

Sternig, Christof, BSc

Implementation and Evaluation of a Virtual Reality Learning Game for Mathematics

Master's Thesis

Graz University of Technology

Institut for Information Systems and Computer Media
Head: Kappe, Frank, Univ.-Prof. Dipl.-Ing. Dr.techn

Supervisor: Ebner, Martin, Assoc. Prof. PhD

Graz, September 2016

Statutory Declaration

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 used sources.

Graz,

Date

Signature

Eidesstattliche Erklärung¹

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.

Graz, am

Datum

Unterschrift

¹Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008; Genehmigung des Senates am 1.12.2008

Abstract

With Virtual Reality stated to be the next big thing, due to affordable head-mounted displays, like Oculus Rift, HTC Vive, etc., more and more companies are joining and experimenting with Virtual Reality. The fast evolution of mobile devices, especially mobile phones, with their increasing computational powers, built-in sensors and the broad availability, has built a new market for different applications. A new movement, the so called "Maker Movement", is rising and electrifying more and more people to create, invent and experiment new items with available technology. With the announcement of Google Cardboard, the marriage of these three subjects was achieved and opened a wide range of possible, cheap Virtual Reality applications, which can be created and used by everyone. This thesis shows the potential of combining making, gaming and education by implementing and evaluating a game prototype by 14 pupils aged 12-13. A simple Virtual Reality math game was created, using the Google Cardboard toolkit, to immerse into a virtual world. The aim of the game is to solve math exercises with increasing difficulty. The pupils were motivated and excited by immersing into the Virtual Reality of the game to solve exercises and advance in the game. After the evaluation the pupils tinkered headsets on their own to play the game again. The results of evaluation are very positive and show the high motivational potential of combining making and game-based learning and its usage in schools as educational instrument.

Kurzfassung

Mit der Entwicklung von leistbaren Virtual Reality Systemen, basierend auf sogenannten "head-mounted displays" wie der Oculus Rift, dem HTC Vive, etc. kam es in den vergangenen Jahren zu einem neuerlichen Virtual-Reality-Boom, dem sich mehr und mehr Firmen anschließen. Die kurzen Evolutionszyklen bei mobilen Geräten, speziell bei Handys, mit ihrer steigenden Rechenleistung, der Vielzahl an mitgelieferten Sensoren und der hohen Verfügbarkeit bieten eine Plattform für verschiedenste Anwendungen. Die "Maker"-Bewegung macht sich diese Technologien zunutze, um eigene Ideen umzusetzen, damit zu experimentieren und um neue Dinge zu erfinden. Diese Bewegung erfreut sich einer weltweit steigenden Beliebtheit. Als Google mit der Ankündigung von Google Cardboard "making", Virtual Reality und mobile Geräte miteinander verband, entstand eine, für nahezu jeden verfügbare Möglichkeit, Virtual Reality Anwendungen zu entwickeln und umzusetzen. Diese Arbeit zeigt das Potenzial, der Verbindung von "making", spielen und lernen anhand eines Lernspiels. Dieses Spiel wurde für Google Cardboard entwickelt und an einer Schule evaluiert. Bei dem Spiel handelt es sich um ein Virtual Reality Mathematikspiel, in welchem der Spieler oder die Spielerin Rechenaufgaben lösen muss, um das nächste Level zu erreichen. Die testenden Schüler waren begeistert als sie mit Google Cardboard in die virtuelle Welt des Spiels eintauchten und lösten motiviert die Aufgaben. Nach der Evaluierung bastelten die Schüler in mehreren Werkstunden ihre eigenen Headsets, um das Spiel zu Hause spielen zu können. Das durchwegs positive Ergebnis der Evaluierung zeigt, dass die Kombination aus "making" und spielendem Lernen äußerst motivierend auf die Schüler wirkt und es auch seine Berechtigung als pädagogisches Instrument hat.

