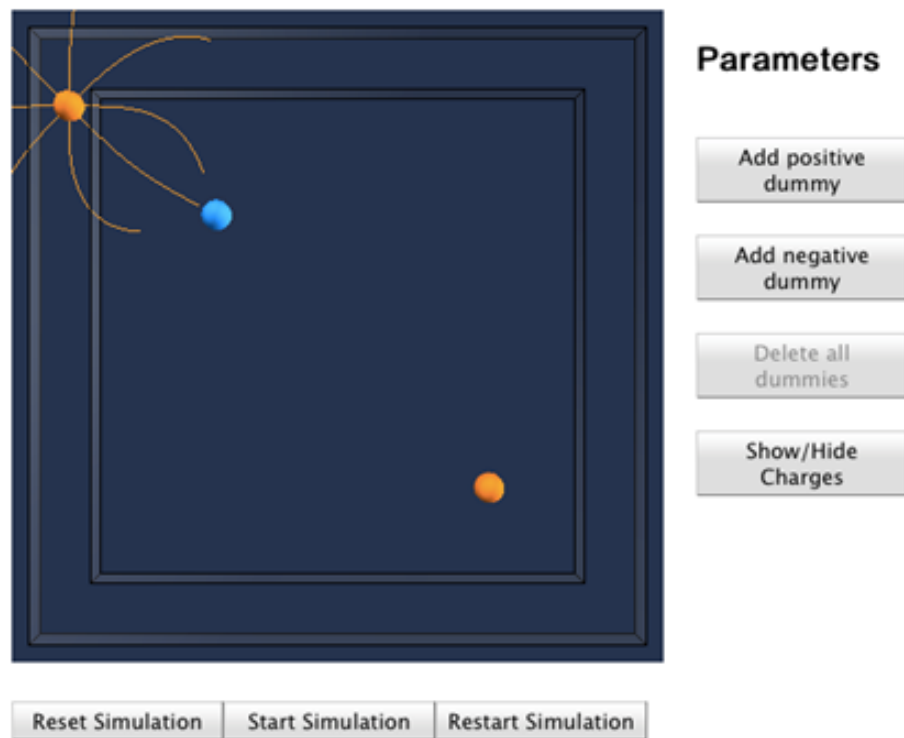


Figure 5.2.: The first extended simulation. The point charge on the left top is the movable point charge with visible field lines. The two others are dummy charges with no effect on the electrical field.

The requirements on the extended simulation are:

- Place a number of hidden charges. For making cheating impossible the hidden point charges should be placed randomly. The random places

5. Design and development of the Interface between TEAL Simulations and edX

should be created by edX and sent to the simulation when starting it. Every student will get an unique set of hidden charges. The simulation will get the initial values for the hidden charges from edX.

- When the student presses the *Check* button, the simulation has to send the places of the dummy charges to edX which then verifies the answer by comparing the places of the hidden charges with the dummy charges.
- Enabling or disabling the Show/Hide button. This button can toggle the visibility of the hidden charges. For learning assignments this button can be enabled, for exam assignments it will be disabled.

5.2. Grading a simulation

The first task of this work was the grading of a student's answer by using the simulation. A GWT created simulation contains a HTML host page which then embeds the JavaScript simulation. For sending the answer from the GWT simulation to edX, the edX function *jsinput* was used¹. *Jsinput* was implemented to communicate with JavaScript applications embedded in an edX course. More precisely it embeds a HTML host page containing the JavaScript application and calls JavaScript functions within the HTML host file for communicating (edX website, 2014b). This functions do not have to be part of the application as long as they exist and accept and return the correct parameters. For this work that meant, for the beginning, the embedded simulation are reduced to the HTML host file holding variables for manual created answers. This variables will then be updated later by the simulation but for now it is enough to have them static as they simplify the task.

For the beginning, the HTML host file has two variables:

1. *positions_hidden_charges* - This is a list of the x and y coordinates of the hidden charges as well as their charge.
2. *positions_dummy_charges* - This is a list of the x and y coordinates of the dummy charges as well as their charge.

As mentioned before, *jsinput* is used for communicating with the HTML host page. *Jsinput* itself has to be enclosed in a edX command called *custom-response* (edX website, 2014b). These two functions are described in the next

¹J.M Claus from edX recommended the using of the edX function *jsinput*.

sections 5.2.1 and 5.2.2. Applied to the example in the introduction of this chapter, the structure has changed as shown in figure 5.3.

The problem-set XML file contains now the *customresponse* tag that includes the *jsinput* tag. *Jsinput* embeds then the HTML host file. Furthermore the XML file includes a python function for grading an answer. The data stream in detail: *Jsinput* communicates with the HTML host file. When a student clicks the *Check* button in the edX page *jsinput* will call the function *grade_function* of the HTML host page which will return the value of the variable *position_dummy_charges* which holds the student's answer. *Customresponse* calls the python grade function of the edX XML page which grades it. The output of this python grade function will be processed by edX. That means for example that edX saves the answer, it records all given answers for evaluations or give the student a mark based on the answer. All this is done by edX itself, the python grade function has just to determine if the given answer is right or wrong (edX website, 2014b).

5.2.1. EdX - customresponse

Customresponse is used to evaluate student's answers using a python script. A code example can be seen in listing 5.1. The *customresponse* tag contains a single text-line where a student can enter the answer. The exercise in this example is to calculate the sum of 6 plus 4 and enter the answer. For grading, *customresponse* takes two parameters. The first parameter *cfn* takes the name of the python check-function. The second parameter *expect* is the expected answer. *Expect* does not have to be set. In this case, the check function has to get the expected value from somewhere else or calculate it on its own (edX website, 2014b).

In this example, when a student enters an answer and presses the check button, *customresponse* will take the entered value and call the check function. The python function then compares the given answer with the expected answer and returns either *Right answer* or *Wrong answer*.

5.2.2. EdX - jsinput

Jsinput is a method that enables edX to communicate with standalone HTML files (edX website, 2014b). This is useful if an application is written in JavaScript and should be embedded in an edX questionnaire. Therefore any application like GWT can be used for creating the simulation. *Jsinput*

5. Design and development of the Interface between TEAL Simulations and edX

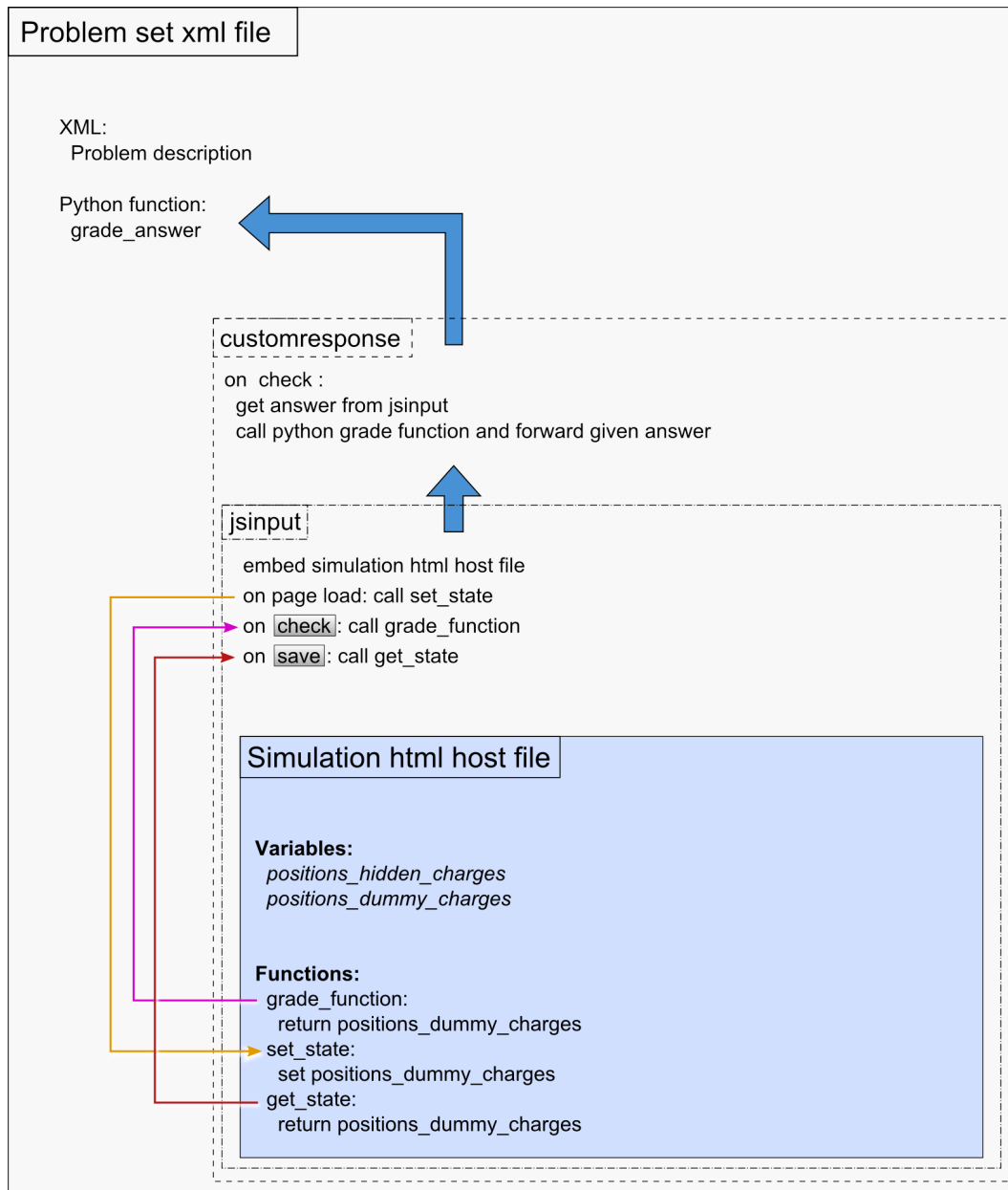


Figure 5.3.: Structure of a course page that embeds a HTML file using jsinput. Illustrating the data stream.

```

1 <problem>
2   <script type="loncapa/python">
3     def grade_answer(expect, answer):
4       sum = int(answer[0])
5       if sum==int(expect):
6         return{'ok': True, 'msg': 'Right_answer.'}
7       else:
8         return{'ok': False, 'msg': 'Wrong_answer.'}
9   </script>
10  <p>What is the sum of 4 + 6 ?</p>
11  <customresponse cfn="grade_answer" expect="10">
12    <textline size="10"/>
13  </customresponse>
14 </problem>

```

Listing 5.1: A Customresponse sample problem, based on (edX website, 2014b).

provides three functions for interacting with a HTML file. They can be used for grading student's work with the simulation, saving and reloading it.

The three functions in detail (edX website, 2014b):

- *grade_fn*: For grading student's work. This is a JavaScript function that returns the students answer. It is not grading it!
- *get_state*: For getting the current state of student's work and saving it. It returns the current state of the simulation. The difference to *grade_fn*, that the state can be more complicated than the students answer and therefore returning more values can be required. For example, if one would ask in a simulation just for the overall charge, the answer would be just a number. But for reloading the simulation after a brake also the positions of the dummy charges are needed. In this case *grade_fn* would return the number and *get_state* the positions of the dummy charges.
- *set_state*: For reloading student's previous work. This function takes input values for reloading a previous state.

These requirements lead to the clear separation of edX and the JavaScript simulation. Furthermore, it is appropriate for this work where the edX course and the physics simulations are written by different people. It means that the developer of the simulations just has to implement these three functions and does not need to worry about grading, evaluating student's identification, state saving and so on. This is all done by edX. On the other hand, the developer of the edX course can be sure that these three functions exist in the HTML file. He can use them without knowing how

5. Design and development of the Interface between TEAL Simulations and edX

```
1 <?xml version="1.0"?>
2 <problem>
3   <script type="loncapa/python">
4     import json
5     def grade_answer(expect, ans):
6       answer_and_state = json.loads(ans)
7       answer = json.loads(answer_and_state["answer"])
8       x = int(answer['x'])
9       if x == expect:
10        return{'ok': True, 'msg': 'Good_Job!'}
11      else:
12        return{'ok': False, 'msg': 'Try_again.'}
13   </script>
14   <customresponse cfn="grade_answer" expect="3">
15     <jsinput
16       gradefn="gradefn"
17       set_statefn="setstate"
18       get_statefn="getstate"
19       width="818"
20       height="700"
21       html_file="/static/simulation.html?size=400"/>
22   </customresponse>
23 </problem>
```

Listing 5.2: A jsinput sample problem, based on (edX website, 2014b).

the simulation is created, how the students answer is prepared or anything else. He gets the answer and can evaluate it using known edX XML and python code.

An example code for a minimal *jsinput* problem set can be seen in listing 5.2. The parameters *jsinput* can take are the three functions described above as well as (edX website, 2014b):

- *height* and *width*: Two integer values that set the size for area of included HTML page edX course.
- *html_file*: The path to the embedded HTML file

The parameters and their properties can be seen in table 5.1.

5.2.3. Grading using python

The previous sections describe how to send given answers from the HTML host file to the edX XML course code. Python is used for the evaluation of

5.3. Starting a simulation with initial values

Attribute Name	Value Type	Required	Default
html_file	Url string	Yes	None
gradefn	Function name	Yes	gradefn
set_statefn	Function name	Yes	None
get_statefn	Function name	No	None
height	Integer	No	500
width	Integer	No	400

Table 5.1.: Attributes for *jsinput* (edX website, 2014b).

the answers. The code example is the same as for *jsinput* 5.2. The defined grade function is called *grade_answer* and takes the two parameters *expect* and *ans*.

The structure of the parameter *ans* depends on the use of *jsinput*. **If *get.state* and *set.state* are not set**, *ans* contains only the given answer as a string. **If *get.state* and *set.state* are set** the answer is a json string containing the answer **and** the state!

For the example code 5.2 that means, in line 6 the value of the parameter *ans* is loaded in *answer_and_state* which contains now as the name says the state **and** the answer. Only the answer string is needed for this example. Hence it is stored in a separate variable called *answer* as seen in line 7. This variable now holds the given answer as a list of values. In line 8 a specific value called *x* is taken which will then be compared to the expected variable. Finally *True* or *False* is returned depending on the given answer.

5.3. Starting a simulation with initial values

The previous section 5.2 describes how to transfer data from the HTML host file to the edX XML file. In this section the other way around is explained. The requirement was to create random variables for the positions of the hidden charges in the edX XML file and send them to the HTML host file. The reason why the values have to be created in the edX XML file is that edX offers some advantages which should be utilized. One is the capability of choosing if edX should create a new random variable on each attempt a student makes or just once for every student and then this variable will on every attempt be the same. That is important for this work as just one randomly created set of positions for the hidden charges per student should be used. This gives the student the possibility of checking if the given

5. Design and development of the Interface between TEAL Simulations and edX

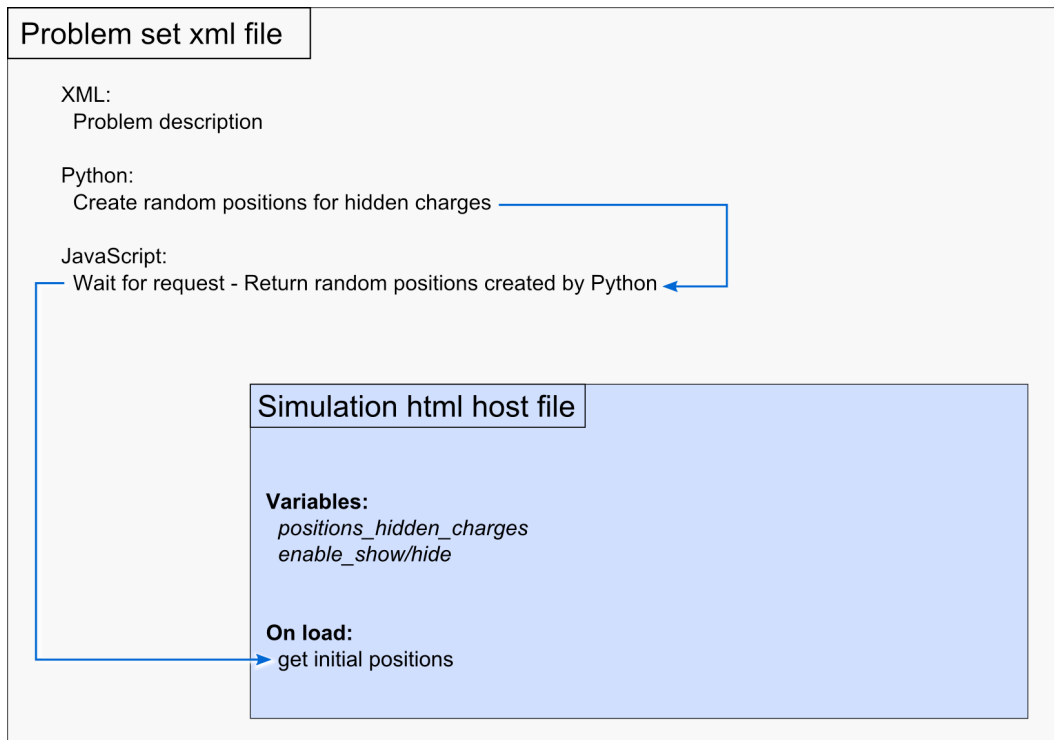


Figure 5.4.: Structure of a course page that embeds a HTML file using postmessages for sending the values of the initial positions of the hidden charges to the HTML host file.

answer is correct or on a fail retry to find it. The other reason was to give the course creator the possibility to influence on the positions of the hidden charges and hence, difficulty of the assignment. For the extended simulation that means the course creator can limit the range of the randomly created positions. This is important as hidden charges near to the box boundaries are easier to find than centered charges.

Again the initial example 5.1 is extended with the new methods. The resulting structure and data stream can be found in figure 5.4. As in the example for grading the students answer, the structure shows only the simulation's HTML host file, hiding the simulation itself for better understanding and as it is not involved in the communication.

5.3. Starting a simulation with initial values

```
1
2 <?xml version="1.0"?>
3 <problem display_name="Point_Charges_Extended" rerandomize="
  per_student" attempts="1000">
4
5   <script type="loncapa/python">
6     hidden_charges = numpy.zeros((2,3))
7
8     hidden_charges[0][0] = random.randint(-5,5)
9     hidden_charges[0][1] = random.randint(-5,5)
10    hidden_charges[0][2] = random.choice[-1,1]
11
12    hidden_charges[1][0] = random.randint(-5,5)
13    hidden_charges[1][1] = random.randint(-5,5)
14    hidden_charges[1][2] = random.choice[-1,1]
15  </script>
16
17  ...
```

Listing 5.3: Creating randomly positions for two hidden charges using python.

5.3.1. Creating the initial values with python

The positions of the hidden charges are created by the python function in the edX course XML file. In the final simulations this positions should be randomly created. For this prototype simulation static positions are created as it makes testing easier. A python code example for randomly created values is shown in listing 5.3. Note the first line. Here edX is told to create a random variable just once for every student. The example code creates the positions and charges for two point charges, where the possible position is between -5 and 5 and the charge is either -1 or 1.

5.3.2. Iframes and postmessages

The JavaScript postmessage method is used to transfer the created positions from the edX course XML file to the simulations HTML host file. By using the *jsinput* function, the HTML host file will be embedded within an iframe HTML tag. For security reasons a direct access to JavaScript variables in the host file is not allowed as the embedded site can also be on an unknown server. Hence messages are sent for communication. A benefit of this circumstance is that the extended simulations can be stored on an external server too.

5. Design and development of the Interface between TEAL Simulations and edX

```
1 element.addEventListener(event\_type , message\_event);
```

Listing 5.4: An example for a JavaScript event listener.

The `postmessage` method always needs a sender and a receiver. The receiver listens for a message which causes a so called message event.

The communication between the edX XML file and the HTML host file is as followed. Important is the exact chronology!

1. The edX XML page is called.
2. The initial values for the positions of the hidden charges are created by the XML file.
3. An eventlistener is started by the XML file.
4. The HTML host page is embedded by the XML file.
5. The HTML host page also starts an eventlistener.
6. The HTML host file sends a message to the XML file requesting the initial values.
7. The eventlistener of the XML page receives the message and calls the message-event function.
8. The message-event function of the XML file sends a message with the initial values back to the HTML host file.

The JavaScript code of the edX XML file can be seen in listing 5.5. First the message event function is created and below in line 8 the event listener is implemented. The syntax of the eventlistener is as shown in listing 5.4:

The line begins with *element* which defines the element where the listener is added to, in this case the current window. *Event_type* is a string describing the type of the sent event which could be a click event for buttons or as in this case a message. *Message_event* is the function that will be called when a message is received, for this example the defined *initResponse* function.

The *initResponse* function is also just a message sender which gets a single parameter *e* that is used for sending a message back to the sender of the initial message. The message itself is a single string containing all values of the initial positions separated by a colon. The JavaScript code of the HTML host file is similar to the XML file and can be seen in 5.6. The sender in line 14 just sends a message to everyone who listens, which is defined by `""`.