Contents

Abstract	iii
1. Introduction	1
2. State of the Art	4
2.1. Virtual Reality (VR)	4
2.1.1. History	5
2.1.2. Virtual Reality Today	6
2.2. Head-Mounted Displays (HMD)	8
2.2.1. Google Cardboard	9
2.3. Video Games and Education	10
2.3.1. (Digital) Game-Based Learning	11
2.3.2. Gamification	13
2.3.3. Implementing Gamification	15
2.4. Maker Movement	18
2.4.1. History of the Maker Movement	19
2.4.2. Making in Education	20
3. Educational Games	25
3.1. Educational Flash Games	25
3.2. Mobile Educational Games	30
3.3. Educational Games for PC and Consoles	32
4. Prototype	36
4.1. Idea	36
4.2. Gameplay and Game Elements	37
4.2.1. Navigation	37
4.2.2. Level Structure and Items	39
4.2.3. Gameplay	40
4.2.4. Menus and Settings	45

Contents

4.3.	Implementation	47
4.3.1.	Actors and Components	48
4.3.2.	Processes	52
4.3.3.	Events	54
4.3.4.	Rendering System	57
4.3.5.	The Game Logic	62
5.	Evaluation	68
5.1.	Evaluation Setup	69
5.2.	Results of the Evaluation	71
5.2.1.	Statement 1	72
5.2.2.	Statement 2	73
5.2.3.	Statement 3	74
5.2.4.	Statement 4	75
5.2.5.	Statement 5	76
5.2.6.	Observations during Evaluation	77
6.	Discussion and Conclusion	79
6.1.	Discussion	79
6.1.1.	Gameplay	79
6.1.2.	Evaluation	81
6.2.	Future Research Directions	83
6.3.	Conclusion	84
Bibliography		86
A.	Sources	92
A.1.	Actor-Component-System	92
A.2.	Process System	94
A.3.	Event System	101
A.4.	Rendering System	107

List of Figures

2.1.	The sensorama simulator	6
2.2.	Sutherland's HMD and the tracking device.	9
2.3.	Different viewers for Google Cardboard.	11
2.4.	Connections of the different learning approaches.	13
2.5.	A pupil tinkering his cardboard headset.	23
3.1.	A screenshot of Math Lines 10	27
3.2.	An image of the game Flash Cards	28
3.3.	Level one of the game Bus Driver Math	28
3.4.	A screenshot of Math Man	29
3.5.	A screenshot captured when playing "Einmaleins"	31
3.6.	The action part of "Monster Numbers"	32
3.7.	The math challenges of "Monster Numbers"	32
3.8.	A screenshot of Minecraft	33
3.9.	Dr. Kawashima determines the age of the player's brain	34
4.1.	The avatar and the viewing frustum	39
4.2.	The four pickups of the game	41
4.3.	A screenshot from a game in progress	43
4.4.	The main menu	46
4.5.	The highscore screens	47
4.6.	A simple abstraction of the actor-component-system	49
4.7.	The actor package	51
4.8.	Actor components package	52
4.9.	Actions executed between two frames	53
4.10.	The package holding the process-classes	54
4.11.	The event system of the engine	56
4.12.	An image of the rendered Scene when playing with the VR headset	57
4.13.	An image of a sample scene graph	58

List of Figures

4.14. The render system	62
4.15. The game logic displayed as state machine	64
4.16. Flow diagram of the level loading process	66
5.1. Class picture of the pupils evaluating the prototype.	68
5.2. The cardboard headset used for evaluating the prototype. . .	69
5.3. The evaluation of the statements.	71
6.1. The headsets tinkered by the pupils.	84

List of Tables

2.1.	Available and announced VR headsets and the companies behind them (Kelly, 2016; Kuusisto, 2015; "Daydream - Google VR," 2016).	10
4.1.	The level properties for the five levels of the prototype.	44
4.2.	The specifications of the mobile device used to implement, debug and evaluate the game.	48
5.1.	The four different smileys used to rate the statements of the evaluation.	72
5.2.	Final ratings of statement 1.	73
5.3.	Final ratings of statement 2.	74
5.4.	Final ratings of statement 3.	75
5.5.	Final ratings of statement 4.	76
5.6.	Final ratings of statement 5.	76
6.1.	A list of the issues and suggested solutions identified during the evaluation and testing of the prototype.	82