The message-event is again a simple function which takes one parameter including the received message. As described above it is a string containing all values for the positions of the hidden charges. The function takes this

5.3. Starting a simulation with initial values

```
1
2 <script type="text/javascript">
3   initRespond = function(e)
4   {
5     e.source.postMessage( $hidden_charges[0][0] + ':' + $
6       hidden_charges[0][1] + ... , e.origin );
7   }
8   window.addEventListener( 'message' , initRespond );
9 </script>
```

Listing 5.5: JavaScript code for postmessage communication of the edX XML file

```
1
2 handleResponse = function(e)
3 {
4   positions_hidden_charges[0][0] = e.data.split(':')[0];
5   positions_hidden_charges[0][1] = e.data.split(':')[1];
6   positions_hidden_charges[0][2] = e.data.split(':')[2];
7
8   positions_hidden_charges[1][0] = e.data.split(':')[3];
9   positions_hidden_charges[1][1] = e.data.split(':')[4];
10  positions_hidden_charges[1][2] = e.data.split(':')[5];
11 }
12
13 window.addEventListener( 'message' , handleResponse );
14 parent.postMessage( 'message' , "*" );
```

Listing 5.6: JavaScript code for postmessage communication of the HTML host file

5. Design and development of the Interface between TEAL Simulations and edX

string, splits it at every colon and stores the values in the local variable *positions_hidden_charges*.

5.4. Development of a test course using *jsinput*

The previous sections 5.2 and 5.3 describe the communication between the edX XML file and the HTML host file of the simulation. Each section describes one direction of the communication. In this section both of them are merged and finally the actual simulation is embedded. Therefore the structure of the example shown in image 5.1 is extended again to the final test course which is shown in image 5.5.

The shown structure consists of the previous described communication between the edX XML file and the simulation's HTML host file. Additionally the simulation which is written in Java is embedded and its communication with the HTML host file is described.

When a student clicks the simulations button *Start Simulation*, the simulation loads the initial values of the positions for the hidden charges and places them. Furthermore it enables or disables the *Show/Hide* Button depending on the value of *enable_show/hide*.

When a student clicks the *Restart Simulation* button, the same procedure as with *Start Simulation* is processed. Additionally dummy charges are placed depending on the values of *positions_dummy_charges*.

The last job the simulation has to do is to update the *positions_dummy_charges* on every placement or movement of a dummy charge by the student. This ensures that at every moment during the running of the simulation the current positions of all dummy charges are saved in the HTML host file and hence *jsinput* can read them for grading the simulation whenever the *Check* button is clicked.

5.5. Loading, saving and sending values

Values of variables are used within different programming languages and different files in this work. This section provides detailed information on this topic and describe the different situations which came up during the work on this thesis.

5.5. Loading, saving and sending values

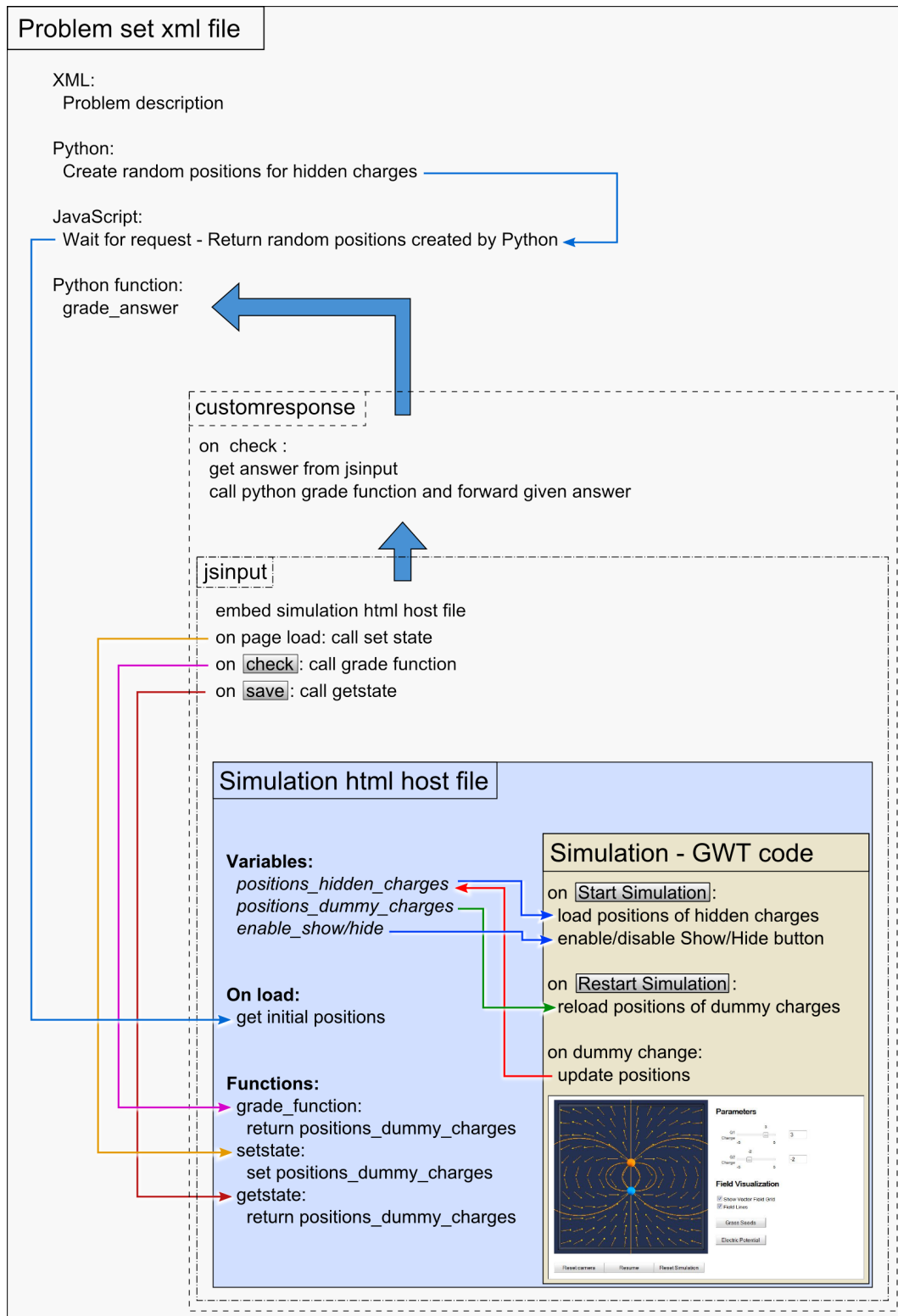


Figure 5.5.: Structure of the final test course page that embeds a HTML file containing a GWT created simulation.

5. Design and development of the Interface between TEAL Simulations and edX

```
1 <problem display_name="Point_Charges_Extended">
2   <script type="loncapa/python">
3     python_variable = 4
4   </script>
5
6   <script type="text/javascript">
7     console.info('The_value_of_the_python_variable_is_',
8       $python_variable);
9   </script>
10 </problem>
```

Listing 5.7: Example for reading a Python variable with JavaScript

5.5.1. One variable in one file used by two programming languages

As described in section 5.3 for the initial positions of the hidden charges random variables were created within the edX course XML file using python. To start the simulation, the values of this variables are sent with JavaScript to the HTML host file. The question was: How does JavaScript get the value of the Python variable? As long as this happens within one XML file the solution is very easy. JavaScript can directly access python variables within the same file. The syntax of the code can be seen in the code example 5.7.

In this example the variable *python_variable* is created using python. Then it is printed to the browsers console using JavaScript. Therefore JavaScript can access the Python variable by using its name with a leading "\$". That also works for saving values with JavaScript to a Python variable.

5.5.2. One variable in two files used by the same programming language

This problem is discussed in detailed in section 5.3 so it is mentioned here just for completion. To transfer the initial values from the XML course file to the HTML host file the variables of these two files can not be accessed directly from each other, neither can they be displayed via iframe in the same browser window. This is a security mechanism of JavaScript and hence very important. The solution was to use postmessages for sending the values between the two files. For further information please see the section 5.3.

5.5. Loading, saving and sending values

```
1 public native String getValue(int i, int j)
2 /*-{
3     return $wnd.e_initValues[i][j];
4 }-*/;
```

Listing 5.8: Example for reading a JavaScript variable with GWT (Adam et al., 2013).

```
1 public native void updateDummyPos(int id, int x)
2 /*-{
3     $wnd.DummyPos[id] = x;
4 }-*/;
```

Listing 5.9: Example for writing a JavaScript variable with GWT (Adam et al., 2013).

5.5.3. One variable in two files used by two programming languages

As the values for the hidden charges are stored directly in the HTML host file they have to be loaded by GWT into the simulation. Therefore GWT offers the *JavaScript Native Interface* (JSNI). It gives programmers the possibility to write JavaScript code within the uncompiled Java Code. An example for reading a JavaScript variable within Java can be seen in listing 5.8. An example for writing JavaScript variables within GWT can be seen in listing 5.9.

At this point a very useful function should be introduced that can be used for writing text in the browser's JavaScript console. The code can be seen in listing 5.10.

There are a few rules every JSNI method has to fulfill Adam et al. (2013):

- The function has to be declared as native.
- The function has to have an empty body. Furthermore it has to end with a semicolon.
- The JavaScript code has to be written within `/*-\{` and `\}-*/`.

```
1 public static native void log( String msg )
2 /*-{
3     console.log( msg );
4 }-*/;
```

Listing 5.10: Example for using a Python variable with JavaScript (Adam et al., 2013).

5. Design and development of the Interface between TEAL Simulations and edX

For an example see listing 5.10.

5.5.4. Reloading a previous state - The way from XML to JavaScript to Java

As described, `jsinput` offers a `set_state` function for reloading the previous stored state of the simulation. But for the special case of this work where the simulations are written in Java there is a fact that needs further attention: The formatting of strings by different programming languages.

In many tests of the interface the problem came up where communication did not work although the code seemed to be right. As mentioned earlier, debugging becomes complicated when so many different languages are used to write the simulations and course pages. It took quite long to figure out what went wrong. The problem was that `jsinput` returned and forwarded the answers and states always as a long string.

For example if the position of the following two point charges and their charge are given:

- ChargeId 1; X: 5; Y: 2; Charge: 1
- ChargeId 2; X:-3; Y: 3; Charge: -1

This values as a string could look like:

```
1,5,2,1,2,-3,3,-1
```

Unfortunately they can also look like this:

```
1,5,2,1,2,- 3,3,- 1
```

In the last string, there is a space between every “-” and the number following!

Furthermore, in different programming languages a string ends with a different indicator. The most common one is the “\n”.

Sending strings or variables through different languages can be a problem. Hence, the data stream of the example simulations is shown in the structure figure 5.6. It shows how the simulation’s state is saved and reloaded again. As seen in the figure, edX demands from `get_state` one single string. And it

5.5. Loading, saving and sending values

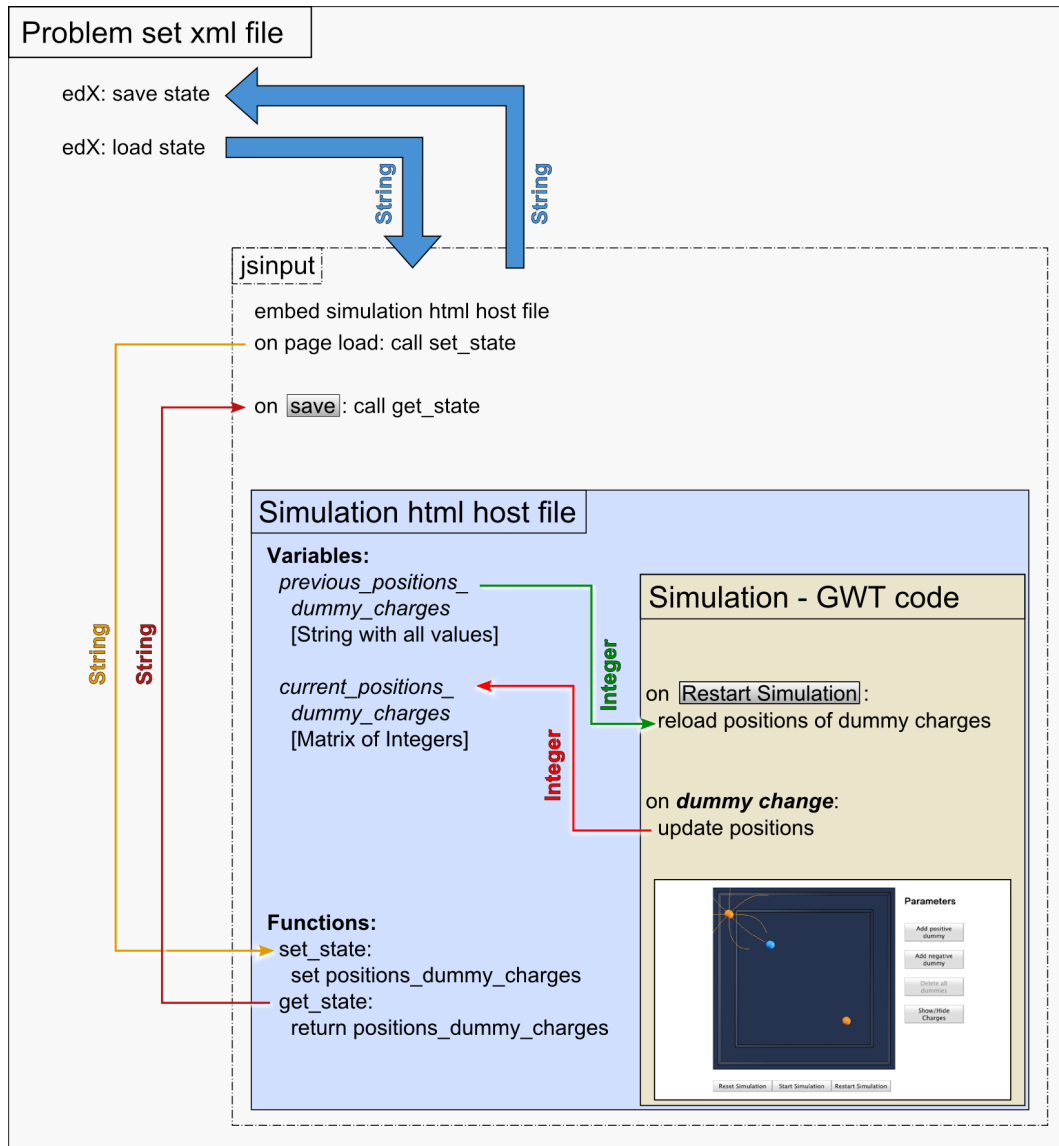


Figure 5.6.: Data stream of saving and reloading the simulations state and the used data types.

5. Design and development of the Interface between TEAL Simulations and edX

delivers *set_state* the same single string back. That means, the HTML host file also sends and saves the values as string.

At this point a decision had to be made how to handle this strings and where they should be converted from and to integers. Both is done using JavaScript. Once direct in JavaScript and once using JSNI in the Java Code. To prevent problems with reloading and overriding the dummy positions they are stored twice in the HTML host file. First for the reloaded positions from a previous state. This positions are stored through the whole simulation till the next time the student stores the new state. So the student can reload the previous state as often as he wants. Second the current dummy positions are stored and updated whenever they are moved by the student.

String to Integer

The first case was the converting of a string containing all values to integers. This had to be done on reloading a previous state of the simulation. The edX course XML file sends this string using *jsinput* to the HTML host file. This string is stored unchanged in the HTML host file. On loading the dummy positions the string is split into substrings where every substring contains one value. The code for this can be seen in 5.11.

Basically the string stored in the HTML host file contains the values separated by ",". But as described above there can be other characters like spaces or something else written in the string. Due to this, they needed to be cleaned up before conversion. This is done in line 4 in the sample code where all spaces and characters which should not be in there are deleted. After that the string is split in substrings and the *i*-th element is taken.

This is done within the Java Code using JSNI for writing JavaScript code. The function in 5.11 is a Java function using JavaScript code as described earlier in this section. In this function the string is loaded from the HTML host file. After that it is cleaned up, split into substrings and the *i*-th element is taken. This element is converted to an integer and returned to Java.

Integer to String

The second case is to convert the integers to a single string. That has to be done twice. Once within *set_state* and once within *gradefn*. Both return a single string and particularly they send the same string.

5.5. Loading, saving and sending values

```
1 public native int getDummyValue(int i)
2 /*-{
3     var myString = $wnd.statestr;
4     myString = myString.replace(/[[\]]/g, '');
5     var myValue = myString.split(',')[i];
6     var myInt = parseInt(myValue);
7     return myInt;
8 }-*/;
```

Listing 5.11: Example for using a Python variable with JavaScript

In the HTML host file a matrix is stored holding the dummy positions and the charge. How to save a value in the HTML host file using JNI in Java is described in the example code 5.9. When the students presses the *Check* or *Save* button the HTML host file has to deliver a single string containing all the dummy values. Therefore all the string values are taken and afterwards “stringified” with JavaScript. This can be done in two different ways. Both of them are used in this work, one for *gradefn* and one for *set_state*. The reason for this is that the used method in *gradefn* is a little more work but better to debug. Once this function was running stable the simpler method was used within *get_state*.

The code of the function that is returning the string for *gradefn* is seen in 5.12. In line 3 all values are stored in a so called *linked list*. At this point linked lists are not described in detail as this would go too far. Shortly said, for every value a name is stored additionally. For example, in the expression `d1y: e\curDummyPos[0][1]`, the element `d1y` holds the value of `e\curDummyPos[0][1]`.

In line 13 the string is printed to the browsers console for debugging. And in line 14 the linked list is “stringified” and returned.

For comparison purposes, the easier code which was used for the function that is returning the string for *get_state* is also shown. It can be seen in listing 5.13. It looks very similar to the method used for *gradefn*. The difference is that an array is used instead of a linked list. In this case only the values are stored. This leads to less code and also less data that is sent to edX. The disadvantage is, that it is more difficult to debug.

5. Design and development of the Interface between TEAL Simulations and edX

```
1 function getGrade()  
2 {  
3   givenAnswer = {d1x: e.curDummyPos[0][0],  
4                 d1y: e.curDummyPos[0][1],  
5                 d1c: e.curDummyPos[0][2],  
6  
7                 ...  
8  
9                 d10x: e.curDummyPos[9][0],  
10                d10y: e.curDummyPos[9][1],  
11                d10c: e.curDummyPos[9][2]};  
12  
13   console.info("getGrade_called . Given_answer:", JSON.  
14               stringify(givenAnswer));  
15   return JSON.stringify(givenAnswer);  
}
```

Listing 5.12: Returning one single string containing all positions of the dummy charges
- using a linked list

```
1 function getState() {  
2   givenAnswer = [e.curDummyPos[0][0],  
3                 e.curDummyPos[0][1],  
4                 e.curDummyPos[0][2],  
5  
6                 ...  
7  
8                 e.curDummyPos[9][0],  
9                 e.curDummyPos[9][1],  
10                e.curDummyPos[9][2]];  
11   console.info("GetState_called . Given_answer:", JSON.  
12               stringify(givenAnswer));  
13   return JSON.stringify(givenAnswer);  
}
```

Listing 5.13: Returning one single string containing all positions of the dummy charges
- using an array

5.6. The synchronization problem

For the first method which uses a linked list an example string would look like (shortened):

```
{"d1x":10,"d1y":-2,"d1c":1, ... "d10x":-10,"d10y":2,"d10c":-1}
```

The same example string for the second method using an array looks like (shortened):

```
[10,-2,1, ... ,-10,2,-1]
```

The methods used to obtain integers from strings differ. In the case of a linked list an element can be selected by it's name. In the case of an array the index of the element has to be used.

For example, the python code for getting the x value of the first dummy of the stringified linked list would look like:

```
dummy_1_x = int(answer['d1x'])
```

Getting the same element out of the stringified array would look like:

```
parte_answer = answer.split(',')
dummy_1_x = int(parted_answer[0])
```

5.6. The synchronization problem

At this point an important question, which came up during the work on this thesis, should be discussed: Why are the three JavaScript functions, *gradefn*, *get_state* and *set_state*, for *jsinput* implemented in the HTML host file and not directly within the GWT Java code as this code will be compiled to JavaScript code? The short answer: To ensure a stable communication. A more detailed answer is given further down in this section.

There were different approaches for the communication between edX and the GWT simulations which where less difficult as the used version. Unfortunately for every attempt a specific situations was found where they did not work correctly. This problem was called the *The Synchronization Problem*. The reason was found in the complex problem definition. Firstly the circumstance that the simulations were not pure JavaScript implementations. They were written in Java and just compiled to JavaScript. What sounds negligible emphasized as a real problem. Secondly, the strict separation of the course and the simulation code had to be kept.

5. Design and development of the Interface between TEAL Simulations and edX

GWT offers a feature called *JavaScript Native Interface* (JSNI). This offers the capability of writing JavaScript code directly in the Java code file. It was implemented to use JavaScript functions which are not ported or portable to Java, such as the window manipulation functions included with JavaScript. For this work, this feature seemed to be a real benefit as the three functions *gradefn*, *get_state* and *set_state* could be implemented directly in the Java Code using JSNI. Hence they would be part of the extended simulations. That would make a direct communication between edX and the extended simulation possible without the intermediate step in the HTML host file.

The described approach was implemented during this work and it seemed to work. Unfortunately, while testing the extended simulations specific situations were found where communication did not work. The problem was found in the optimization of GWT while compiling the Java code to JavaScript. As every bit sent over the Internet slows down an application the GWT compiler tries to keep the code as small as possible. That means on the one hand that functions which are not used can be completely excluded. On the other hand a function that is not needed at start can be loaded later to keep the start procedure as fast as possible. And this led to the malfunction of the communication because GWT could not know that the three *jsinput* functions have to be present from the beginning of the extended simulation, especially the *set_state* function.

At the time this thesis was written, for the best of our knowledge, there was no possibility to tell GWT when a function has to be ready loaded. One solution would be to implement a query if the functions are ready loaded and if not to wait for a while and try again. But this could on the one hand lead to a longer and noticeable delay for the user and on the other hand it is not good programming style.

The solution was to use the HTML host file as a interim stage between the edX course and the GWT simulation where the necessary values are always present and usable for both of them.

5.7. Summary

In this section the communication between the edX XML course file and the extended simulation created with GWT is described. For the beginning, the basis, an edX course page including the original point charges simulation,

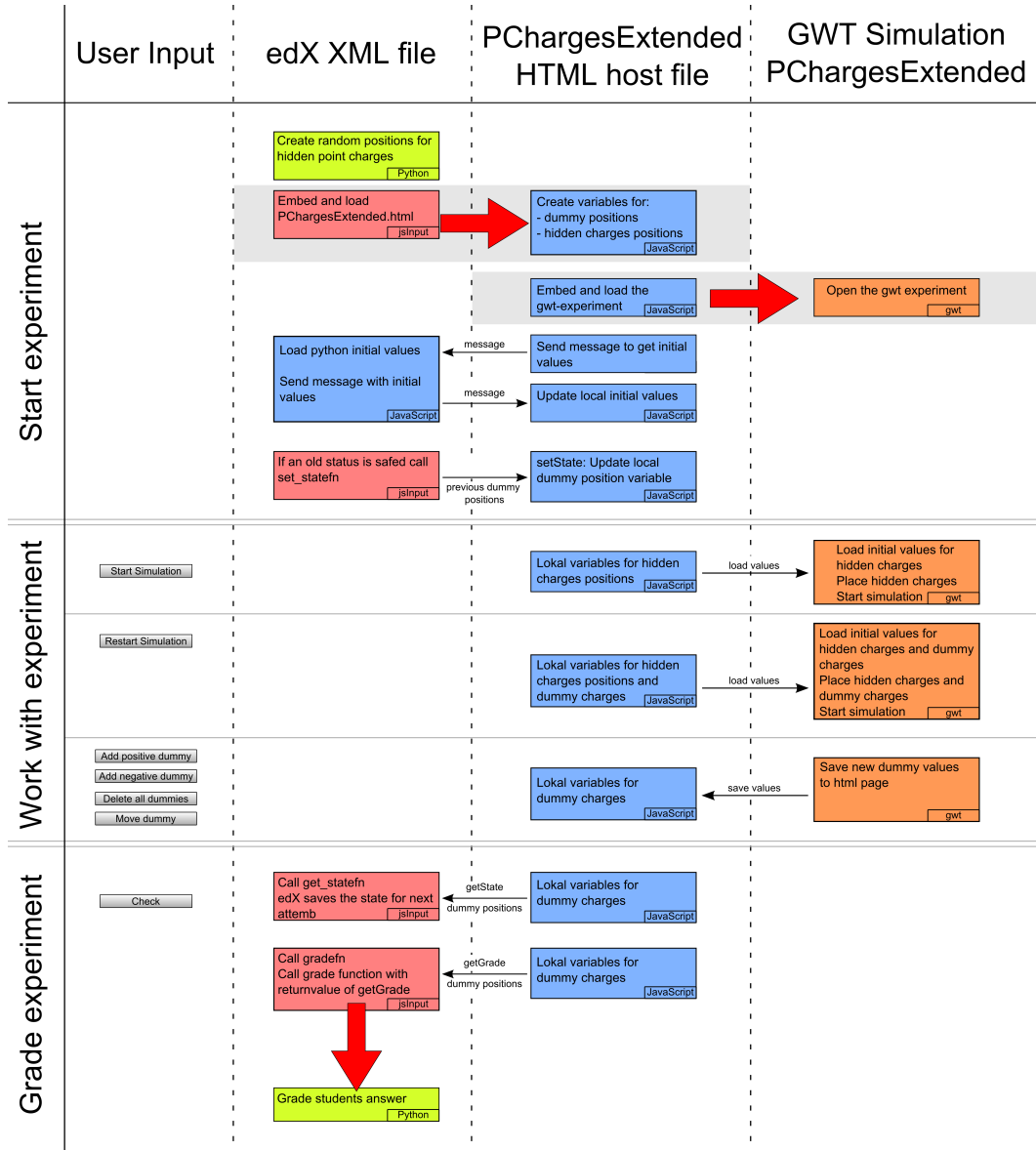


Figure 5.7.: Flowchart of the complete communication between edX and our extended simulation.

5. Design and development of the Interface between TEAL Simulations and edX

is presented. Afterwards this basis is extended step by step until the final interface which provides a stable communication between edX and a GWT compiled simulation.

For communicating between edX and a JavaScript application the edX function *jsinput* is used. This function has to be embedded within a *customresponse* tag, which is also an edX function. For *jsinput*, three JavaScript functions have to be implemented in the HTML host page of the JavaScript application, *gradefn*, *get_state* and *set_state*. The first attempt during this work was to implement these three functions direct in the Java code using GWT's JSNI interface. Using this interface JavaScript code can be implemented directly in the Java code.

This approach seemed to work, but communication was not stable. There were situations where *jsinput* did not work correctly. The problem was found in GWT's optimization of the Java code while compiling it to JavaScript. GWT did not know that the three functions have to be loaded right after starting the simulation. Hence, it could happen that *jsinput* was calling one of the three functions but they were not loaded at that time.

To overcome this problem, the described solution using the HTML host file as an interim stage is used. The three *jsinput* functions are written directly within the HTML host page and not within the Java code of the simulations. This solution is of course not as clean as a programmer would hope it could be, but it is running stable.

At this point another overview of the interface should be given but with the focus on the chronology as this is very important to guarantee a stable communication. Therefore a flowchart is shown in figure 5.7. It illustrates which function and which variable have to be available and updated at what point in time. The reason why it is so important to strictly keep this chronology is that there are very few possibilities included in JavaScript for querying if a function or variable exists at runtime. Furthermore, no methods are offered for dealing with the state if a function is not loaded when calling.

During the work on this thesis *jsinput* was still in beta stage. Moreover *jsinput* was constantly improved and enhanced and for the future there may be better and easier solutions for the interface.

6. The extended TEAL simulations

This section describes the three TEAL simulations which were extended during this work. For every simulation the requirements are outlined followed by the implementation and details of the finished simulation.

The goal of this master thesis was to create an interface between edX and the simulations and furthermore to extend one or more TEAL simulations for answering questions using it. During the work on this thesis three of them were extended. The three simulations in chronological order:

- Point Charges Extended - Section 6.1
- Gausses Law Extended - Section 6.2
- Amperes Law Extended - Section 6.3

The order was based on the order of their occurrence in the edX course 8.02x.

After a working example simulation was created, as described in the previous chapters, a concept for the first extended simulation was prepared. It contained all changes of the original simulations for asking educationally meaningful questions which can be answered by using the new extended simulation. This had to be done for every simulation on it's own as they are different.

What all extended simulations should have in common is the fact that they should be as simple as possible. That meant, the course creator should get as many possibilities and latitude as possible. Hence, he can use them in different ways for different questions and with different levels of complexity. One idea was to use various problem sets to explain the physical laws behind the simulation where every set could be used to describe specific effects for different situations. Another idea was to increase difficulty with every problem set to slowly explain problems.

6. The extended TEAL simulations

6.1. The first extended TEAL simulation: Point Charges Extended

This simulation was already extended for the prototype and for the tests as described in the previous chapters. Hence, at this point a basically working simulation did exist which was able to communicate with an edX course. Now a real problem set had to be developed for a meaningful simulation.

6.1.1. Requirements and Design

When the student starts a simulation a number of hidden charges should be placed in the inner box. A visible positive charge is placed between the inner and outer box which can be moved around the inner box. This charge has visible field-lines with finite length. By moving the visible charge the student should find the hidden charges. Therefore the student can place dummy charges with no effect on the field-lines to mark the places of the hidden charges. When the student checks her answer the positions of the placed dummies is sent to the XML course file where the answer is graded.

The number of hidden charges should be determined by the course creator within the XML course code using python. The maximum number of hidden charges should be five as this seemed to be the highest number of charges that fits in the simulation space.

The places for the hidden charges should also be determined by the course creator within the XML course file so he can define the complexity level. Again python should be used as this would give the course creator the possibility of random created places. With python edX can set every random number to *one per student* or *one per attempt* as described in chapter 5.3.

The dummies should be only placeable on a grid as otherwise it would be too hard to find the hidden charges. The size of the grid has to be adjusted while testings.

The grading should be done by python in the XML file to utilize all benefits of edX like statistics, overall course marks and others as described in chapter 5.2.

A Show/Hide Button should be implemented which shows and hides the hidden charges. The course creator should get the possibility of turning

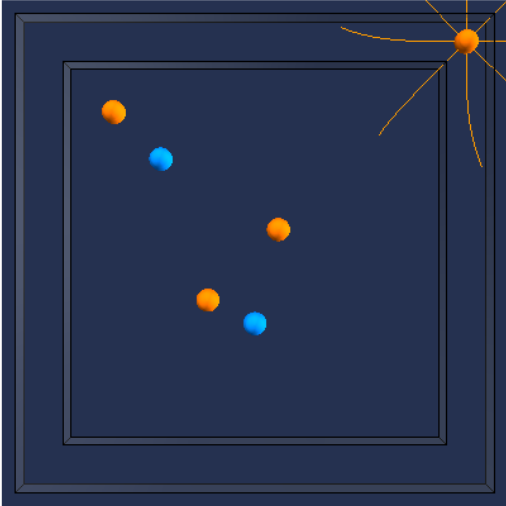
this button on or off for every problem set so she can create learning sets including it and exam sets excluding it.

6.1.2. The extended simulation

Figure 6.1 shows the final *Point Charges Extended - Simulation*. It displays the started simulation with the movable charge on the right top. Within the inner box the five hidden charges are displayed. They can be turned visible by the Show/Hide button next to the simulation on the right.

POINT CHARGES EXTENDED

There are up to 5 hidden charges. Find the position of these charges. You can add up to five positive and five negative charges.



Parameters

Add positive dummy

Add negative dummy

Delete all dummies

Show/Hide Charges

Reset Simulation

Start Simulation

Restart Simulation

Check

Save

Show Answer(s)

You have used 0 of 1000 submissions

Figure 6.1.: The final *Point Charges Extended - Simulation*

The simulation in figure 6.1 shows the test set for the hidden charges. The idea was to place the maximum number of hidden charges at different difficult places. Therefore charges were placed near the border, near the center as well as near to each other and more separate.

6. The extended TEAL simulations

The buttons in detail:

- **Start Simulation:** Starts the simulation. Loads and places the hidden charges and enables the 4 dummy buttons.
- **Restart Simulation:** Loads and places the hidden charges, reloads previous set dummy charges and enables the 4 dummy buttons.
- **Reset Simulation:** Resets the view of the simulation as it can be rotated.
- **Add positive dummy:** Places a positive dummy which can be used for marking a hidden charge. This button is only enabled till 5 positive dummy charges are placed.
- **Add negative dummy:** Places a negative dummy which can be used for marking a hidden charge. This button is only enabled till 5 negative dummy charges are placed.
- **Delete all dummies:** Deletes all dummies from the simulation. This button is disabled if no dummies are placed.
- **Show/Hide Charges:** It shows and hides the hidden charges. This button can be disabled by the course creator for exam problem-sets.
- **Check:** With this button the student's answer is checked and graded. The places of the dummy charges are sent to the edX XML course file and graded there by python. Afterwards a notification shows the student how many charges he found correctly. Furthermore, the student's answer is stored and can be reloaded whenever he wants to. This button is only enabled till the *Due Date* of the problem set.
- **Save:** Saves the student's answer. It saves the positions of the dummy charges. The stored state can be reloaded whenever the student wants to. This button is always enabled also after the *Due Date*.
- **Show Answers:** This button is not used for this simulation. It is embedded by edX.

6.1.3. Different states of the running simulation

The upper part of figure 6.2 shows the simulation after loading the course page but with the simulation not started. At this point the hidden charges are not placed. The field-lines are straight without bending. The lower part of figure 6.2 shows the running simulation with the test set of the hidden charges as described above. The field-lines are bended through the influence of the placed hidden charges.



Figure 6.2.: Top: The simulation after page was loaded
Bottom: The simulation when started

6. The extended TEAL simulations

Figure 6.3 shows the test set for the hidden charges once with hidden and once with visible charges. In the top simulation the dummy charges are placed on the left and right side of the simulation. This are the positions where the dummies are placed when added.



Figure 6.3.: Top: Running simulation with hidden charges and placed dummies on the sides.

Bottom: Running simulation with hidden charged shown. This was our test set for the hidden charges.

Figure 6.4 shows a position of a hidden charge that is very hard to find as it is near the center and the field-lines are not touching it. Important: The horizontal field line is not directly pointing to the center of the positive hidden charge. This is because of the influence of the other charges. On an exam course with Show/Hide button disabled this charge would be very difficult to find. But it also illustrates very well the influence of all charges on the field-lines.

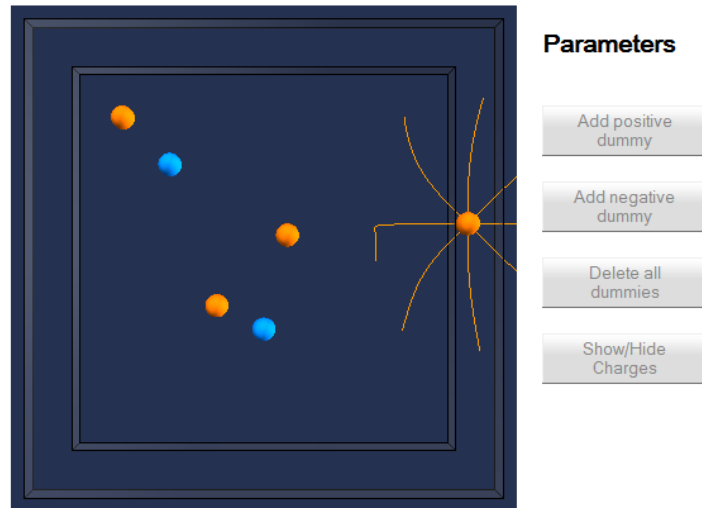


Figure 6.4.: Example for a hard to find position of a hidden charge.

6.1.4. The prototype of Point Charges Extended

At the end of this section about the first extended simulation a prior version is presented to illustrate which problems came up. The prototype is shown in figure 6.5.

The idea was the same: Finding hidden charges with one moving charge. But the area where the visible charge could be moved was the upper right quarter as seen in figure 6.5.

It is easy to see that with this setup, hidden charges placed at the positions of the white marker charges in the simulation can not be found by moving the visible charge in the small box only. Furthermore, hidden charges in the whole lower left quarter would be nearly impossible to find.

6. The extended TEAL simulations

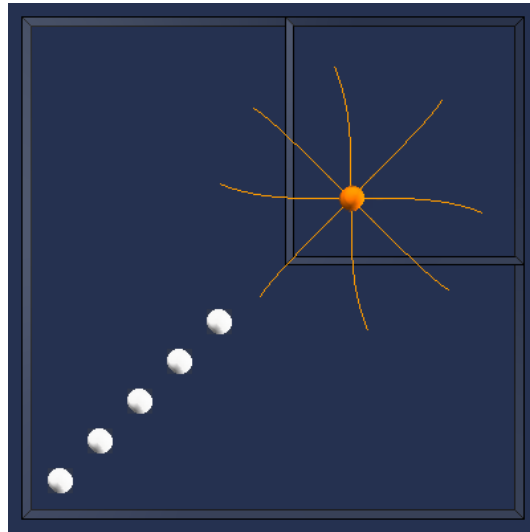


Figure 6.5.: A prototype of the Point Charges Extended simulation.

6.2. The second TEAL simulation: Gausses Law Extended

The second TEAL simulation which was extended was the *Gausses Law Simulation*. It was made to illustrate the Gausses Law on a Gaussian surface next to point charges. The Gaussian surface can be either a sphere or cylinder which is moved by sliders on the right side of the simulation while the charges are moved by dragging. Yellow arrows display the local electric field on the surface.

6.2.1. Requirements and Design

When the student starts the simulation a number of hidden charges should be placed in the simulation as well as the Gaussian surface. The hidden charges should now be found by moving the Gaussian surface. In this simulation the student can place point charges. The assignment is to neutralize the hidden charges by placing the opposite charge over a hidden charge.

The yellow arrows should be by default scaled by the magnitude of the electrical field. That means, when an opposite charges is placed on a hidden charge, the field turns zero.

The next points of requirements are very similar to the requirements of the first simulation. Hence only a short review is given. For more details see chapter 6.1.1.

- When the student checks his answer, the positions and charges he placed are sent to the edX XML course file where they are graded by Python code
- The number of hidden charges should be determined by the course creator using python and should be at maximum five.
- The positions of the initial charges should be created within the python code of the XML course file.
- The position of the placed charges has to follow a grid.
- The grading should be done by python in the XML file.
- A Show/Hide Button should be implemented which shows and hides the hidden charges.

6. The extended TEAL simulations

6.2.2. The extended simulation

Figure 6.6 shows the final *Gausses Law Extended - Simulation*. It shows the running simulation with the Gaussian surface and 5 point charges which are visible *hidden charges*. It illustrates the test set for this simulation.

GAUSS LAW FLUX EXTENDED

Choose Gaussian Surface

- Gaussian Cylinder
- Gaussian Sphere

Gaussian Surface Position and Orientation

Y Position: (range: -3 to 3)

X Position: (range: -3 to 3)

Rotation Angle: (range: -180 to 180)

Choose E Field Scaling

- Make all E Arrow lengths the same
- Scale E Arrow length by (magnitude E)^{0.3}

Add/Remove dummies

Add positive dummy (orange)

Add negative dummy (blue)

Delete all dummies

Reset camera Start simulation Retart simulation Show / Hide

Check Save Show Answer(s) You have used 0 of 1000 submissions

Figure 6.6.: The final *Gausses Law Extended - Simulation*

The buttons in detail

Some of the buttons are similar to the Point Charges simulation. Their description can be found in chapter 6.1.2. The new buttons and sliders are described as followed:

- **Choose Gaussian Surface:** For switching the Gaussian surface between a cylinder and a sphere. By default the sphere is chosen.
- **Restart Simulation:** Loads and places the hidden charges, reloads previous set dummy charges and enables the 4 dummy buttons.
- **Three sliders:** They are used for moving the Gaussian surface within the simulation and rotate it, where rotating is only useful with the cylinder.
- **Choose E field Scaling:** Here the student can switch between scaled length and equal length of the yellow arrows. Sometimes the field can be very small and hence the arrows are very short. With the equal length setting the arrows are always well seen. This mode had to be adapted because with the equal length setting the arrows were always shown. So they had to be hidden when the field turns zero.
- **Reset camera:** The simulation can be rotated. This is sometimes very helpful for finding the exact place of a hidden charge.

6. The extended TEAL simulations

6.2.3. Different states of the running simulation

The upper part of figure 6.7 shows the simulation after loading the course page but with the simulation not started. At this point the hidden charges are not placed and as the field is zero the arrows are not visible. The lower part of figure 6.7 shows the running simulation. The hidden charges are placed in the test configuration as described before.