Listings

4.1.	The rule for exercise generation	42
4.2.	The definition file for the avatar used in the game.	50
4.3.	The interface implemented by components which can be registered as listeners.	55
4.4.	The level of detail calculation in the MeshNode. Bigger distances lead to a bigger divisor and the rendered object is drawn with fewer triangles.	60
4.5.	The MeshNode method <code>isVisible</code> is using the frustum culler to determine if the object should be rendered or not because it is not in the field of view of the player.	61
4.6.	The level definition file of the first level in the game.	65
A.1.	The <code>AbstractActorComponent</code> defining the basic properties and methods for every specific actor component.	92
A.2.	The <code>ProcessManager</code> responsible for updating and managing the processes running during playing.	94
A.3.	The base class for every process able to run in the engines process system.	97
A.4.	The source code of the event manager.	101
A.5.	The base class every event has to inherit from to fit into the event system.	105
A.6.	A shortened source code of base class for every node the scene graph can contain.	107
A.7.	The shortened <code>Scene</code> class used to traverse the scene graph and render the scene.	110

1. Introduction

Since the dawn of Virtual Reality (VR) there have been major improvements in technology in recent years. With the evolution of VR headsets and the announcements of products like Oculus Rift, HTC Vive, Samsung Gear VR etc. more and more manufacturers have and will release their headsets to the market, to make Virtual Reality available to the public.

Nowadays, mobile devices are complex computers which are gradually replacing Personal Computers (PCs) in all-day life (Gartner Press Release, 2016). Due to their high availability, the variety of embedded sensors, cameras and the availability of high computational power, mobile phones especially lie in the focus of researchers and developers. Children are growing up with these technologies and are therefore used to mobile devices. They know how to interact with them and how to use them for surfing the internet, messaging, gaming, etc. (Grimus & Ebner, 2014). In the last few years a new movement, namely the maker movement, emerged, and is growing strong (Schön, Ebner, & Kumar, 2014). People all over the world invent, experiment and create and therefore are part of this movement. Pushed by small, affordable computers like the Raspberry Pi or the Arduino, offering new possibilities to invent and experiment with ideas and combine technologies. Makers meet at annual maker fairs to showcase inventions and experiments. When Google introduced Google Cardboard, making was combined with VR and mobile technology. Google Cardboard is a do-it-yourself Virtual Reality headset, consisting of a cardboard structure holding a smartphone. The smartphone is serving as the screen of the headset, to simulate Virtual Reality. This marriage of making, VR and a mobile device offers a great possibility to use concepts of digital game-based learning to create an immersive learning experience. Using VR techniques for learning, education or advanced training is an interesting research field with high potential.

1. Introduction

This thesis deals with learning in virtual environments and evaluates the integration of Virtual Reality applications in school by implementing a prototype of a math game. The research interest focuses on the following three research questions:

- 1. How could a VR-device be integrated into a mathematical learning scenario?**
- 2. Which lessons could be learned while evaluating this learning scenario in a secondary school class?**
- 3. How does the learning experience look like?**

To answer the above research questions, a simplistic Virtual Reality game is implemented, which should assist and motivate pupils in repeating already learned mathematical operations like addition and multiplication. The implemented prototype is evaluated to determine the need for such educational games and their possible motivational effects. School children of a secondary school class tested and rated the prototype and gave feedback about issues and positive aspects of the game. To combine the game-based learning approach with making, the pupils created their own cardboard headsets in extra lessons to be able to play the game with their own smartphones.

The structure of the work at hand divides into six chapters including this introduction. Chapter 2 introduces the history and state-of-the-art of the used technologies like Virtual Reality, head-mounted displays, digital game-based learning, project-based learning and the maker movement. The aim of this chapter is the explanation of key terms used throughout the paper, so readers can follow the meaning of this text easily. In the second chapter a few sample educational games for different platforms are outlined to get a feeling of existing educational games. Chapter 4 explains the game mechanics of the prototype and technical aspects of the implementation in detail. The first part of the chapter can be seen as a manual for the game, the second part describes the modules and concepts of the underlying game engine driving the game.

The game has been evaluated at a school by children aged 12 to 13, the process of the evaluation is described in the first part of chapter 5. A simple feedback system was defined, to enable the pupils to evaluate the prototype by rating five statements. The results of this evaluation process are

1. Introduction

summarized in the second part of the chapter. Finally, chapter 6 wraps up the impact of the math game on the pupils, the results of the evaluation and discusses positive and negative aspects of the prototype and the evaluation process. With the results at hand, further improvements of the shortcomings and possible extensions to the prototype are addressed and future research ideas are formulated.

2. State of the Art

After introducing the purpose and the aims of this work, this chapter defines some key words used throughout the text, so the reader understands the used concepts, their history and the state-of-the-art to better understand the upcoming chapters.

2.1. Virtual Reality (VR)

Virtual Reality is a computer generated illusion of a real world, in its perfection it tricks the human senses and mind to be indistinguishable from the real world it simulates. The goal of creating such a perfect world although is an utopian desire which cannot be achieved (Stanković, 2015, p. 4). Common sense defines VR by head-mounted displays like Oculus Rift and/or data gloves, but this is not a good definition for Virtual Reality (Burdea & Coiffet, 2003, p. 1). It can also be accomplished by projectors in combination with PCs called the CAVE (Cruz-Neira, Sandin, & DeFanti, 1993) and all sorts of computer games are creating a VR experience. The difference of VR from a 3D movie is the possibility to interact with the created world, change the state of the world and get a feedback about the effects (Stanković, 2015, p. 9). By providing the possibility of interaction, consumers of VR can immerse into and get a feeling of presence in the virtual world. According to Burdea and Coiffet (2003, p. 3), interaction and immersion are two of three key features of VR with the third feature being imagination. Virtual Reality is often used to simulate real world processes, the restriction of the parameters of the simulation, to fully map the extent of a real world process without breaking the simulation, often lies in the hands of VR developers and their imagination.

2. State of the Art

Another important definition, with respect to Virtual Reality, are Virtual Environments (VE). “VEs provide the illusion of presence in a place different from one’s current physical surrounding” (Stanković, 2015, p. 4). VEs can be roughly categorized according to users taking part in the environment and the degree of realism, therefore Stanković (2015, p. 10) differs between four different types of VEs:

- Single user, text only (2D) – i.e. text adventures
- Single user, realistic 3D – AAA games on modern console generations or the PC (a AAA game, pronounced “triple-A game”, is a high-budget game developed by a large game studio)
- Multi user, text only (2D) – social networks like Facebook
- Multi user, realistic 3D – Massively Multiplayer Online (MMO) games or general purpose VEs

The prototype implemented and described in chapter 4 tries to create the feeling of presence in the virtual world by using Google Cardboard and a mobile phone as head-mounted display. The implemented game creates a single user, three-dimensional virtual environment. After defining Virtual Reality and virtual environments the next chapter briefly outlines the history of VR.

2.1.1. History

Burdea and Coiffet (2003, p. 3) date the birth of Virtual Reality back to 1962, when Morton Heilig patented his “Sensorama Simulator” (Heilig, 1962). Figure 2.1 shows an image of the simulator described in Heilig’s patent. The purpose of the Sensorama was, to simulate a motorcycle ride through New York featuring a vibrating seat, a wind simulator, 3D video feedback and even an odor generator, which made it possible to smell the different odors of places the user passes. Although the simulator created immersion it did not feature interaction, one key feature of a modern VR experience, as previously defined. The first system including immersion and interaction was created by Lippman (1980), which featured a virtual representation of Aspen Colorado including three modes of display and free navigation. With sci-fi literature picking up on the topic in the 1980s, “the term Virtual

2. State of the Art

Reality was used in its present meaning for the first time by Jaron Lanier in 1989" (Stanković, 2015, p. 26). In the 1990s media got interested in the technology, the first CAVE, as stated previously, was created in 1992. The hype reached its peak with the implementation of Second Life, a massively multiuser online virtual community. Although media forecast VR as the next big thing, inventions like data gloves or head-mounted displays failed to enter the market. With the invention of mobile devices and social media popping up, VR systems were abandoned by media and public lost interest (Stanković, 2015).



Figure 2.1.: An image of the Sensorama simulator invented by Morton Heilig (Evens, 2007).

2.1.2. Virtual Reality Today

With every big company jumping onto the VR train again, Palmer Luckey started a successful Kickstarter campaign on August 1, 2012, for his VR

2. State of the Art

headset Oculus Rift. With his ideas proved that it is possible to create a high-end VR headset priced below \$300. Facebook acquired the company in 2014 for \$2 billion and showed the world VR is, again, the next big thing (Kuusisto, 2015). Many applications especially games created for Virtual Reality are fueling the desire to buy or use VR-headsets to enter virtual worlds. Before describing different head-mounted displays and the history of VR headsets, some applications of Virtual Reality are outlined.