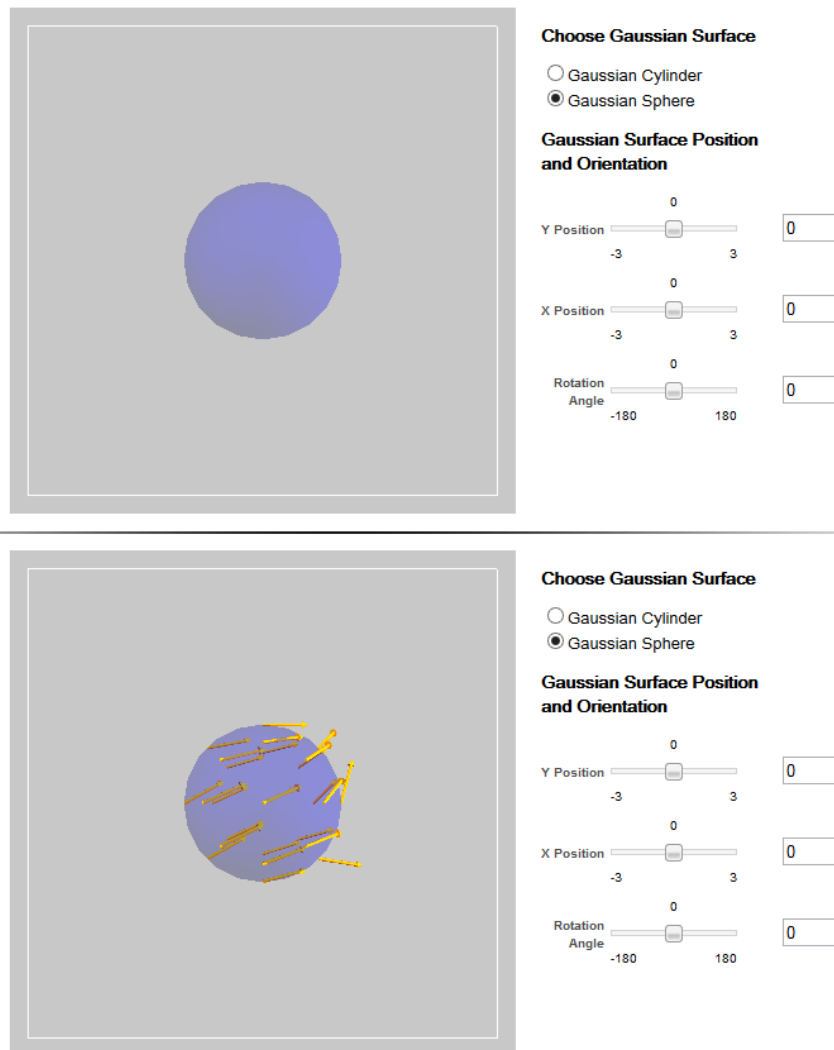


Figure 6.7.: Top: The simulation after page was loaded
Bottom: The simulation started

The upper part of figure 6.8 shows the test set for the hidden charges with visible *hidden charges*. In the lower part of figure 6.8 the charges were placed correctly over the hidden charges and the field is zero.

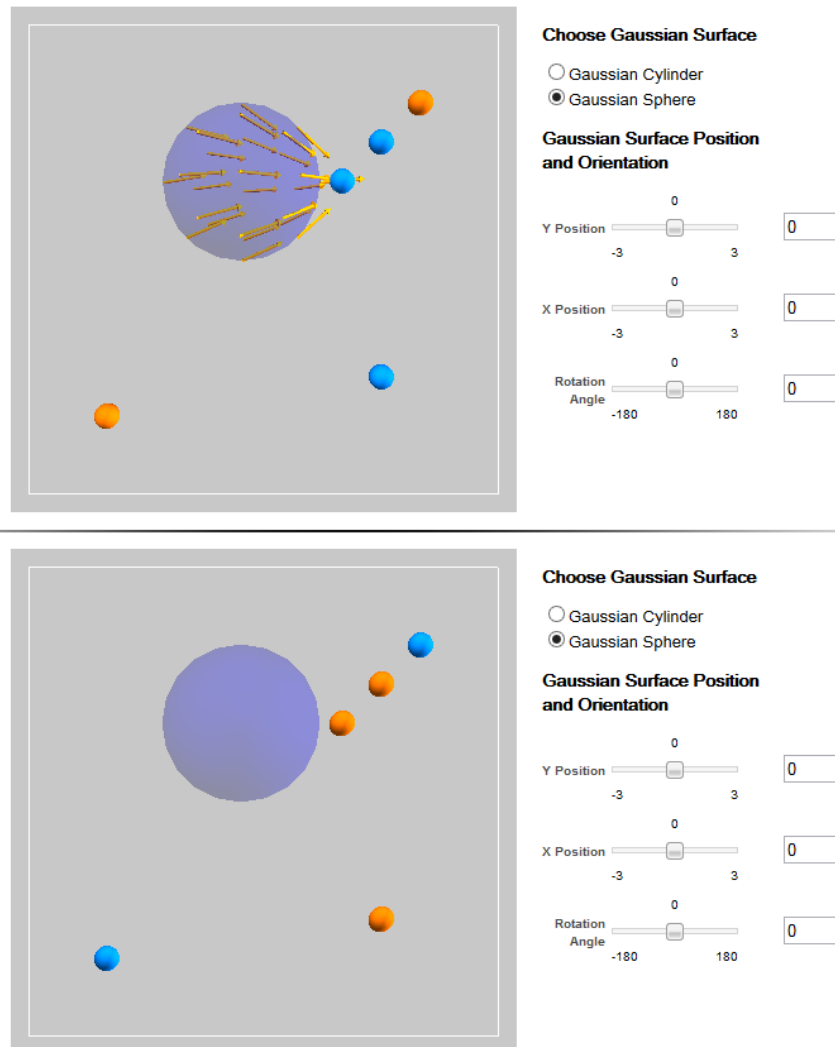


Figure 6.8.: Top: Running simulation with visible *hidden charges* in our test configuration. Bottom: Running simulation with correct placed charges over the hidden charges.

6. The extended TEAL simulations

The last figure 6.9 present from the second extended simulation shows the visible *hidden charges* as well as all possible placeable charges. It should demonstrate the difference to the first extended simulation where only dummy charges could be placed with no effect on the field. Furthermore one can see that the placeable charges are a little larger to distinguish them from the *hidden charges*.

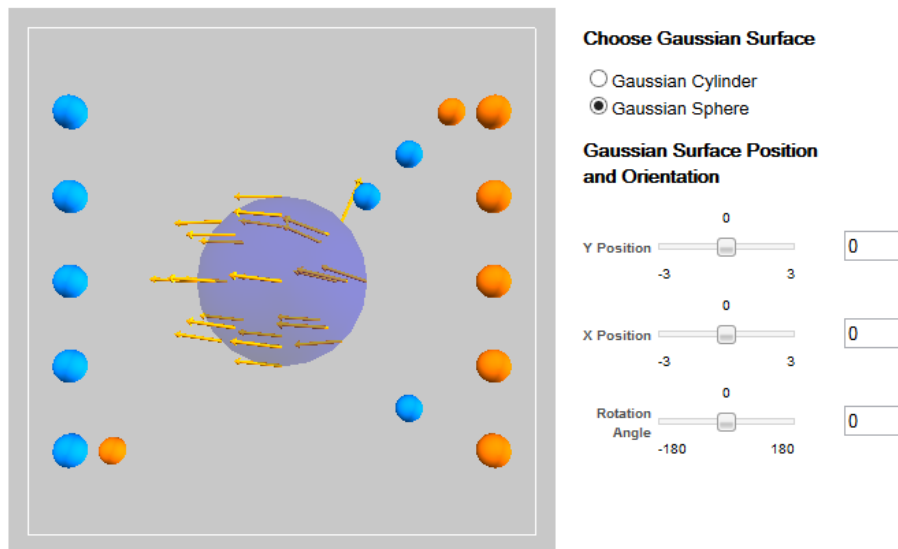


Figure 6.9.: Placed charges and their influence on the field arrows.

6.3. The third extended TEAL simulation: Amperes Law Extended

The third simulation that was extended was the Amperes Law simulation. It demonstrates the impact of line currents on an open Amperean surface which could be either a circle or rectangle. The line currents can be moved by dragging with the mouse while the Amperean surface can be moved by sliders next to the simulation.

6.3.1. Requirements and Design

When starting the simulation a number of hidden *line currents* are placed within the simulation. They should be found by moving the Amperean

surface. The system from the second simulation was used again, where the student has to eliminate all hidden line currents by placing opposite currents on top of them.

The points of requirements are very similar to the requirements of the first and second simulation. Hence only a short review is given. For more details see chapter 6.1.1 and 6.2.1.

- The blue arrows should be by default scaled by the magnitude of the field.
- When checking the answer the positions of the placed line currents should be sent to the edX XML course file where they are graded by a Python function.
- The number of hidden line currents should be set by the course creator.
- The initial positions of the hidden line currents should be set by the course creator using Python.
- The students can place the line currents only on a grid.
- Grading should be done by the edX XML File using Python.
- A Show/Hide button should be implemented.

6. The extended TEAL simulations

6.3.2. The extended simulation

Figure 6.10 shows the final *Amperes Law Extended* - Simulation. It shows the *Amperean Rectangle* as well as 5 hidden *line currents* which are enabled by the *Show/Hide* button. The image shows the test set for this simulation. The buttons have the same function as in the first and second simulation. For more details see section 6.1.2 and 6.2.2.

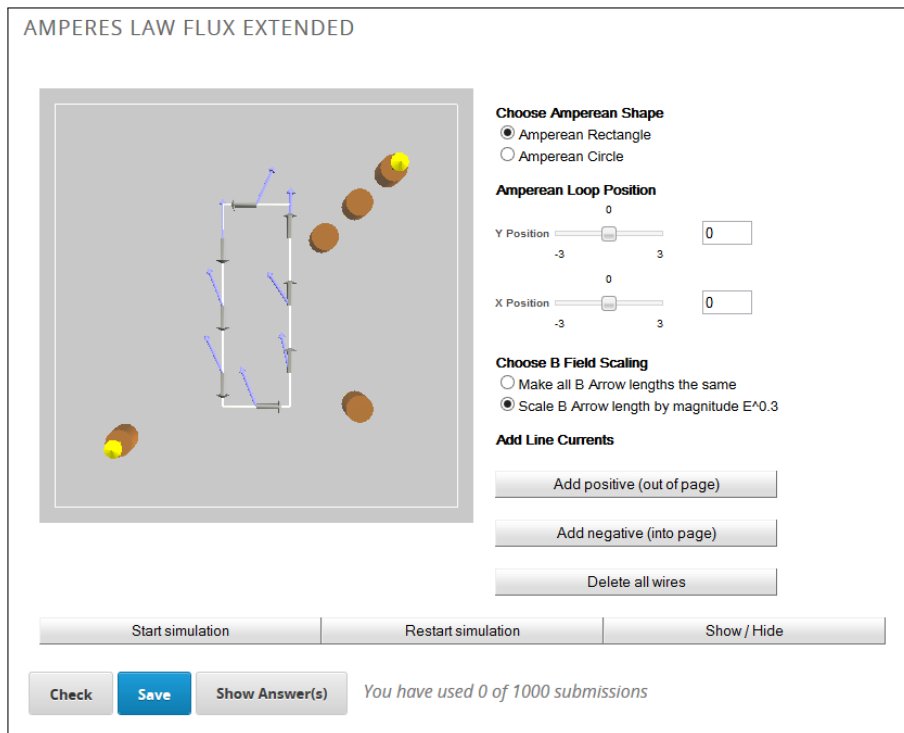


Figure 6.10.: The final *Amperes Law Extended* - Simulation

6.3.3. Different states of the running simulation

The upper part of figure 6.11 shows the simulation after loading the page. The simulation is not running. The lower part shows the simulation running with the hidden line currents at the test configuration, as described in figure 6.10.

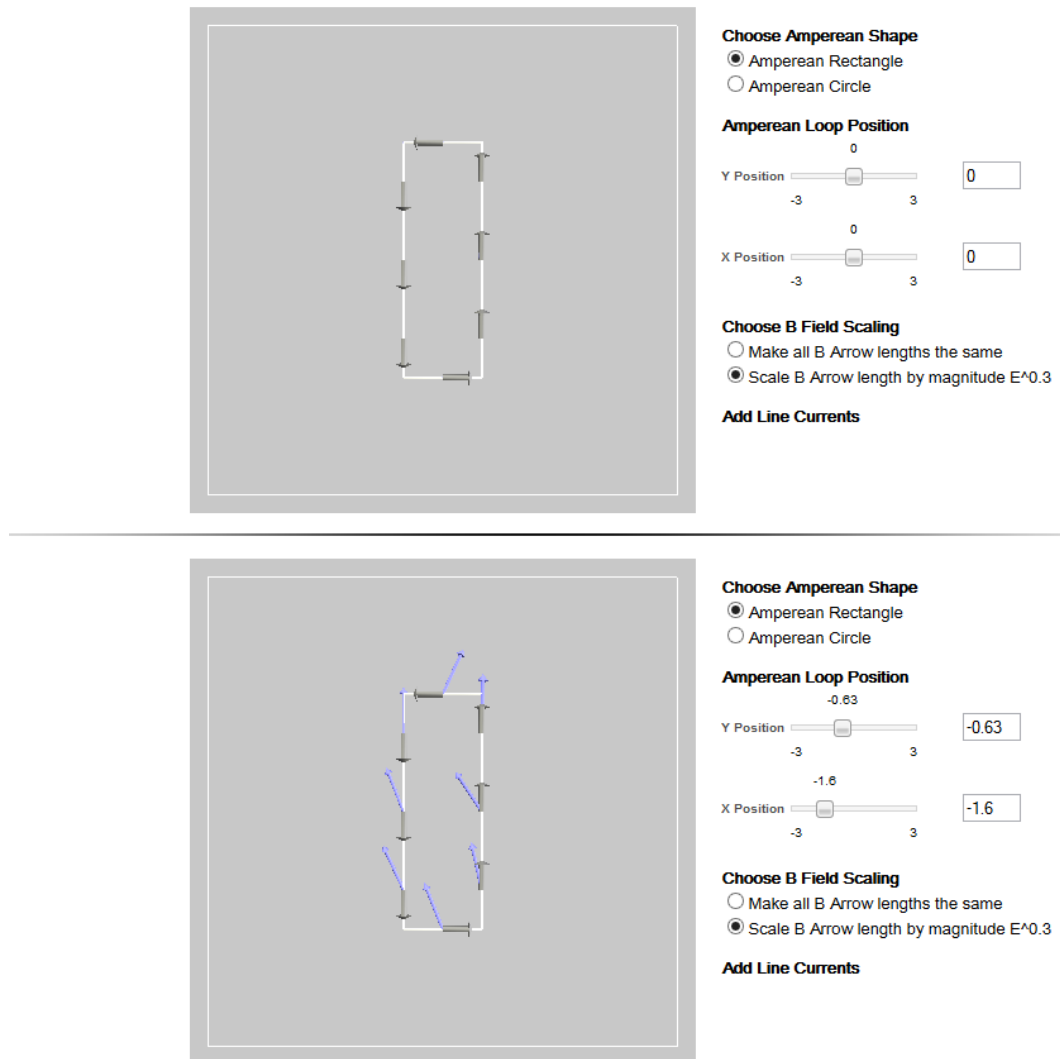


Figure 6.11.: Top: The simulation after page was loaded
Bottom: The simulation started

6. The extended TEAL simulations

Figure 6.12 shows the result when an opposite line current is placed on top of a hidden line current. The field gets zero and the blue arrows disappear.

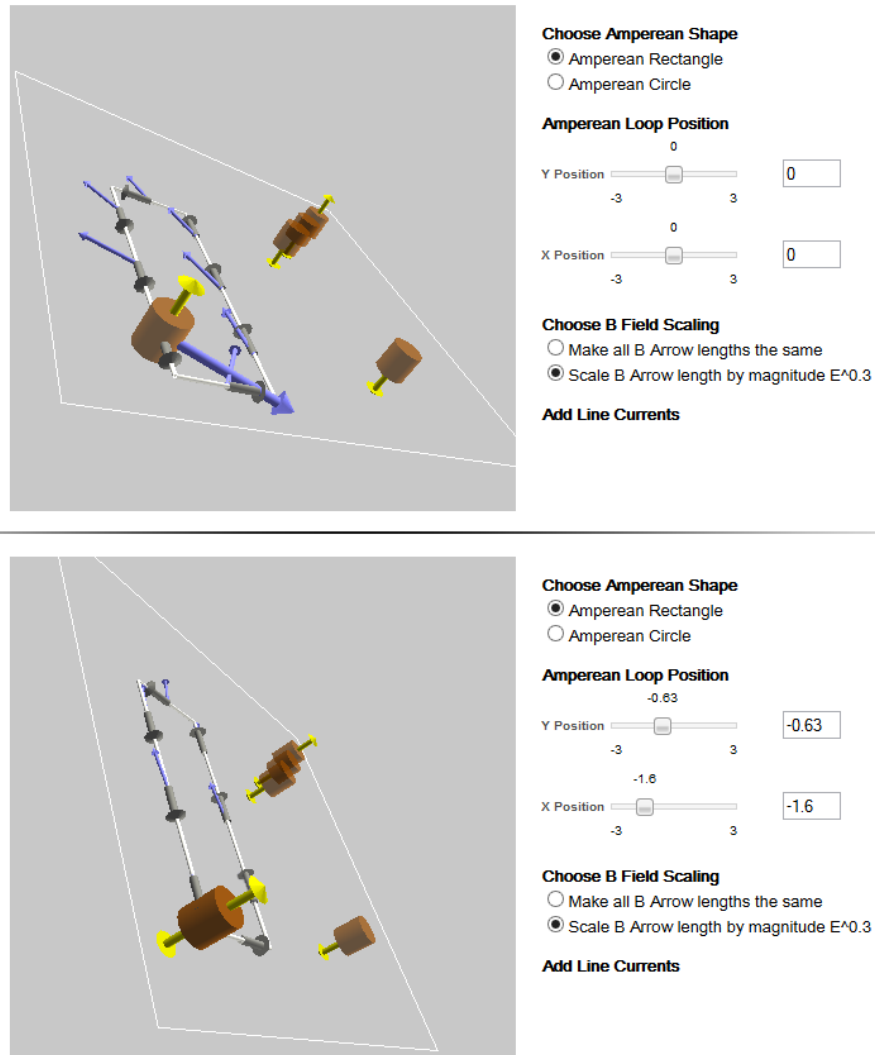


Figure 6.12.: Top: Field induced by a line current.
Bottom: An opposite line current placed on top of the first current. The field gets zero.

The last figure 6.13 for the third extended simulation shows the placed line currents and their impact on the field.

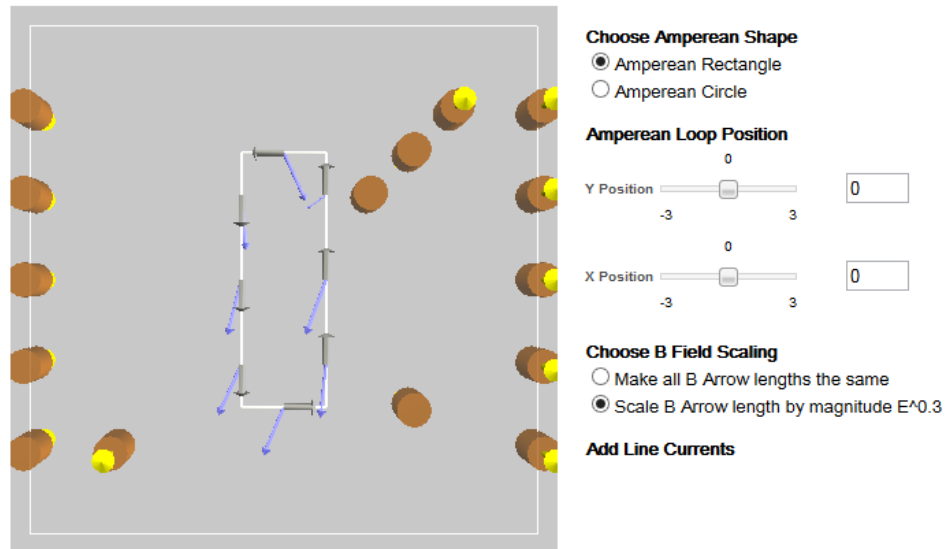


Figure 6.13.: Simulation with all line currents placed.

6.4. Findings

This section describes the two main findings that outcome during the work on this masters thesis. First, placing hidden elements which have to be found by students is a very complex and difficult process. During different tests with hidden elements it showed up that different places are always different hard to find. One the one hand that means randomly placed hidden elements can result in giving some students an advantage with easier to find elements then others. On the other hand this fact can be used for creating a number of following problem sets with increasing difficult level. Second, testers of the new extended TEAL simulations often described them as games. It seemed by extending the TEAL simulations they were accidentally gamified.

6.4.1. Placing hidden elements

Testings of different positions of the hidden charges showed that this topic needed more attention then initially thought. The recognized problem was

6. The extended TEAL simulations

described shortly in section 5.3. For example in figure 6.3 a very easy to find hidden charge is shown whereas in contrast figure 6.4 shows a very hard to find hidden charge. Before these tests the idea was to place the hidden elements randomly within the simulation maybe with some space to the outer border. Already the first test showed that this assumption was wrong. By placing the hidden elements complete randomly the assignments for the students would be completely different hard to solve. The main reason for placing hidden elements randomly was to prevent students from cheating. To solve the problem of different hard to solve problem sets the possible positions of the hidden charges were limited to different but similar hard to find positions.