In the past, complex VR systems like flight simulators or virtual battlefields for tactical combat training, used by the military, have been used mainly for training purposes rather than for entertainment. Today video games, with their complex three-dimensional worlds, are Virtual Reality systems with respect to interaction and immersion, generating big profits and are part of all-day life of many people. Interactive maps and virtual tours through museums or buildings are another interesting field for VR-applications, enabling the user to walk through a museum or visiting a place by selecting it via an interactive map in front of the computer or via a head-mounted display (Stanković, 2015). Virtual Reality is also used for educational purposes like training students in assembling complex mechanical machinery (Sportillo, Avveduto, Tecchia, & Carrozzino, 2015) or by creating a virtual chemistry and physics laboratory to enable students to learn under laboratory-like conditions (Ali, Ullah, Rabbi, & Alam, 2014; Daineko, Ipalakova, Dmitriyev, Giyenko, & Rakhimzhanova, 2015). In medicine Virtual Reality offers the possibility of simulating surgeries (Lo Presti et al., 2014; de Paolis, Ricciardi, & Giuliani, 2014), or to visualize 3D models of organs by superimposing their MRI or CT images, to create an effective way to examine the corresponding results (Ricciardi, Pastorelli, de Paolis, & Herrmann, 2015). The before mentioned applications give only a brief overview about research fields and possible usages for Virtual Reality systems.

This chapter described and defined Virtual Reality and showed some important aspects and applications concerning VR. The upcoming text outlines the evolution of head-mounted displays which are the definition of Virtual Reality in the common sense as stated previously.

2. State of the Art

2.2. Head-Mounted Displays (HMD)

In his paper from 1968, Sutherland introduced a prototype which today is known as the first head-mounted display. His invention included two cathode ray tube (CRT) displays, used to render a simple wireframe scene in front of the user's left and right eye respectively and an arm-like structure holding the HMD because of its weight and to keep track of the user's head position. Sutherland realized the importance of stereo vision to create the illusion of depth in the rendered images which creates the feeling of immersion, but pointed out that, "it is less important than the change that takes place in the image when the observer moves his head" (Sutherland, 1968, p. 757). To achieve head tracking a sensor was added to the HMD structure to generate view dependent images of the displayed scene to enable users viewing the scene from different angles. An image of the tracking device and the HMD structure is shown in figure 2.2 to get a feeling of the size of this first prototype of a head-mounted display. In the years after the presentation of his head-mounted display, devices to enable interaction with virtual scenes, the usage of force feedback to gain an even stronger feeling of immersion and tracking devices were tested and researched. Although Sutherland's paper was published 50 years ago, today's HMDs are using the same concepts as introduced by Sutherland, two displays used to create a stereoscopic view and head tracking enabling view dependent rendering of the virtual scene.

The biggest problem of VR and its systems were the missing applications, it was said to be a technology without a problem (Kuusisto, 2015). Although used for training purposes as mentioned earlier, HMDs were too expensive for mass production and the missing "killer application" (Kuusisto, 2015, p. 21) prevented the breakthrough for head-mounted displays and Virtual Reality. Maybe Luckey Palmer and his Oculus Rift finally get VR going. Table 2.1 shows a summary of current head-mounted displays, their availability and the companies standing behind them.

2. State of the Art

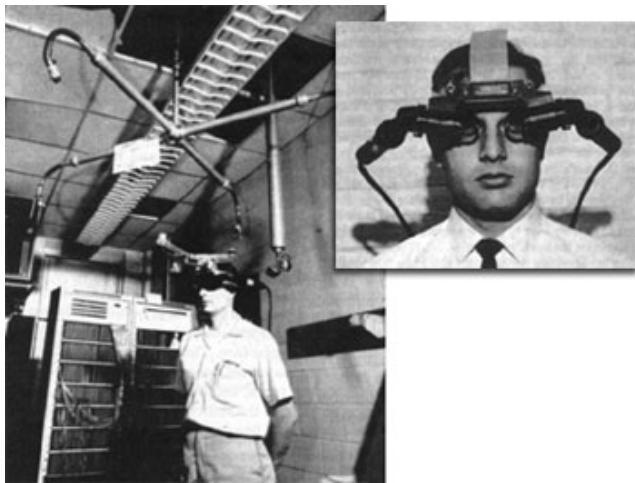


Figure 2.2.: Images of the head-mounted display invented by Sutherland. The left image shows the tracking device, the right image shows the head-mounted display with its two CRT displays (Evens, 2007).

2.2.1. Google Cardboard

As mentioned in the introduction of this work, computational power of mobile devices is growing year-by-year and the equipment of different equipment and sensors like cameras, GPS, Bluetooth, etc. are used to create different kinds of applications. With the built-in tracking sensor of modern mobile devices it is possible to query the device's position and orientation which in turn is needed to use the smartphone as a stereoscopic screen for VR applications. On June 26, 2014 Google introduced Google Cardboard, a simple cardboard headset construction enclosing a mobile phone, at Google I/O bringing Virtual Reality to Android devices (Lardinois, 2014). Originally, the idea emerged from the 20 percent project (Google employees are given one weekday to dedicate it to their very own projects) of David Coz who wanted to watch a stereoscopic YouTube video, but could not because of a missing VR-headset. He realized that a makeshift headset wrapped around a mobile device in combination with its sensors available to recognize the head's position would do the trick. After building the prototype, Google decided to make an own VR project named Google Cardboard (Metz, 2015).

2. State of the Art

Company	Name	Available
Facebook	Oculus Rift	Now
HTC	Vive	Now
Sony	Playstation VR (Project Morpheus)	October 2016
Google	Cardboard	Now
Google	Daydream	Fall 2016
Samsung	Gear VR	Now
Consortium of Companies (Intel, Razer, etc.)	OSVR	Fall 2016

Table 2.1.: Available and announced VR headsets and the companies behind them (Kelly, 2016; Kuusisto, 2015; "Daydream - Google VR," 2016).

To use Google Cardboard Apps and play VR games on the mobile phone two components, a viewer and a smartphone are needed, as stated earlier. A multitude of viewers can be purchased online covering different price categories or one can create a headset by using the construction plans available on the official Google Cardboard homepage (<https://vr.google.com/cardboard>, last visited 23-09-2016). A small sample of different viewers is shown in figure 2.3. Software development kits (SDKs) for Android, Unity and iOS are also available on the official site to develop own Virtual Reality Apps for mobile phones ("Cardboard - Google VR," 2016). This Google project is mentioned in this section specifically because it is the basis for the VR-headset, the pupils tinkered in school according to the construction plans, to play the designed prototype described in chapter 4.

2.3. Video Games and Education

Video game industry has grown big over the years, with the evolution of games and game mechanics, realistic graphics and sounds, more and more people are playing video games. The Entertainment Software Association (ESA) releases a yearly report, showing essential facts about the computer and video game industry. The current report points out, that the average American gamer is 35 years old (Entertainment Software Association, 2015),

2. State of the Art



Figure 2.3.: The image shows a sample of different viewers to play VR games or use the existing VR applications available in app stores ("Get Cardboard - Google VR," 2016).

in 2005 the average gamer's age was 30 (Entertainment Software Association, 2005). 51% of American households own a dedicated game console and 42% are playing video games on a regularly basis (three hours or more per week) (Entertainment Software Association, 2015). With this statistics increasing on a yearly basis, using video games for educational purposes gets an interesting industrial factor. Kapp (2012) reviews researches and studies about the effectiveness of games and game elements used in a learning context and concludes, that learners can benefit from such technologies, when implemented and presented in the right way.

This section explains some key terms like digital game-based learning and gamification and their use in learning applications.

2.3.1. (Digital) Game-Based Learning

The term digital game-based learning (DGBL) describes the process of learning while playing a computer game (Prensky, 2007) and can be seen

2. State of the Art

as a specialization of the umbrella term game-based learning (GBL). GBL focuses on learning by playing any kind of game, i.e. board games, card games, video games, etc. match the definition of "game" in terms of GBL.