One idea was to give the same assignments and vary the signs of the charges for every student. But this seemed to be too easy for the students to figure out that the positions are all same for every student and only the charges are different.

The final solution was to create a number of similarly difficult problem sets, about 20, and choose for every student randomly one of this problem sets. Additionally the first idea could be applied and the charges could be shuffled to create more different problem sets.

An idea for teaching was to create more problem sets that get harder and harder or to create sets showing special problems. These learning sets could be with enabled Show/Hide button so the student can easily check the positions. After the learning sets the exam set with disabled Show/Hide button is presented. For example starting with one hidden element what should be easy to find. And with every problem set another hidden element is added. Also special cases can be implemented, for example with very narrow placed hidden elements to show the influence of them to each other and the field. After some of this learning sets the student should be well prepared for the exam set.

6.4.2. The accidental gamification

During the work on the extended simulations they were presented to different people to get feedback as well as for testing and finding bugs. One observation was, that almost all of them talked about the extended simulations as games and about playing with them. In section 2.3 of this work the term *gamification* is discussed. Referred to the described definition of gamification, the extended simulations fulfill two points:

6.4. Findings

1. Challenge - During the learning sets the user of the extended simulations can check his answer and gets instantly feedback if he was successful.
2. Fun - Almost every person who tested the simulations had fun using it.

That does not mean that during this work games were created but it shows that the extended simulations fulfill some features of games.

6.5. Summary

This chapter describes the three extended simulations:

- Point Charges Extended - section 6.1
- Gausses Law Extended - section 6.2
- Amperes Law Extended - section 6.3

These three simulations were chosen as they are used in the first weeks of the course 8.02x. Fortunately from the programmers point of view these simulations were similarly to each other. Hence code could be reused for the second and third extended simulation. Otherwise, time would have been definitely too short to extend three simulations.

As the previous chapter 5 describes the technical background and the interface, this chapter is focused on the simulations themselves, on the requirements, the results and solutions as well as the handling.

The prime requirement for all simulations was that the extended simulations can be used for answering questions asked by the edX course. The idea for this three simulations was to place hidden elements in the simulations which have to be found by the student. This should make the students use the simulations on the one hand and on the other hand the work with the simulations should make the students understand the physical background behind the simulations.

The requirements summarized:

- When starting the simulation, hidden elements should be placed.
- The number of hidden elements should be determined by the course creator within the edX XML course file. The maximum should be 5 elements.
- The positions of the hidden elements should be defined by the course creator within the edX XML course file.
- The student can place dummy elements for marking his found positions.
- The dummy elements should be placeable on a grid.
- Grading should be done by Python within the edX XML course file.
- A *Show/Hide* button should be implemented which can be enabled and disabled by the course creator for every problem set.

7. Evaluation

This section covers an evaluation on how well the goals of this work were achieved. The work has influence on two different groups at the MIT. The first group is represented by the course creators and mentors of 8.02x. This group will not be part of the evaluation of this chapter as the whole work was initiated by this group. All the goals were defined by them, as described in section 6, and the progress was supervised by them. The second group is represented by the students of 8.02x. This section summarizes the evaluation of this work done by the students.

As described in section 2.7, an evaluation of 8.02x, held in spring 2014, showed, that students liked checkable answers. 79% of the students found them “Extremely Helpful” (Rayyan & Chu, 2014). That was the reason why the simulations were extended so that students can use them for answering questions.

The three main questions which should be answered within this section are:

1. Do students like the new simulations and do they enjoy working with them?
2. Does the use of the simulations improve learning?
3. How do students like answering questions using the simulations and getting instant feedback?

7.1. Procedure

The evaluation was done using the *Point Charges Extended Simulation* (see chapter 6.1). After a pre-questionnaire, the participants were introduced to this simulation and a task, containing five exercises, had to be solved. Finally, a short exam was done to show if students did understand the simulation and learned the physical effects of this exercise. After the experiment, the participants had to answer a post questionnaire.

7. Evaluation

The evaluation procedure consists of 5 steps for every participant:

1. **Pre questionnaire (5min):** Includes general questions about the participants. The questionnaire is included in the materials appendix on the CD.
2. **Introduction to point charges and interface (5min):** This is an introduction to the physics behind the *Point Charges Extended* simulation, the edX platform, the course page containing the simulation and the simulation itself.
3. **Testing and working with the *Point Charges Extended Simulation* (18min):** The participants will get five different problem sets of the simulation. Beginning with a very easy set they will get harder with every step. With this exercises the participants should learn and understand the influence of point charges on each other. During this exercises the *Show/Hide*-button will be enabled so that the participants can check their positions as often as they want.
4. **Exam problem set (8min):** The participants will get a problem set where they should find hidden point charges. The *Show/Hide*-button will be disabled and the participants can press the *Check*-button only once. This will be a hard to solve problem set and should show if the participants did understand the physics behind the simulation. Their answer is evaluated instantly and presented to them.
5. **Post questionnaire (5min):** Includes specific questions on the participants work with the simulation. Furthermore it includes questions about motivation and engagement of the participants. The questionnaire is included in the materials appendix on the CD.

7.1.1. Pre- and Post-Questionnaires

The pre-questionnaire consisted of eight questions about the participants, their age, education, learning type and their experience with computers and the Internet. Additionally, three question about their previous knowledge about physics are asked as well as their knowledge about point charges in particular. The whole questionnaire is attached in appendix [D.1](#).

The post questionnaire should evaluate the usability of the course page containing the *Point Charges Extended Simulation*, the training sets and the exam set. Therefore, the System Usability Scale (SUS) ([Brooke, 1996](#)) was used. It contains ten questions based on the Likert scale with a range from 0 to 5. Furthermore four questions about the interactive part of the simulations are asked. After that, two questions are asked to evaluate the game and

simulation character of the extended simulation as well as the learning and playing character. Finally, three open ended questions about the work with the simulation and the whole interface are asked. The whole questionnaire is attached in appendix D.2.

7.1.2. Test setup

The test sets and exam set were setup on a Laptop running a local edX server. The sets were implemented in a fork of a 8.02x physics course from the MIT. The used simulation for all training sets and the exam set was the *Point Charges Extended Simulation*, as shown in figure 6.1. The different sets are described in section 7.1.3 and shown in figure 7.1. Every participant could use a mouse or the laptop's built in touchpad for working. After an initial instruction, the participants had to use the sets on their own with no help from the instructor. The participants could use the test and exam sets for as long as they wanted. The overall time was measured.

7.1.3. The five training sets

The participants will be given five training sets to learn the physics behind the *Point Charges Extended* simulation and get used to the simulation and its interface. The five training sets are shown in figure 7.1. Each of the sets illustrates a different behavior of the point charges. For better understanding the point charge which can be moved by the participant for finding the hidden charges will be termed "moving charge".

The first problem set contains one single positive hidden charge near the border of the inner box. As the moving charge is also a positive charge, this simulation visualizes the influence of two positive point charges on each other. This charge is placed within the range of the field lines of the moving charge. The first problem set is shown in figure 7.1a.

The second problem set contains one single negative hidden charge in the middle of the simulation. The field lines are not reaching this charge. This problem set explains the interaction of a positive point charge with a negative point charge. Furthermore it demonstrates how to find a point charge if it is not reaching the field lines of the moving charge. The second problem set is shown in figure 7.1b.

7. Evaluation

The third problem set contains two hidden charges, a positive one and a negative one. It explains how two hidden charges affect each other. They are placed vertically over each other to show the influence just in one direction. The third problem set is shown in figure 7.1c.

The fourth problem set contains three hidden charges, two positive ones and one negative one. One positive charge is placed near to the negative charge. The second positive charge is placed away from the negative charge. This simulation should show the difference of finding a positive charge once influenced by a negative charge and once without any influence. The fourth problem set is shown in figure 7.1d.

The fifth experiment contains two positive hidden charges. One of them is inside the reach of the field lines of the moving charge. The second charge is placed behind the first one outside the reach of the field lines of the moving charge. Tests during the development have shown that this setup is hard to find as the second point charge is hidden behind the first one and hence can be missed easily. The fifth problem set is shown in figure 7.1e.

7.1.4. The exam set

After the participants have worked through the 5 training sets they are asked to solve the exam set. This set is shown in figure 7.1f. The places of the hidden charges in this set are quite hard to find, as tests with colleagues have shown during this work. Goal of this exam is to evaluate how helpful the different test sets have been for understanding the influences of point charges on each other.

The exam can not be restarted, every participant can submit his answer only once. The solution will be shown immediately after the answer was submitted.

7.2. Participants

The evaluation was done by 13 participants. All participants already had a basic physics course during their study at the Graz University of Technology or have finished High School and hence will have a basic physics course in near future. Six participants were male (46%) and 7 were female (54%). The participants were between 25 and 55 years old (23% between 25 and 30;

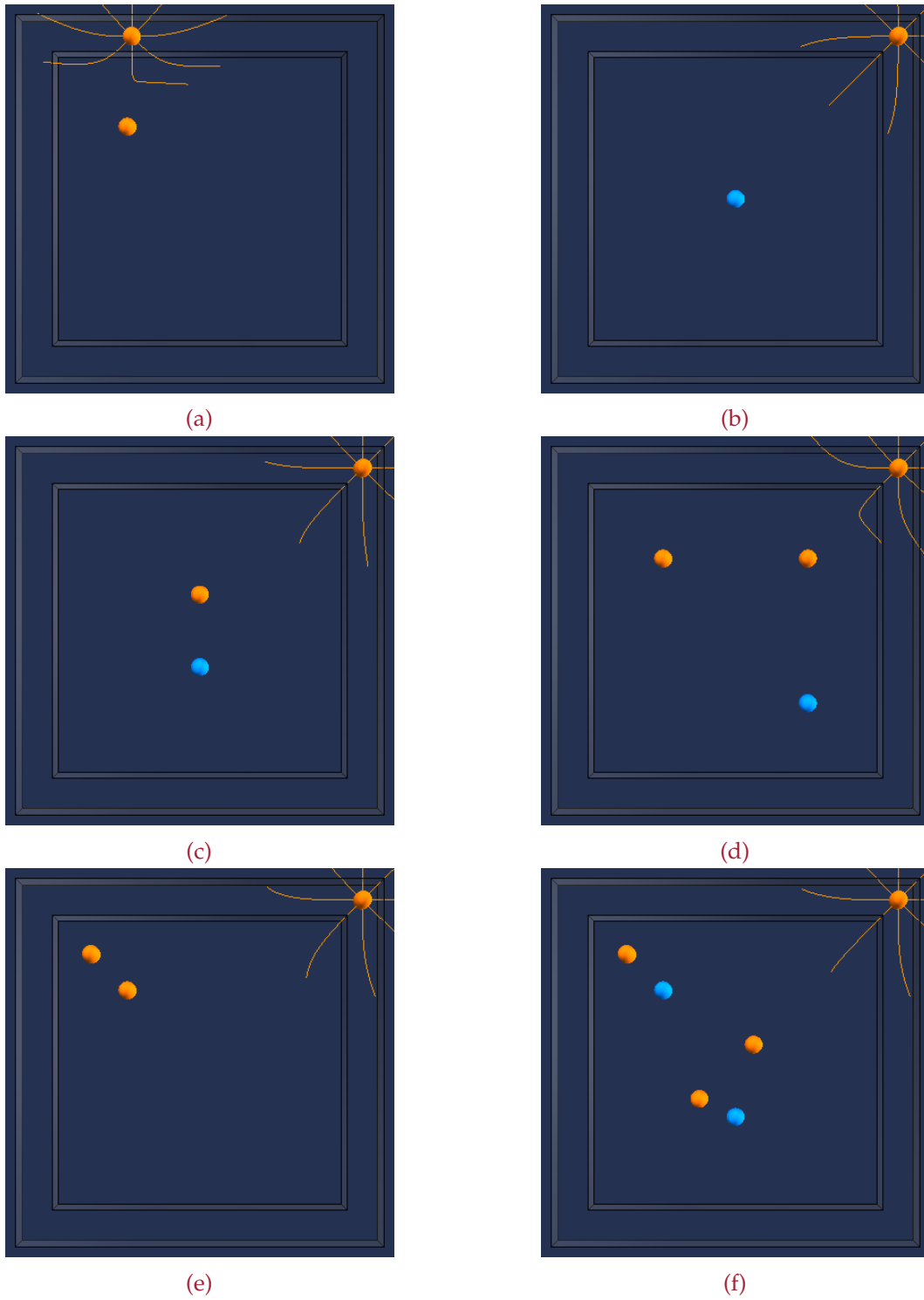


Figure 7.1.: Figures (a), (b), (c), (d) and (e) show the 5 training sets for the evaluation. Figure (f) shows the set for the final exam of the evaluation.

7. Evaluation

69% between 31 and 50; 8% over 50). All participants were familiar with the Internet and used it for work as well as at home. Two participants had a physics education (Bachelor). All learning types (auditory, visual and kinesthetic) were present in similar proportions.

7.3. Results

This section covers the results of the pre-questionnaire, the experiments and the post-questionnaire. The two questionnaires are composed of a collection of answers based on the Likert scale, open ended answers and comments. Furthermore the work of the participants with the simulations were logged and analyzed.

7.3.1. Pre-Questionnaire

The pre-questionnaire consisted of personal questions about the participants as well as two questions for evaluating the previous knowledge about physics and point charges in particular. Therefore, the participants were asked to draw field lines between two point charges. The first question consisted of two point charges, a red one and a blue one. For a correct answer, the participant had to describe the red point charge as a positive charge and the blue charge as a negative charge. Furthermore, the field lines between these two charges had to be drawn. This question should also show if the users knew, that point charges with different charge attract each other. Four participants, about 31%, were able to answer this question correct. Both users with the MSc in physics were among them. All others were not able to answer this question correct. The second question was similar to the first one but this time two blue point charges were used. The used questionnaire is attached in appendix [D.1](#).

7.3.2. Experiment

After a short introduction to the course page, the interface and the simulation, the participants could work free through the 5 training sets and the exam sets. There was no time limit. During their work, the participants were observed and the time they needed was measured.

Learning

As the extended simulations should make learning physics easier and more efficient the question “How much did the participants learn” was extremely interesting.

In the final exam the participants had to find 5 hidden charges as shown in figure 7.1. The grading was done by the following schema: For finding a hidden charge exactly the user got two points. For finding a hidden charge in an region of ± 2 the user got one point.

The mean result of all students where 6.3 points of 10 reachable points. This seems to be less but at this point it is important to say that the exam set of hidden charges was extremely hard and reaching 10 points is almost impossible. Even the creator of the exam set was not able to find all of them after two weeks he had placed them. The goal of this simulation was not to teach students where exactly hidden charges are placed. Instead, it was important to teach students how point charges influence each other. Hence, if students found the right amount of hidden charges and almost the correct positions, the goal of the extended simulations was fulfilled. The reason why the hidden charges where placed in such a hard constellation was to find the limits of the point charges simulation. For using the simulation during the real physics course at MIT it is important to know which tasks are solveable and which are too hard or even impossible to solve.

The result is examined in more detail here. First, as the goal was to teach the students where the hidden charges are approximately placed, five points per student would be a good performance. As the mean of points was at 6.3 this goal can be stated as fulfilled. Furthermore, the minimum of four points was reached only by one participant. As shown in section 7.3.2 this student was not motivated during the whole evaluation. All other participants reached at least five points or more.

Another aspect that should be discussed here is the question if there is a difference at the exam between participants with no previous knowledge about point charges, participants with previous knowledge and the two participants which were physicians. The comparison of these thre groups is shown in figure 7.2. The results show, that the physics experts could reach more points at the exam even with less trials on the test sets. The two groups of participants with and without previous knowledge reached almost the same number of points. Interestingly the participants with previous

7. Evaluation

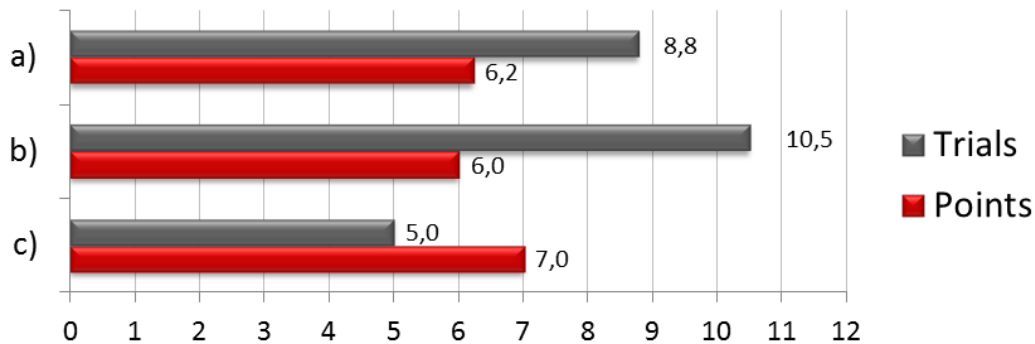


Figure 7.2.: Comparison of trials, at the test sets and points, at the exam, for the three groups of participants: a) Participants with no previous knowledge; b) Participants with previous knowledge; c) Participants with physics education

knowledge reached in average slightly less points although they did more trials with the test sets.

Motivation of the participants

This section investigates the motivation of the participants during the evaluation and its impact on the results of the final exam. Describing the motivation of the students is a little tricky. First of all, every participant had the choice to use the test sets as often as wanted. Additionally, they could skip the test sets entirely and just try the final exam. As every participant tried every test set at least once this can already be described as motivated.

In this thesis, the motivation was calculated by the number of test sets the student was willing to solve completely. Therefore, the number of trials a student needed to solve a test set was not counted, only if he was willing to find the solution or just pressed the *Show/Hide* button to see the solution and moved on. The result can be seen in figure 7.3. It shows that the motivation was quite high. On average the participants had 8.5 trials and 3 solutions of five where found. The figure also shows, that the motivation has only a short influence on the result at the exam. This also emphasizes how helpful the simulations can be, even if they are used only once. Furthermore, the influence of the number of trials a student made on the test sets is shown in figure 7.4. It shows the same findings as in the previous figure 7.3. The number of trials has only a small influence on the result at the final exam.

7.3. Results

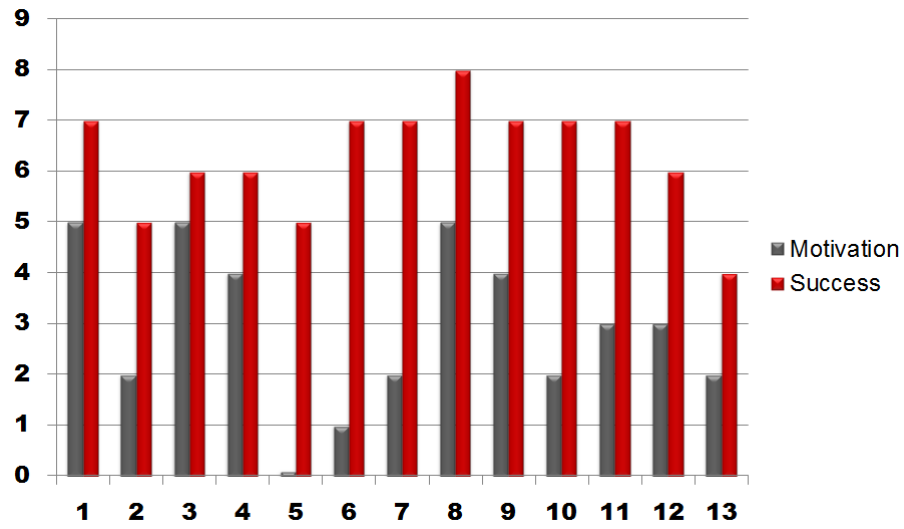


Figure 7.3.: Comparison of motivation during the test sets and success at the exam set.

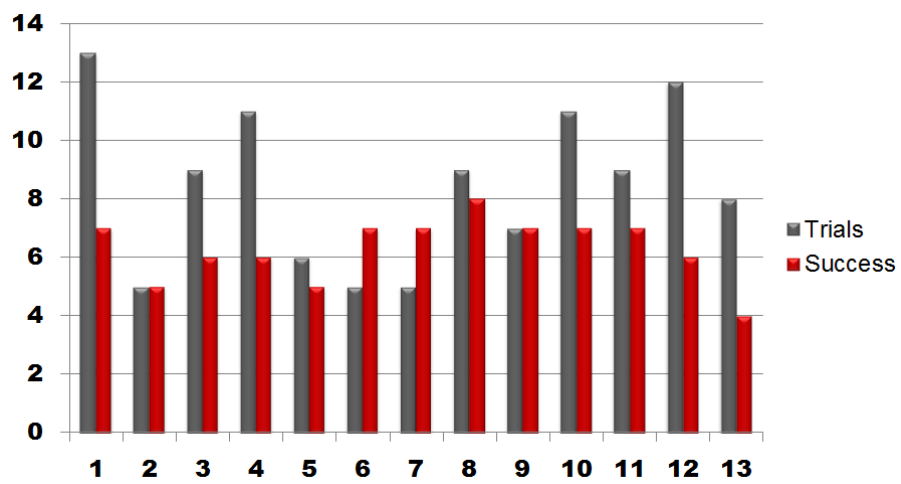


Figure 7.4.: Comparison between the number of trials at the test sets and success at the exam set.