Since the creation of the first video games in the 70s and 80s, the idea of learning by playing computer games was present. With the evolution of the game market, from a small industry for young men interested in technology, to a billion-dollar industry, more and more studies about learning in combination with video games emerged. From the early 2000s the two terms "Serious Games" and "Digital Game-Based Learning" replaced the term "Edutainment", short for education and entertainment (Breuer, 2011).

The term serious game, defined by Abt in 1971, describes all games with an educational purpose without the intention to be played only for amusement (Abt, 1987). Digital game-based learning should feel like playing a computer game without recognizing the learning effect or discovering the content to be taught by the game (Prensky, 2007). It "is any marriage of educational content and computer games" (Prensky, 2007, p. 145). Breuer (2011) points out, that DGBL and serious games as defined beforehand, have common elements as shown in the following enumeration, with serious games going beyond when for example being used as distraction for painful therapies. Although studies and researches differentiate between DGBL and serious games, the aim of both concepts is, to use games for an educating purpose going beyond mere entertainment. Breuer lists six common features of DGBL and serious games:

1. **Interactivity** – learning by doing and experimenting.
2. **Multimedia-based** – visualizing/preparing content and feedback by using 3D models, audio etc.
3. **Involvement** – the game should be fully engaging to keep the player from distractions.
4. **Challenge** – increasing difficulty but beginner friendly, the game should always challenge the individual skills to keep players motivated.
5. **Reward** – rewards and feedback of progression should push self-efficacy and motivation.
6. **Social Experience** – provide communication channels to connect players.

2. State of the Art

Considered as sets, digital game-based learning is a proper subset of serious games and game-based learning and further of entertainment education or short “Edutainment”, which stands for all kinds of educational content enriched with entertaining elements, i.e. interactive museums. E-learning on the other hand delivers learning content via and through electronic or digital media, for example videos on the web, digital slides or audio files. By adding different grades of game-like elements to the content, e-learning intersects with Edutainment and its specializations, but it can stand on its own without adding entertainment. Serious games fully contain DGBL and have much in common with GBL but can go beyond as mentioned earlier (Breuer, 2011). Figure 2.4 shows the sets of the different concepts and their connections.

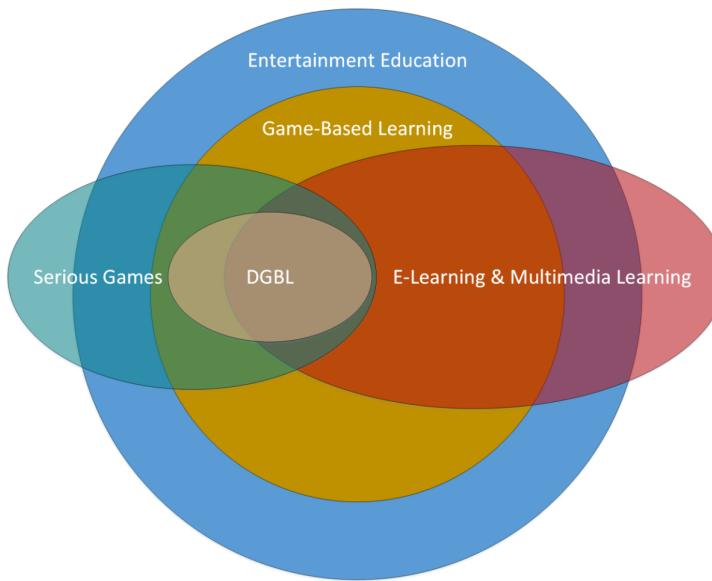


Figure 2.4.: The different learning approaches displayed as sets and their connections (Breuer, 2011).

2.3.2. Gamification

According to Deterding, Dixon, Khaled, and Nacke (2011) gamification has been a contested term since its first appearance in 2008 and there have been

2. State of the Art

several definitions of gamification (Kapp, 2012). With the lack of academic attempts to define gamification, Deterding et al. (2011) mentioned two ideas current usages of the word are fluctuating between. “The first is the increasing adoption, institutionalization and ubiquity of (video) games in everyday life” (Deterding et al., 2011, pp. 9-10). The second idea is the usage of game design and game elements in non-game contexts or products, adding enjoyment which in turn is leading to engagement, motivation and duration. Deterding et al. (2011) define gamification as “the use of game design elements in non-game contexts” and thus favor the second idea stated before.