7. Evaluation

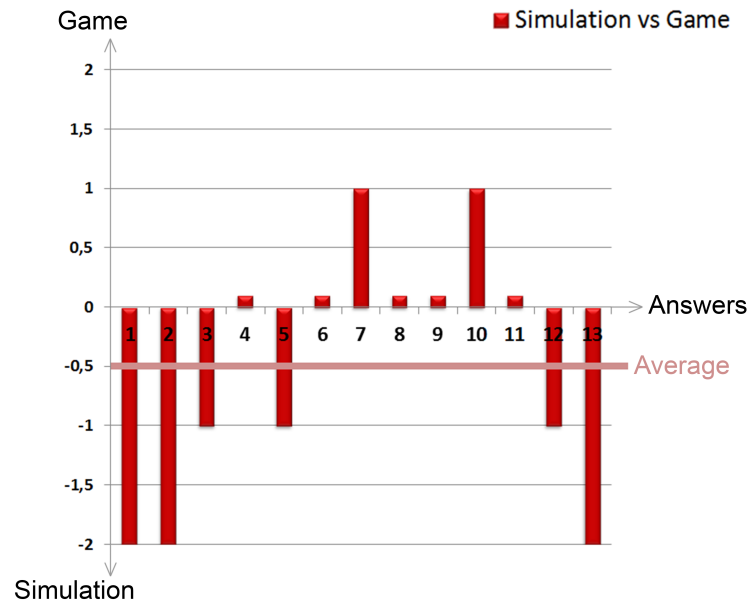


Figure 7.5.: Evaluation of the question "Would you describe working with the *Point Charges Extended Simulation* as using a simulation or a game?"

Simulation versus Game - Learning versus Playing

One main goal of using the edX system and the virtual experiments was to motivate the students to learn physics. The extended simulations are meant to be enjoyed and make learning funnier and easier. This led to two questions for the participants of the evaluation: While using the extended simulations, do the participants feel more like -

- using a simulation or playing a game?
- playing or learning?

These questions again used the Likert scale from *Simulation* (-2) to *Game* (+2) and *Learning* (-2) to *Playing* (+2). The results are shown in figure 7.5 and 7.6. The participants felt a little bit more like using a Simulation than a Game with an average at -0.5. Despite that, the participants felt more like playing a game than learning. The average of the second question was at 1.1. This shows, that using the extended simulations feels more like playing than like learning.

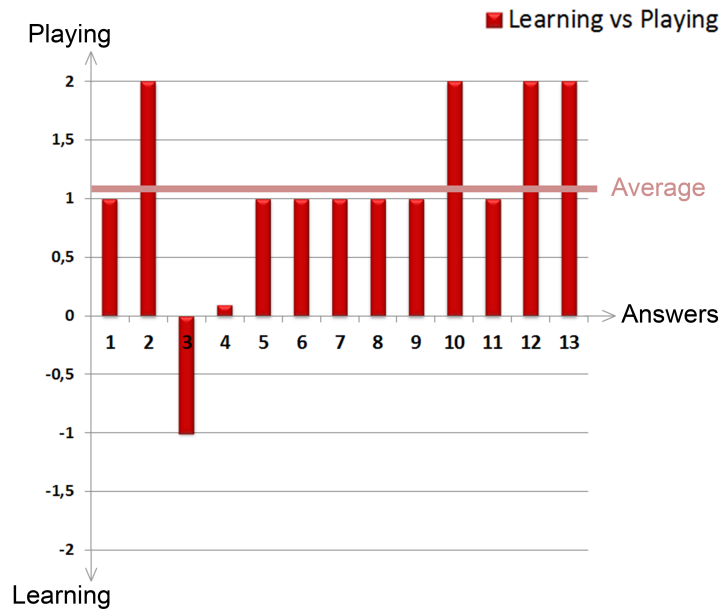


Figure 7.6.: Evaluation of the question “Did you feel more like *learning for an exam* or *playing a game* during your work with the *Point Charges Extended Simulation*?”

7.3.3. Post-Questionnaire

This questionnaire consisted of sixteen questions with answers based on the Likert scale. The first eleven had a scale between 1 (strongly disagree) and 5 (strongly agree). The remaining five had a scale from 1 to 5 but between hard and easy, standalone and online, original simulation and extended simulation, simulation and game and learning and game. The first ten questions are based on the *System Usability Scale (SUS) Test* (Brooke, 1996). SUS provides a standardized general measurement of usability resulting in a usability scale which reaches from 0 to 100. The used questionnaire is shown in appendix D.2.

Most participants found that the simulations were easy to use ($M=4,69$; $SD=0,48$) and would like to use the extended simulations in future frequently ($M=4,62$; $SD=0,51$). Even more found the instant feedback after submitting the answer very important ($M=4,77$; $SD=0,44$).

7. Evaluation

Usability

The usability of the extended simulations was evaluated using the *System Usability Scale* (SUS). This is a general test for evaluating the usability of industrial systems. The score ranges from 0 to 100 and higher numbers reveal better usability. All participants rated the usability with a SUS score of 75 or higher. The highest score was 95 and the average was 88.5 (SD=6.0).

Open ended questions

All participants mentioned that the extended simulations can improve learning compared to the traditional learning without virtual simulations and compared to the original simulations with no instant feedback. Additionally the online system was liked very much.

The participants really enjoyed working with the simulations and mentioned the benefits over traditional learning. One participant answered on the question, that he enjoyed most *“Observing the changes of the field lines in real time”*. Others enjoyed the *“Animation”* and that *“It was interactive”*. Having fun by dragging the moving charge, often called as *“Spider”*, emphasizes the feeling of playing a game rather than learning for an exam.

Even though the participants really appreciated the instant feedback after the exam it was not enough to tell the participants how many points they had achieved. One participant said: *“After Check, I did not see my Result compared to the solution”*. Showing the students their given answer and the solution could improve the understanding for further exams as they could learn from their errors they made.

Two answers of the question, *What did you enjoy most during your work with the simulations?*, describe very well how the simulations could motivate the users: *“Solving the problem.”* and *“Knowing the answers.”* This also shows the importance of well balanced test sets. While challenging but solvable sets can improve ambition and motivation, sets which are too easy to solve bore students and too hard test sets can demotivate, as one participant answered on the question what he disliked most: *“Not knowing the answers”* or *“Not being able to solve the problem.”*

An important feature of the edX course is to give the students the possibility to use every test sets as often as they want, also during the exam. One participant mentioned, that he really enjoyed to *“Be able to jump between the*

individual test sets and show again the solution for this set.". This give the user the possibility to call to mind what they learned at specific test sets.

Negative points where mostly caused by the prototype setup of the simulation pages for this evaluation. For example, one participant mentioned that he disliked most, that *"The simulations is not started automatically."* For the test sets, every simulation had to be started by pressing the *Start Simulation* button. Almost every participant moved the moving charge after opening an test set without starting the simulation what always caused frustration as it took them a few seconds to realize that the simulation was not running. Another point was that the movement of the charge stopped completely when the mouse was moved inside the inner box. Here a fine tuning of the movement constraints should be done.

All simulations are implemented three dimensional and by clicking somewhere between the point charges the simulation can be rotated. But this feature was also a point of criticism: *"Missing the moving charge with the mouse rotates the scene ... unnecessary!"*

During the questionnaire the participants where asked for additional features. One participant mentioned, *"Seeing the field lines. This would improve the understanding of physics between point charges."* Another participants said it more direct: *"Show field lines!"* This feature was also requested by the participants while they were working with the test sets. The function of showing the field was implemented in the original simulations and hence would be easy to be reactivated for the extended simulations.

Additionally a free to use simulation parallel to the given test sets was requested: *"It would be good to allow placing the hidden charges by our self to play around with the resulting test set."* This would give the participants the possibility of creating their own training sets.

7.4. Discussion

In total the evaluation was very successful. All participants really enjoyed working with the extended simulations and the final exam showed that all participants understood the influence of point charges on each other. Furthermore, most participants felt more like playing than learning as shown in figure 7.5 and 7.6. After the evaluation, almost every participant mentioned that he would like to have this system at the Graz University of Technology.

7. Evaluation

The reason why the simulations were developed at the MIT was to motivate the students to learn physics and make learning more efficient. The evaluation has shown, that the participants were really motivated to work with the simulations and had fun using them. It also shows, that participants which were less motivated and used every training set only once also reached good results at the final exam. This shows, that the chosen five test sets were a good preparation for the final exam. As only one participant reached four points (Five were needed to get a grade 1 on the exam) and all other five or more, it can be said that the *Extended Point Charges Simulation* performed very well!

The difference in the results of the final exam between participants without previous knowledge about point charges and physics experts was quite small and all participants were able to reach enough points for a good grade. This shows again the efficiency of the simulations.

After working with the simulations, the participants were asked to answer the post questionnaire, which first ten questions were about the usability of the whole system. That includes the simulations as well as the whole interface around them and the edX course page where the simulations were embedded. The usability was evaluated by using the SUS test, which is a score with range from 0 to 100. The participants rated the system with a score between 75 and 95 ($M=88.5$; $SD=6.0$). This is a great result which shows that not only the simulations were a big success but also the whole interface and the edX course page were really liked by the participants.

Open ended questions at the end of the post questionnaire could be used for describing problems while using the simulations and suggestions for improvements. One problem, that nearly every participant had, was that the simulations were not started automatically after loading the course page. Additionally the moving charge needs a fine tuning of the collisions with the borders.

Two improvements were stated often. First, the wish for seeing the whole field of the test sets when the hidden charges are shown. Participants stated, that this would improve the learning a lot, as they could see more details in the field. The second improvement was to provide the participants with a free to use simulation, where point charges can be placed. This would give the participants the possibility of creating their own test sets and can help learning specific problems.

8. Future Work

Professor Belcher is a visionary. He always tried to find new learning methods and additional experiments to help students learn the topics of the basic physics courses. That started with experiments in front of webcams, went to the virtual TEAL simulations and came to the online platform edX. This work is part of the evolution of the basic physics courses at the MIT.

8.1. Near future - The rest of the seven TEAL simulations

During this master thesis a concept for the rest of the simulations was designed and how they could be used for interactive questions. Indeed, it was clear that six months are way too short to extend all of the seven TEAL simulations, but it was also done to know which requirements the interface will have to fulfill in the future. During the entire development, the interface was tried to be kept as open and universal as possible to make sure it can be used for future work.

At this point, the ideas for the remaining four simulations are presented which shall be on the one hand good for education and on the other hand realistic to implement from the programmers point of view. It is important to keep in mind that these are only ideas which were not tested. As the work on the three extended simulations has shown, there can be a huge difference between the first ideas and the final simulations. The reasons can be different. For example, one problem was that an assignment was simply unsolvable with the tested implementation. Another idea seemed to be interesting from the programmers point of view but from the educational point of view it was senseless. Other ideas were simply tried out and presented to colleagues for feedback.

8. Future Work

Floating Coil

In this simulation a current carrying coil rests on a platform centered on the axis of a permanent magnet. With the slider on the right the current in the coil can be set which can pull the ring towards the magnet or push it away from the magnet. The simulation is shown in figure 8.1.

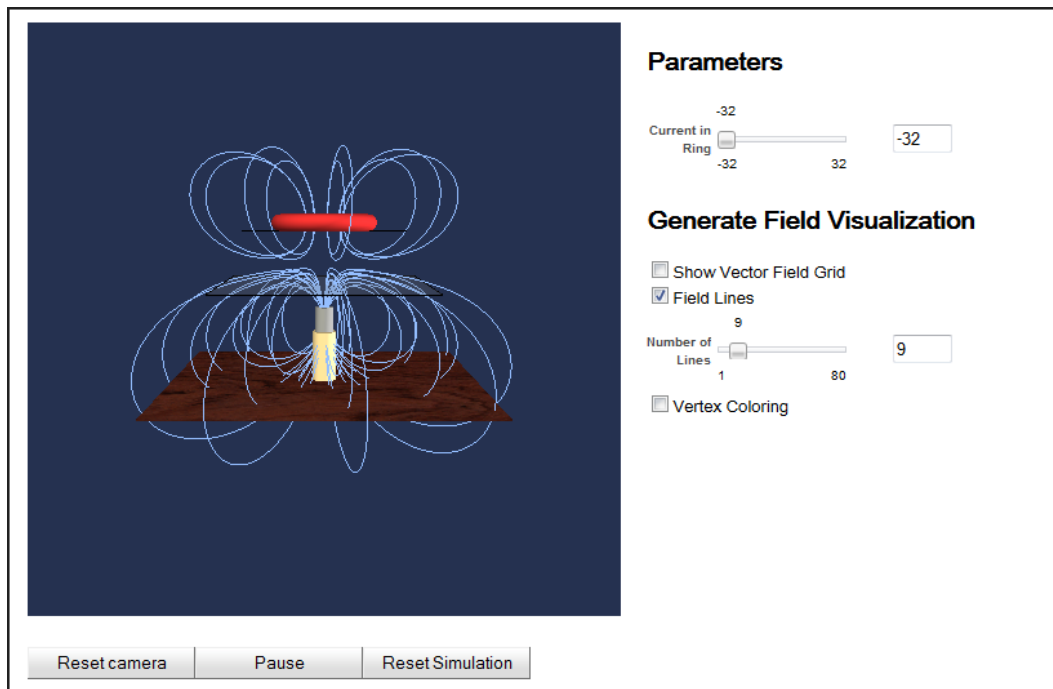


Figure 8.1.: The Floating Coil simulation

One idea was to randomly rotate the magnet, with the north pole or the south pole at the top. The student has to find out which pole is at the top depending on which current he chose, a positive or negative one.

Another question could be which current is used to elevate the ring above a given height. Therefore, a height-meter would be implemented, which could be a single text box.

Faraday's Law

In this simulation a magnet is movable on the axis of a ring with a self-inductance and a resistance. It is shown in figure 8.2.

8.1. Near future - The rest of the seven TEAL simulations

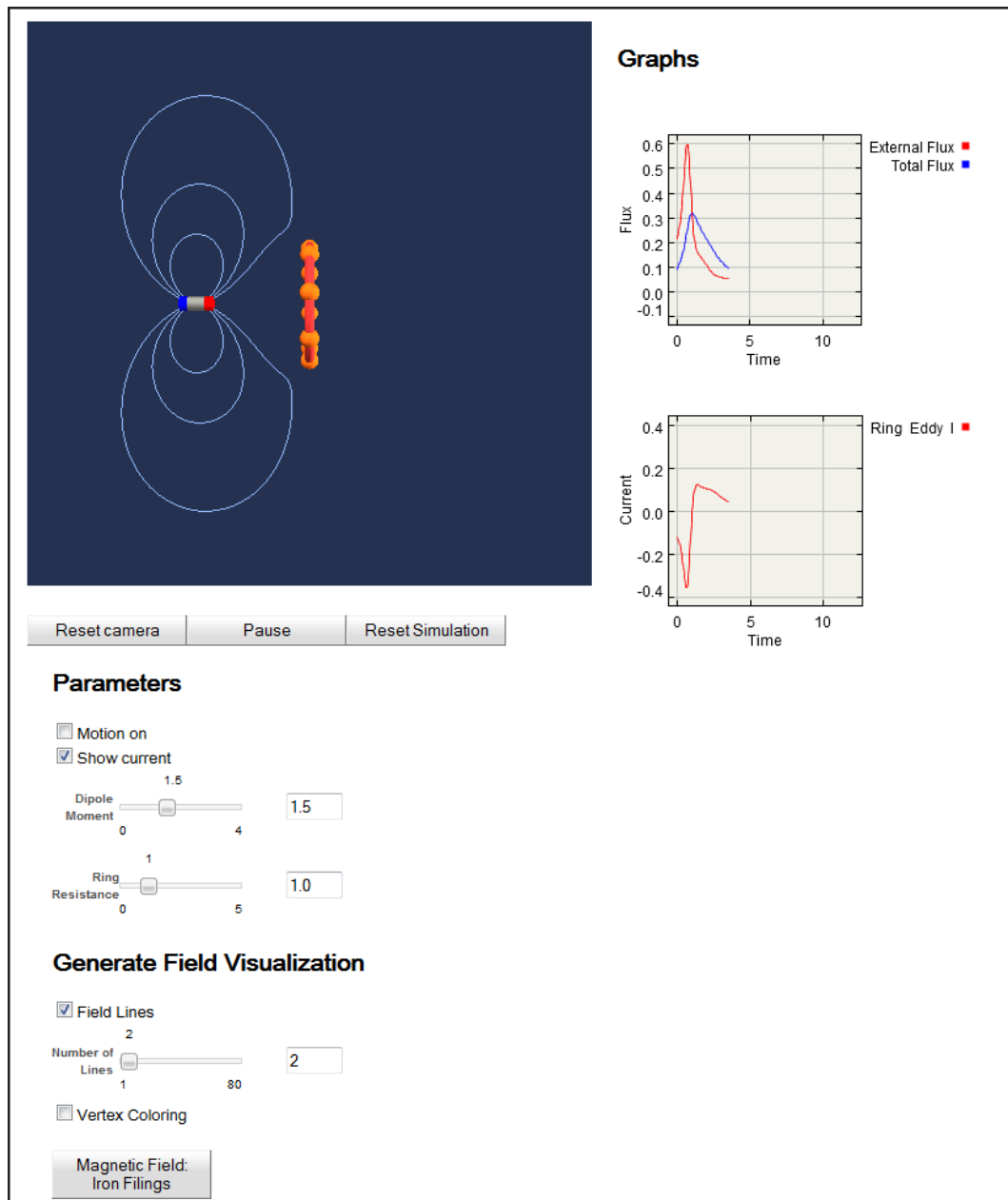


Figure 8.2.: The Faraday's Law simulation

When moving the magnet along its axis through the ring, the ring will start rotating. The rotation speed depends on the speed of the movement of the magnet. The rotation direction depends on the direction of the movement of the magnet. Some different states of the simulation can be seen in figure 8.3.

8. Future Work

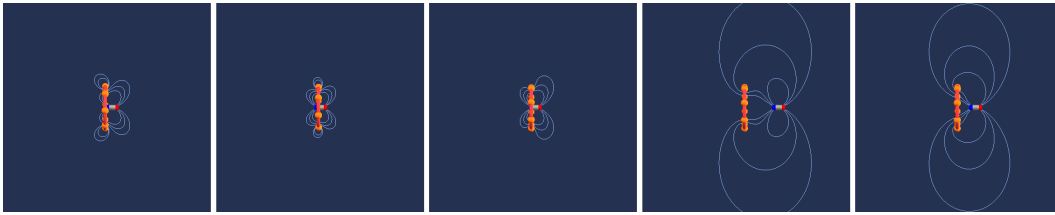


Figure 8.3.: A series of the Faraday's Law simulation

A further option is to let the magnet automatically move sinusoidally through the ring what leads to the sinus-shaped curves which can be seen in figure 8.4 on the right.

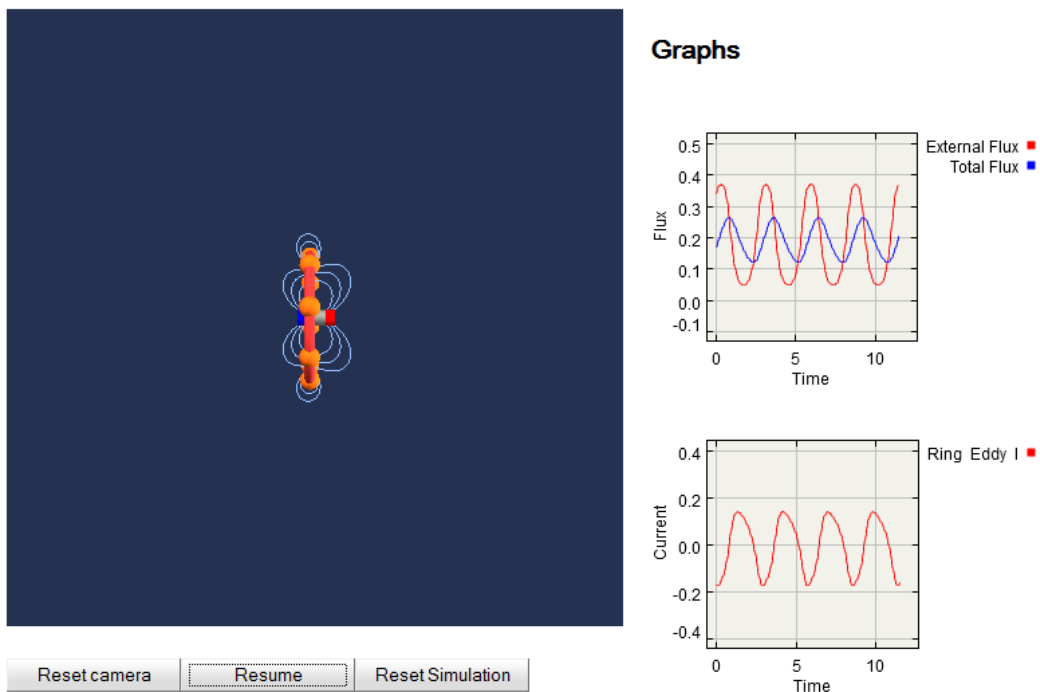


Figure 8.4.: The Faraday's Law simulation with sinusoidally moving magnet.

Questions for this simulation could be to give the student a screen-shot of the flux and current curves showing sinus-shaped waves. The student has to set the right *Dipole Moment* and *Ring Resistance* to obtain the same curves as questioned.

Furthermore, new sliders could be implemented for the movement of the magnet, one for the speed and one for each endpoint of the magnet. The

8.1. Near future - The rest of the seven TEAL simulations

student should then again set this sliders to get the same curves as questioned.

Charge by Induction

Figure 8.5 shows the *Charge by Induction* simulation. With the slider on the right the current of the big charge in the middle can be set. There are a number of point charges between the cylinders. Furthermore, two other visualizations can be displayed as shown in figure 8.6.

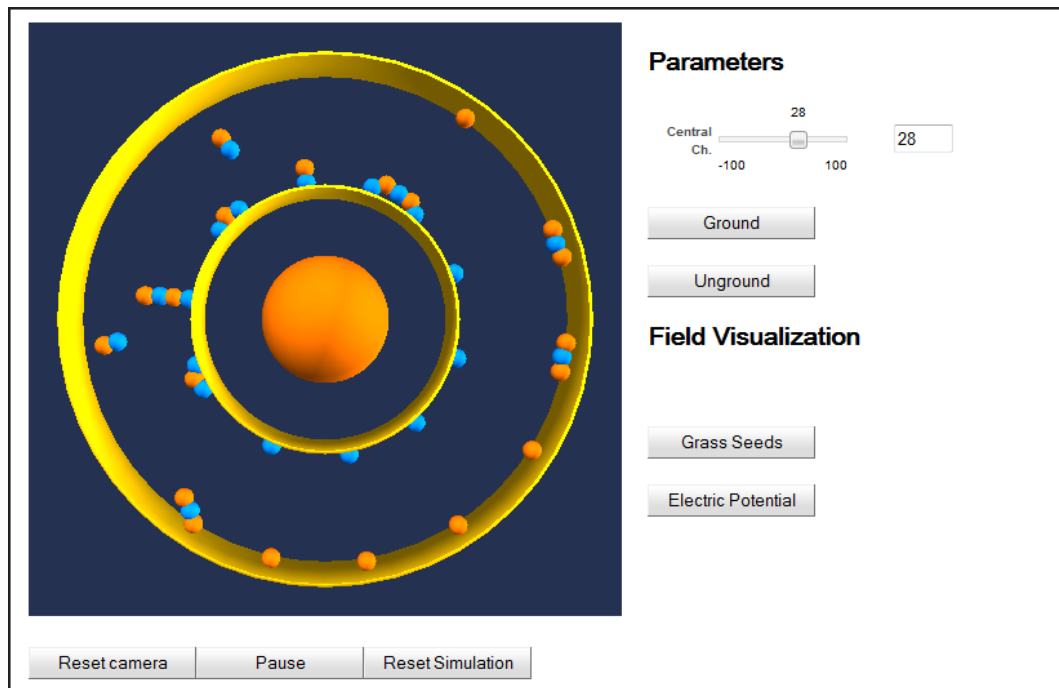


Figure 8.5.: The Charge by Induction simulation.

Figure 8.6 shows one state of the simulation in all three possible visualizations.

8. Future Work

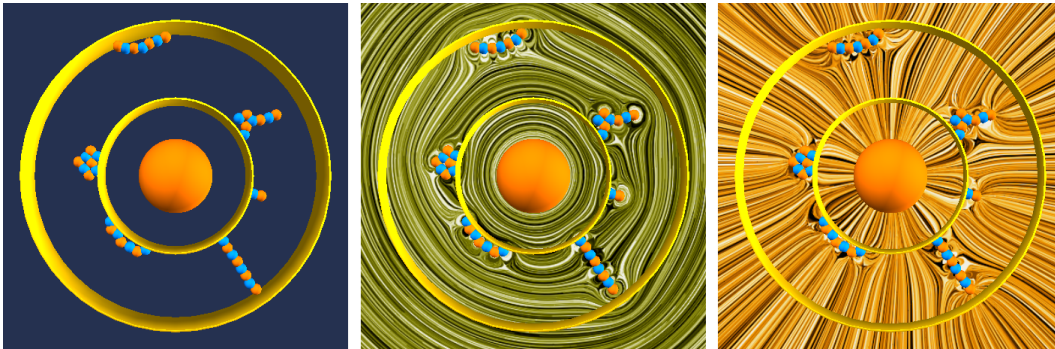


Figure 8.6.: Different visualizations of the Charge by Induction simulation.

One idea for this simulation was that the course creator can place point charges, as many as he wants and where he wants. Then the two visualizations are stored. The student gets this visualizations as background of his simulation and has to place point charges to accomplish the same field visualization. For this idea again a number of different sets can be created and every student could get one of them randomly.

Plane Wave simulation

The simulation shows a pink sheet of positive electric charges. Electromagnetic waves are generated by shaking the plane. The yellow arrows represent the radiation electric field, the blue arrows the radiation magnetic field. The sheet can be moved up and down by the user, as seen in figure 8.7. The sheet can also be automatically moved up and down which can be seen in figure 8.8. The movement leads to sinus-shaped arrows. The idea was to give the student a screen-shot of this simulation with sinus-shaped arrows of specific length. The student should then have sliders for automatically moving the sheet to obtain the same sinus curve. The student should have to set the direction of the movement (up/down and backward/forward) as well as the speed of each direction.

8.1. Near future - The rest of the seven TEAL simulations

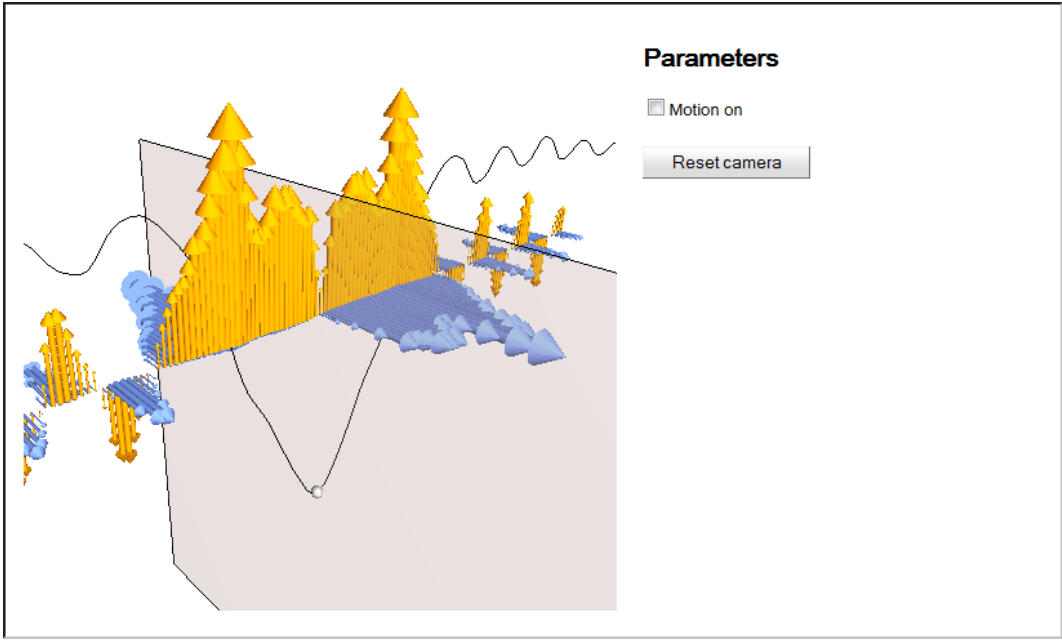


Figure 8.7.: The plane wave simulation.

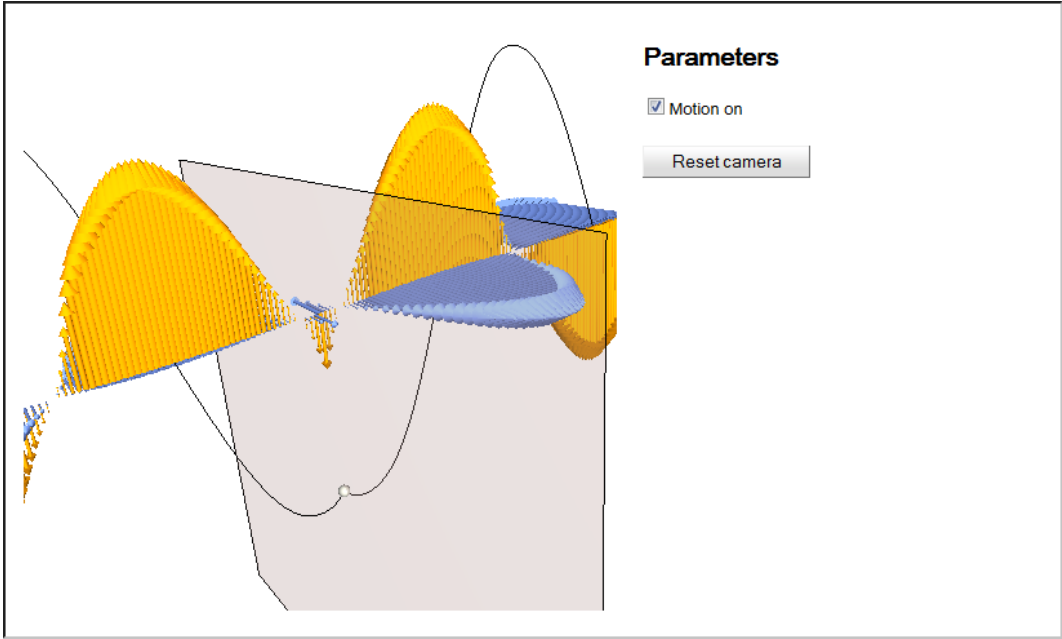


Figure 8.8.: The plane wave simulation.

8. Future Work

8.2. Ultimate goal - A complete new framework for simulations

One of the disadvantages of the seven simulations which were ported to JavaScript was, that each one was ported on its own and by different people. That means there is code rewritten for each simulation. And so is the new code which was added for extending them. The reason for this circumstance was time, as this was the fastest and easiest way. The physics courses with the simulations (and now the extended simulations) are still in development and evolution.

A huge benefit of e-learning and online courses is the detailed feedback from the students about every part and aspect of them. The seven simulations have shown that they can help students understanding physics but evaluations have shown that students made insufficient use of them. The extended simulations are the next evolutionary step of the basic physics courses. And depending on future evaluations they will be improved again.

The next step will be to create a whole new framework for physics simulations. The programmers at CECI call it a *player* for the simulations. An environment where many different simulations can run and interact with an enveloped system like edX.

The benefits for such a player would be on the one hand an easier and faster implementation of new simulations as it would provide a broad library. This can help reuse code in all simulations and prevent duplicated code. On the other hand it could implement a more stable and extensive interaction with edX. This could be done in cooperation with edX and hence, it could be used by other institutes too.

8.3. Improvements on jsinput

One problem during this work was the documentation of jsinput. At the time this work was written, jsinput was still in development and changed through the last years. And so did the documentation. That is the reason why different versions of jsinput documentation were found and often only trial and error worked to find out how things work. During this work, also feedback was given to edX and their developer which will be considered in future versions of *jsinput*.

8.3. Improvements on jsinput

One thing that will happen is, that jsinput will get an mechanism to send initial values from edX to JavaScript applications. And of course there will be a lot of improvement in future of jsinput. Perhaps a new version of jsinput will be released what would make our work a lot easier.

9. Summary

The basic physics courses at the MIT are not the favorite courses of students. To increase the motivating students for learning and furthermore make learning for efficient, the TEAL classroom was introduced. It offers different multimedia techniques as well as interactive experiments, the so called TEAL simulations. Over the last years the physics courses were brought to an e-learning course using the online platform edX. Part of this online courses were seven TEAL simulations which were ported to JavaScript. Being more exact, the simulations were written in Java and compiled to JavaScript using GWT. The simulations should help students understand the learning topics. Evaluations of previous courses have shown that students did not or just very short work with these simulations. So the idea was born to make them interactive and use them for answering questions through online homeworks and exams.

The first part of this work was to examine if it is possible to create an interface between edX and the seven TEAL simulations to achieve communication and interaction between them. To solve this problem the edX function *jsinput* was used. *Jsinput* was made to provide a communication between the edX course, which is written in XML, and stand alone JavaScript applications. Unfortunately two problems came up while using *jsinput*. First, the existing seven TEAL simulations were not written in JavaScript, instead they were written in Java and afterwards compiled to JavaScript. First tests showed that it was not possible to set up a stable communication using *jsinput* with the existing simulations, there were always synchronization problems. The second problem was the lag of sending initial values to the JavaScript application using JavaScript.

For the synchronization problem the HTML host file of the JavaScript simulations were used as a buffer between edX and the simulations. That means, edX did not directly communicate with the simulations, it was communicating only with the HTML host file. On the other hand, the simulations were also communicating only with the HTML host file. This led to a stable and reliable communication.

9. Summary

For the initial values problem a work around was developed. As *jsinput* could not be used, a second communication method, using *postmessages*, was implemented. This are messages sent by JavaScript between the edX XML course file and the HTML host file. Again, this was no direct communication between edX and the simulations.

Once the technical problems where solved, the first simulation was extended. The *Point Charges Simulation* was chosen to be the first simulation that was extended as this would be the first simulation used during the physics online course. The idea was to place hidden charges which have to be found by the students. Furthermore these hidden charges should be placed randomly to avoid cheating between students. The second simulation which was extended was the *Gausses Law Simulation* and the third simulation was the *Amperes Law Simulation*. Both extended simulations implement a similar approach to the first extended simulation using hidden elements which have to be found by the students.

An evaluation was done using the new *Point Charges Extended Simulation* as well as a pre- and post-questionnaire. The pre-questionnaire consisted of questions about the participants and their previous knowledge about physics. The post-questionnaire was based on the standardized questions of the *System Usability Score* (SUS) test. Additionally, questions about the environment and pedagogical aspects where asked followed by open ended questions. For the experiments a local edX server was set up running the test system. The participants could work with five training sets and afterwards had to do an exam set. These sets used the *Point Charges Extended Simulation* and hidden point charges were placed which had to be found by the participants.

The evaluation shows, that the new challenge of finding hidden elements gave the simulations more game character and hence learning was felt more like playing a game. This was one main goal for developing the extended simulations. The participants also mentioned that using the extended simulations while learning for an exam can improve the learning and make it more effective and also more fun. This was also seen in the results. Almost all participants where able to reach grade 1 at the final exam, only one participant had one point less. A comparison of participants with and without previous knowledge about physics showed no difference in the results. Only the physics experts where able to reach slightly more points. Furthermore, the usability of the interface was rated very high with an average score of 88.5 of 100. Here the possibility of jumping between the test sets and reusing them as often as the participants wanted was very appreciated.

An improvement that was mentioned very often was the wish to show the whole field of the simulation during the test sets and not only the short field lines of the moving charge. This could improve learning very much as more details could be seen. Also a free to use set was requested where students can create their own training sets and train for specific problems.

The evaluation shows that the extended simulations can be a real improvement and benefit for upcoming basic physics courses at the MIT.

List of Figures

2.1. Reasons why users enroll and start the MOOC (Gütl et al., 2014).	9
2.2. Reasons why users left the MOOC (Gütl et al., 2014).	9
2.3. A TEAL class room (Dori & Belcher, 2005).	15
2.4. Mid-Semester evaluation of 8.02x, spring 2014 (Rayyan & Chu, 2014)	17
2.5. Mid-Semester evaluation of 8.02x, spring 2014 (Rayyan & Chu, 2014)	18
3.1. An example for an edX course with an interactive homework opened.	24
3.2. An example for the segmentation in separate files of the edX course 8.02x.	27
3.3. The GWT Compiler in Detail (Adam et al., 2013)	29
3.4. Starting a GWT application (Adam et al., 2013)	30
4.1. The original point charges simulation.	35
5.1. Structure of an edX course page which includes a non interactive simulation.	40
5.2. The first extended simulation. The point charge on the left top is the movable point charge with visible field lines. The two others are dummy charges with no effect on the electrical field.	41
5.3. Structure of a course page that embeds a HTML file using jsinput. Illustrating the data stream.	44
5.4. Structure of a course page that embeds a HTML file using postmessages for sending the values of the initial positions of the hidden charges to the HTML host file.	48
5.5. Structure of the final test course page that embeds a HTML file containing a GWT created simulation.	53
5.6. Data stream of saving and reloading the simulations state and the used data types.	57

List of Figures

5.7.	Flowchart of the complete communication between edX and our extended simulation.	63
6.2.	Top: The simulation after page was loaded Bottom: The simulation when started	69
6.3.	Top: Running simulation with hidden charges and placed dummies on the sides. Bottom: Running simulation with hidden charged shown. This was our test set for the hidden charges.	70
6.4.	Example for a hard to find position of a hidden charge.	71
6.5.	A prototype of the Point Charges Extended simulation.	72
6.6.	The final <i>Gausses Law Extended</i> - Simulation	74
6.7.	Top: The simulation after page was loaded Bottom: The simulation started	76
6.8.	Top: Running simulation with visible <i>hidden charges</i> in our test configuration. Bottom: Running simulation with correct placed charges over the hidden charges.	77
6.9.	Placed charges and their influence on the field arrows.	78
6.10.	The final <i>Amperes Law Extended</i> - Simulation	80
6.11.	Top: The simulation after page was loaded Bottom: The simulation started	81
6.12.	Top: Field induced by a line current. Bottom: An opposite line current placed on top of the first current. The field gets zero.	82
6.13.	Simulation with all line currents placed.	83
7.1.	Figures (a), (b), (c), (d) and (e) show the 5 training sets for the evaluation. Figure (f) shows the set for the final exam of the evaluation.	91
7.2.	Comparison of trials, at the test set,s and points, at the exam, for the three groups of participants: a) Participants with no previous knowledge; b) Participants with previous knowledge; c) Participants with physics education	94
7.3.	Comparison of motivation during the test sets and success at the exam set.	95
7.4.	Comparison between the number of trials at the test sets and success at the exam set.	95
7.5.	Evaluation of the question "Would you describe working with the <i>Point Charges Extended Simulation</i> as using a simulation or a game?"	96

7.6. Evaluation of the question "Did you feel more like <i>learning for an exam</i> or <i>playing a game</i> during your work with the <i>Point Charges Extended Simulation</i> ?"	97
8.1. The Floating Coil simulation	102
8.2. The Faraday's Law simulation	103
8.3. A series of the Faraday's Law simulation	104
8.4. The Faraday's Law simulation with sinusoidally moving magnet.	104
8.5. The Charge by Induction simulation.	105
8.6. Different visualizations of the Charge by Induction simulation.	106
8.7. The plane wave simulation.	107
8.8. The plane wave simulation.	107

List of Abbreviations

CECI	Center for Educational Computing Initiatives
e-learning	Electronic Learning
ETH Zürich	Eidgenössische Technische Hochschule Zürich
GWT	Google Web Toolkit
HTML	Hypertext Markup Language
JSNI	JavaScript Native Interface
MIT	Massachusetts Institute of Technology
MOOCs	Massive Open Online Courses
SDK	Software Development Kit
STEM	Science Technology Engineering and Mathematics
TEAL	Technology-Enabled Active Learning
TEALsim	Technology-Enabled Active Learning Simulation
TU Graz	Technical University of Graz
URL	Uniform Resource Locator
XML	Extensible Markup Language

Bibliography

- Adam, T., Robert, H., Jason, E., and Tökke, A. (2013). *GWT in Action*. Manning Publications Co., 2 edition.
- Belcher, J., Mckinney, A., Bailey, P., and Danziger, M. (2006). TEALsim Documentation: The Software. Retrieved June 08, 2014 from <http://web.mit.edu/viz/soft/visualizations/tealsim/docs.htm>.
- Brooke, J. (1996). Sus-a quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4-7.
- Brown, S. (2013). Back to the future with moocs. *ICICTE 2013. Proceedings*, pages 237-246.
- Clayton, G. and Theodora-Ismene, G. (2005). Learning through Simulation or Simulated Learning? An Investigation into the Effectiveness of Simulations as a Teaching Tool in Higher Education.
- Crooltall, D., Oxford, R., and D., S. (1987). Towards a reconceptualization of Simulation: From Representation to Reality. *Simulation/Games for Learning: The journal of Sagset*, 17(4):147-171.
- Dori, Y. J. and Belcher, J. (2005). How Does Technology-Enabled Active Learning Affect Undergraduate Students' Understanding of Electromagnetism Concepts? *Journal of the Learning Sciences*, 14(2):243-279.
- edX website (2014a). Retrieved November 21, 2014 from <https://www.edx.org>.
- edX website (2014b). EdX documentation: Building and Running an edX Course. Retrieved December 12, 2014 from https://odl.mit.edu/wiki/Running_edX_Locally.
- edX website (2014c). EdX documentation: Running edX Locally. Retrieved November 14, 2014 from https://odl.mit.edu/wiki/Running_edX_Locally.

Bibliography

- Gibbons, A. S., Fairweather, P. G., and Anderson, T. A. (1997). Simulation and Computer-Based Instruction: A Future View. In *Instructional Development Paradigms*, chapter 43, pages 769—801.
- Gütl, C., Rizzardini, R. H., Chang, V., and Morales, M. (2014). Attrition in MOOC : Lessons Learned from Drop-Out Students. In *Learning Technology for Education in Cloud. MOOC and Big Data*, pages 37–48. Springer.
- Hernández, R., Pardo, A., and Kloos, C. D. (2007). Creating and Deploying Effective eLearning Experiences using .LRN. 50(4):345–351.
- Iosup, A. and Epema, D. (2014). An experience report on using gamification in technical higher education. *SIGCSE '14 - Proceedings of the 45th ACM technical symposium on Computer science education*, pages 27–32.
- Jenkins, H., Klopfer, E., Squire, K., and Tan, P. (2003). Entering the Education Arcade. *Computers in Entertainment (CIE)*, 1(1):8:1–8:11.
- Kiryakova, G., Angelova, N., and Yordanova, L. (2014). Gamification in education. Proceedings of 9th International Balkan Education and Science Conference.
- Lee, J. J. and Hammer, J. (2011). Gamification in education: what, how, why bother? definitions and uses. 15(2):1–5.
- Lunce, L. M. (2006). Simulations: Bringing the benefits of situated learning to the traditional classroom. *Journal of Applied Educational Technology*, 3(1):37–45.
- MIT iCampus website (2014). Retrieved November 28, 2014 from <http://icampus.mit.edu/projects/teal>.
- MIT physics department website (2014). Retrieved November 17, 2014 from <http://web.mit.edu/firstyear/2018/subjects/physics.html>.
- MIT website (2014a). Retrieved October 29, 2014 from <http://www.mit.edu>.
- MIT website (2014b). General Institute Requirements of the MIT. Retrieved October 02, 2014 from <http://web.mit.edu/catalog/overv.chap3-gir.html>.
- MITxVM webpage (2014). MITx / edX Tools. Retrieved December 12, 2014 from <http://people.csail.mit.edu/ichuang/edx>.

- Muntean, C. I. (2011). Raising engagement in e-learning through gamification. In *Proc. 6th International Conference on Virtual Learning ICVL*, pages 323–329.
- Pirker, J. (2013). The Virtual TEAL World - An Interactive and Collaborative Virtual World Environment for Physics Education. Master's thesis, Graz University of Technology.
- Rayyan, S. and Belcher, J. (2014). *8.02 TEAL+x: Students Say "Yes" to MITx in 8.02 TEAL*. Retrieved December 12, 2014 from http://web.mit.edu/fnl/volume/272/rayyan_belcher.html.
- Rayyan, S. and Chu, C. (2014). Spring 2014 8.02 MITx Mid-Semester Survey. Technical report, Massachusetts Institute of Technology.
- Rizzardini, R., Gütl, C., and Chang, V. (2013). MOOCs Concept and Design using Cloud-based Tools: Spanish MOOCs Learning Experiences. In *Proceedings of the Sixth International Conference of MIT's Learning International Networks Consortium (LINC)*.
- Rodriguez, O. (2013). The concept of openness behind c and x-MOOCs (Massive Open Online Courses). *Open Praxis - Special Issue: Openness in Higher Education*, 5(1):67–73.
- Rutten, N., Joolingen, W. R. v., and Veen, J. T. v. d. (2012). The learning effects of computer simulations in science education. 58(1):136–153.
- TEALsim website (2014). TEALsim Project at MIT. Retrieved December 12, 2014 from <http://web.mit.edu/viz/soft/visualizations/tealsim/index.html>.
- Van de Pavoordt, P. (2012). Gamification of education. Retrieved January 21, 2015 from <http://www.cs.vu.nl/~eliens/sg/local/essay/12/17.pdf>.
- W3schools website (2014). Retrieved March 10, 2014 from http://www.w3schools.com/html/html5_browsers.asp.
- Zhang, D. and Nunamaker, J. (2003). Powering e-learning in the new millennium: An overview of e-learning and enabling technology. *Information Systems Frontiers*, 5(2):207–218.

Appendix

Appendix A.

Code parts

In this section a number of code segments are presented which colleagues where interested in as they want to reuse them in their work on other courses. Only an overview of every code part is given as they are commented very extensive.

A.1. Limit the movement

The following function is used to update the elements of the simulation. The code example is from the Point Charges simulation, hence the elements which are updated are point charges. This function exists similarly in the two other extended simulations.

With this function the movement of the charge that's used for searching the hidden charges is limited. As described in this work this charge should only be movable between the inner and outer box. The simulation itself limits every movement to inside the outer box. Hence only the limitation to movement outside the inner box has to be implemented.

The box size of the outer box is 20 (unit less). The inner box is size of 16. The boxes are centered around the zero point of the environment. Hence the walls of the outer box are plus/minus 10 away from the zero point and the walls of the inner box plus/minus 8.

```
1 // !!! Sets the limits of the moveable region of elements !!!
2 // Updates the elements of the simulation.
3 // Is called while moving a charge.
4 // Takes the parameter boolean initial which is true when
  function called the first time.
5 private void updateScene(boolean initial)
6 {
```

Appendix A. Code parts

```
7 double maxDist = 20.0;
8
9 // Initial placement of all charges
10 if (initial == true)
11 {
12     engine.requestReorder( pc1 );
13     setChargePosition( pc1, maxDist );
14     setChargeAppearance(pc1);
15     for ( int i = 0; i < 5; i++ )
16     {
17         engine.requestReorder( hiddenCharges[i] );
18         setChargePosition( hiddenCharges[i], maxDist );
19         setChargeAppearance( hiddenCharges[i] );
20     }
21 }
22
23 // Movement of the visible charge that is moved by the
24 // student for searching the hidden charges.
25 // Restrict movement between inner and outer boxes.
26 // Box size outer box: +-10
27 // Box size inner box: +-8
28 // Movement is limited to inside the outer box from the
29 // simulation. So we only need to restrict the movement
30 // outside the inner box.
31 if ( pc1.shape.isSelected )
32 {
33     Vector3d pos = pc1.getPosition();
34     pos.add(mouse.getObjectTransform());
35
36     // Restrict the movement in x direction
37     if ( pos.x > 8 || pos.x < -8 )
38     {
39         pc1.setPosition(pos);
40         engine.requestReorder( pc1 );
41         setChargePosition( pc1, maxDist );
42         setChargeAppearance(pc1);
43     }
44     // Restrict the movement in y direction
45     if ( pos.y > 8 || pos.y < -8 )
46     {
47         pc1.setPosition(pos);
48         engine.requestReorder( pc1 );
49         setChargePosition( pc1, maxDist );
50         setChargeAppearance(pc1);
51     }
52 }
53
54 if ( mouse.getObjectTransform().length() > 0 )
55     hideDLIC();
```



```

53 mouse.clearObjectTransform();
54 }

```

A.2. Place elements on a grid

Placing an element on a grid means to only let them rest on a finite number of points inside the environment. From the programmers point of view, placing on a grid means rounding the position values depending on the grid size. In this case a grid size of one was implemented. That means the values are rounded to integers. This is done on the end of every move, when the mouse button is released. The charge then jumps to the next grid point. This is important for students as without a grid finding the exact position of a hidden charge is nearly impossible.

The first function rounds the position of the charge. But as it could be possible that a charge would jump on the place where another charge already rests after rounding the current position the new position is evaluated by the second function below.

```

1 // Every time a dummy is moved, on releasing the mouse button
  // the dummy jumps to the next grid position.
2 // The new position of the dummy is evaluated if there is
  // already a dummy placed.
3 // If the dummy is set on top of another dummy, it is moved to
  // the right of the previous placed dummy.
4 // The new calculated position is then again evaluated because
  // there could also be a dummy placed.
5 // This is done for all used dummies after a move.
6 private void roundPositions()
7 {
8     Vector3d curPos = new Vector3d(); // The current position
      // where the dummy should be placed
9     Vector3d evPos = new Vector3d(); // The position after
      // evaluation
10
11     for ( int i=0; i < posDummies.length; i++ ) //Loop over all
      // used positive dummies
12     {
13         curPos = posDummies[i].getPosition(); // Get current
          // position where the dummy should be placed
14         curPos.x = Math.round(curPos.x); // Place it on the grid in
          // x direction
15         curPos.y = Math.round(curPos.y); // Place it on the grid in
          // y direction

```

Appendix A. Code parts

```
16 while(true)
17 {
18     evPos = evaluatePos(curPos,i,1); // Evaluate the
        position
19     if ( evPos == curPos ) // If the current position was
        free, stop evaluating
20     break;
21     curPos = evPos; // If the current position is taken place
        the dummy on the new calculated position and evaluate
        again
22 }
23 posDummies[ i ]. setPosition( curPos ); // Place the dummy on
        the empty position
24 }
25 for ( int i=0; i < negDummies.length; i++ ) //Loop over all
        used negative dummies
26 {
27     ...
28     // Same code as for positive dummies
29     ...
30 }
31 updateBoxes(); // Update positions in the HIML host file
32 }
```

This function evaluates a position if it is empty or if there is already a charge placed. If the position is already taken the evaluated position is moved to plus one in x direction. This new position is then evaluated again.

```
1 // Evaluates a dummy position.
2 // Takes the position which has to be verified, the dummy id
    and the dummy charge.
3 // The dummy charge is important as the dummies are always
    created and placed in the simulation. Unused dummies are
    hidden and their charge is set to zero. It is possible to
    place a used dummy on an unused hidden dummy!
4 // The charge of the dummy has no effect on the field! It is
    only for displaying the correct color of the dummy!
5 private Vector3d evaluatePos(Vector3d newPos, int dummyId, int
    charge)
6 {
7     for (int i = 0; i < 5; i++) // Loop over all positive dummies
8     {
9         if ( i == dummyId && charge > 0 )
10            continue; // This case would be a comparison with itself
11            Vector3d posPos = posDummies[ i ]. getPosition();
12            if( newPos.x==posPos.x && newPos.y==posPos.y && posDummies[
                i ]. charge!=0 )
13                newPos.x += 1; // Move charge one step to the right
```

```

14 }
15
16 for (int i = 0; i < 5; i++) // Loop over all negative dummies
17 {
18     if ( i == dummyId && charge < 0 )
19         continue; // This case would be a comparison with itself
20     Vector3d negPos = negDummies[i].getPosition();
21     if( newPos.x==negPos.x && newPos.y==negPos.y && negDummies[
22         i].charge!=0 )
23         newPos.x += 1; // Move charge one step to the right
24 }
25 return newPos;
26 }

```

A.3. Place and remove dummies

In this section the code of placing and removing dummies is shown. The important thing is that the dummies always exist and are always placed somewhere in the simulation. By placing they are just set to visible and by removing they are set to invisible. The reason for this implementation is, that starting the simulation needs a little longer to load but when running it is faster as it does not have to create the dummy charges any more.

The first code part is the initialization of the positive and negative dummy elements.

```

1 ...
2 // Create new point charges elements for the dummies
3 posDummies = new PointCharge[5];
4 negDummies = new PointCharge[5];
5 int psn_count = 0; // Used for making dummies pickable for the
6 mouse
7 ShapeNode[] psndpn = new ShapeNode[10];
8 for ( int i = 0; i < posDummies.length; i++ )
9 {
10     posDummies[i] = new PointCharge();
11     posDummies[i].radius = 0.4;
12     posDummies[i].setMass(1.0);
13     posDummies[i].shape = new SphereNode( posDummies[i].radius ,
14         20, positiveChargeAppearance );
15
16     // At this point we just want to create the elements, not
17     // place them or make them visible!
18     posDummies[i].shape.setVisible(false);

```

Appendix A. Code parts

```
16 bg.addChild( posDummies[i].shape ); //This adds the element
    to the simulations environment
17 psndpn[psn_count] = posDummies[i].shape;
18 psn_count++;
19 }
20 for ( int i =0; i < negDummies.length; i++ )
21 {
22     ...
23     // Same code as above for positive dummies
24     ...
25 }
26 ...
```

With the following function a dummy is placed. That means a positive or negative dummy is turned visible and place it to the left or right border of the simulation.

```
1 private void addDummy(double charge)
2 {
3     PointCharge dummyCharge;
4     Point3d dummyPos = new Point3d();
5     dummyPos.z = 0.0; // z is always zero
6
7     if (charge > 0)
8     {
9         // If five or more positive dummies are placed do nothing
10        if ( numPosDummies >= 5 )
11            return;
12        dummyCharge = posDummies[numPosDummies];
13        dummyPos.x = 10; // Place the positive dummies on the right
            border of the simulation
14        dummyPos.y = -2 + numPosDummies; // Place the dummies next
            to each other
15        numPosDummies++;
16    }
17    else
18    {
19        // If five or more negative dummies are placed do nothing
20        if ( numNegDummies >= 5 )
21            return;
22        dummyCharge = negDummies[numNegDummies];
23        dummyPos.x = -10; // Place the negative dummies on the left
            border of the simulation
24        dummyPos.y = -2 + numNegDummies; // Place the dummies next
            to each other
25        numNegDummies++;
26    }
27 }
```

A.3. Place and remove dummies

```
28 // Set the charge for correct dummy coloring
29 dummyCharge.charge = charge;
30 // Set dummy position
31 dummyCharge.setPosition(new Vector3d(dummyPos));
32 setChargeAppearance( dummyCharge );
33 // Set dummy charge visible
34 dummyCharge.shape.setVisible( true );
35 }
```

The next function deletes, or better hides, all dummies. It just go through all the dummies and set them invisible. The dummy counters are set to zero. Finally all invisible dummies are placed at the zero point of the simulation. This is of course not necessary but especially for debugging it is always helpful to have a cleaned up simulation. With this function it can be seen why it is not possible to delete individual dummies. The dummies have no ID or anything else to identify theme selfs. So there is no system which would support the selection of one specific dummy. The reason is again to keep the code small and the simulation fast.

```
1 private void removeAllDummies()
2 {
3     numPosDummies = 0; \\ Set positive dummy counter to zero
4     numNegDummies = 0; \\ Set negative dummy counter to zero
5     // Loop over all dummies. One loop for positive and negative
6     // dummies.
7     for ( int i = 0; i < 5; i ++ )
8     {
9         posDummies[i].shape.setVisible( false ); //Just set the dummy
10        // invisible
11        negDummies[i].shape.setVisible( false ); //Just set the dummy
12        // invisible
13    }
14    clearDummyPositions(); // This function will place the
15    // dummies to the zero point of the simulation.
16 }
```


Appendix B.

Used Software and Hardware

For this work the following software was used:

- VirtualBox 4.3.8
- Vagrant 1.5.1
- Python 2.7.6
- GWT 2.6.1
- Git 1.9.4
- Eclipse Kepler SR2
- Windows 7 x64

The following Hardware was used:

PC 1:

- CPU: Intel i7 4x 3,2GHz
- RAM: 8GB
- Graphics card: NVIDIA GT620

PC 2:

- CPU: Intel Core2Quad 4x 2,4 GHz
- RAM: 6GB
- Graphics card: NVIDIA GT610

Appendix C.

Setup of a local edX test server

In this chapter the important steps of setting up an edX test server are described. For testing a local edX server was set up during this work. The main reason was the possibility of testing code easier and faster. Additionally some parts of the edX server code could be modified for debugging. During this work two versions of the edX server were used. While the code worked well on the newer server, some files of the older server had to be modified.

C.1. Setup of the edX test server - Version 2013

For this server the *mitxom-edx-platform-02sep13a.box* file was used. The manual can be found online at [edX website \(2014c\)](#). This server is used till today although it is old because it needs less RAM and so could run on older machines too.

With this server some problems with the *set_state* function occurred. The reason was that within this version the *jsinput* method was also older. Referring to the comments of the *jsinput* code it was tested only with a small example application since there were simply no courses using it at the time it was written. The problem was that *jsinput* was trying to call the *set_state* function which was, compared to the *set_state* function of the example application, way bigger and therefor needed longer to respond. So *jsinput* ran into a timeout and reported an error saying *set_state* is not working.

Appendix C. Setup of a local edX test server

For fixing this problem these three files of the edX server were modified:

```
/edx-platform/common/static/js/capa/src/jsinput.js
```

```
/staticfiles/js/capa/src/jsinput.js
```

```
/staticfiles/xmodule_js/common_static/js/capa/src/jsinput.js
```

These files are all placed on the edX virtual machine in */home/vagrant/edx_all/*.

The code line which was changed:

```
$(document).ready(setTimeout(walkDOM, 300));
```

It was changed from 300 to 3000.

Remark: Making this change locally on the test server the code wont run any more on any other edX server! This change was done just for debugging to see find the error. This code was not changed to run on an unmodified edX server since this problem was solved with newer edX versions. But as described, on older PCs this edX version is preferred and that is the reason this description is given.

C.2. Setup of the edX test server - Version 2014

For the new version of the edX server the *20140908-mitx-kifli-fullstack.box* file was used. The manual for this course can be found online at [edX website \(2014c\)](#).

With this version a problem with the structure of the course code occurred. In this newer version the edX courses which are created by edX studio had another structure then handwritten courses. This resulted in errors including files as edX could not find them. This happened only locally on the test server.

C.2. Setup of the edX test server - Version 2014

For solving this problem the following lines in the console had to be run in the console:

```
vagrant ssh – sudo disable_mongo_static
```

```
vagrant ssh – sudo restart web_loloadx
```

```
vagrant reload
```


Appendix D.

Questionnaires

D.1. Pre-Questionnaire

Pre questionnaire:

Name:

How old are you?

< 24	25 - 30	31 - 50	> 50
------	---------	---------	------

Highest finished school?

Do you use PCs at your work?

never	sometimes	often	always
-------	-----------	-------	--------

Do you use internet at home?

yes	no
-----	----

Do you use internet on mobile devices?

yes	no
-----	----

How many days per week do you use the internet at home?

1	2	3	4	5	6	7
---	---	---	---	---	---	---

What learning type are you?

auditory	visual	kinesthetic
----------	--------	-------------

How many years of computer experience do you have?

< 5	5 - 10	> 10
-----	--------	------

Do you have a physics education or a job in the field of physics?

yes	no
-----	----

Please describe the charge of the following two point charges and draw the field lines between them:



Please describe the charge of the following two point charges and draw the field lines between them:



D.2. Post-Questionnaire

The first ten questions of the post-questionnaire are based on the *System Usability Scale (SUS) Test* (Brooke, 1996).

Post questionnaire:

Name:

The first ten questions are based on the System Usability Scale (SUS) [1]:

1. I think that I would like to use this system frequently

1	2	3	4	5	

2. I found the system unnecessarily complex

1	2	3	4	5	

3. I thought the system was easy to use

1	2	3	4	5	

4. I think that I would need the support of a technical person to be able to use this system

1	2	3	4	5	

5. I found the various functions in this system were well integrated

1	2	3	4	5	

6. I thought there was too much inconsistency in this system

1	2	3	4	5	

7. I would imagine that most people would learn to use this system very quickly

1	2	3	4	5	

8. I found the system very cumbersome to use

1	2	3	4	5	

9. I felt very confident using the system

1	2	3	4	5	

10. I needed to learn a lot of things before I could get going with this system

1	2	3	4	5	

11. How important did you appreciate the instant feedback after the exam?

1	2	3	4	5	

12. How easy did you feel the learning of the influence of the point charges was?

hard	2	3	4	easy	

13. Do you prefer a standalone Software (TEALsim) or online experiments (edX)?

stand	2	3	4	online	

14. Do you prefer the original simulations without feedback or the extended simulation with feedback to your work?

original	2	3	4	extend	

15. Would you describe the system that you used as a simulation or a game?

Simulation				Game

16. Did you feel more like "Learning for an exam" or "Playing a game" during the evaluation?

Learning				Game

17. What did you enjoy most during your work with the simulations?

18. What did you dislike most during your work with the simulations?

19. Do you think that the extended simulations improve the learning process if you compare the standalone simulations with the extended simulations?

20. Which additional features would improve the extended simulations and the learning platform?

[1] Brooke, J. (1996). Sus-a quick and dirty usability scale. Usability evaluation in industry, 189(194):4-7.