Kapp (2012, p. 10) defines gamification as the usage of “game-based mechanics, aesthetics and game thinking to engage people, motivate action, promote learning, and solve problems”. This definition of gamification is a more suitable one than the definition by Deterding et al. (2011) with respect to this work, because it fits the implemented prototype, its mechanics and methods to convey the learning content. The implemented learning game “VR-Matherallye” as described in chapter 4 uses well-known game-based mechanics like collectible power-ups to increase time, points, etc., a platform the player must not fall off and different styled levels. The fact that it is played via head-mounted display, to enter the virtual world of the game, fully engages players and motivates them to explore the world and its elements. By solving math exercises of increasing difficulty to advance to the next level and score more points, players are kept motivated to play on and therefore learn while playing.

The chosen definition for gamification contains several key terms like game-based mechanics, engagement, motivation or problem solving. By enriching learning contents by different levels of these concepts, different types of gamification can be created. This is important when creating and implementing applications using gamification, which will be discussed in the next section. It will define different types of gamification and the key elements of educational applications to get a better understanding of how the concept of gamification can be used to create an engaging learning experience.

2. State of the Art

2.3.3. Implementing Gamification

Creating learning applications using game elements, gamifying existing systems and contents or designing a digital game-based learning application from scratch is a challenging task. It is important to balance game mechanics and educational content properly to keep players motivated and engaged. The definition of gamification showed some key terms, which have an impact on the look-and-feel of the application at hand. Applications using different kinds of game-based elements with subtle or rare learning moments can feel like ordinary computer games on the one hand, educational content enriched with some game-based elements like highscores, achievements, etc. on the other hand look like a learning website for example. To categorize this variety of applications, Kapp, Blair, and Mesch (2014) define the term interactive learning event (ILE) to combine these different types of gamified applications under one umbrella term. To create a successful ILE Kapp et al. (2014) list four important points to keep in mind:

1. Game-based (fun) elements and instructional, non-entertaining elements should co-exist and grow together to keep the right balance.
2. Interactivity ensures player engagement, player engagement results in learning more and keeping the knowledge for a longer time.
3. Create a good story, the better the story and the story-telling, the more the player is engaged. Story elements and learning goals should be linked for a better learning experience.
4. Proper testing needs to be done in order to evaluate the ILE and its elements.

When ready to implement an interactive learning event, keeping the goals and important steps mentioned before in mind, Kapp et al. (2014) distinguish between three main types of ILEs, namely game, gamification and simulation. In the context of interactive learning events, gamification gets a new meaning and should not be mixed with the previous definition. The three types or categories of ILEs are described and defined in the following sections.

2. State of the Art

Game

A game could be broken down to activities, like matching game items to other game items, collecting objects to progress in the game, solving puzzles, exploring the game world etc. Most games contain a set of these game activities, the user executes to progress or finish the game. For every game, a well-defined game-space exists, the user is interacting within, to reach new checkpoints, landmarks, goals or the end of the game, which is called the winning-state of the game. With the existence of a winning-state, the players know if they, or other players in case of a multi-player mode, have won the game (Kapp et al., 2014).

By interacting with the current game state, the player changes the current state and gets feedback about the changes made and visually recognizes the effects, due to changing or moving game objects. By trial-and-error the user can alter the game-state until the winning-state is reached. While interacting with the game, the user learns which state is favorable and which one is not, wrong or disadvantageous states can be altered to reach a more suitable state. This process of interaction, feedback, decision making and evaluating is known as the game cycle defined by Garris, Ahlers, and Driskell (2002).

In terms of ILE games, two different types can be distinguished, testing and teaching games. In a testing game, already present knowledge of the players is queried over and over again to gain a learning effect through repetition. Teaching games on the other hand are building up knowledge by giving how-to instructions about upcoming game activities or challenges, so the users can succeed by using the previously gained knowledge (Kapp et al., 2014). Combination of these two concepts are possible too, knowledge gets build by instructing the player, this knowledge is then queried in different tasks in the game to repeat and internalize the previously taught contents.

Gamification

In ILE-context, gamification is the usage of game elements to motivate and engage learning people, to work through educational content to achieve their learning goals. Gamification could be achieved by just using one game element like badges, achievements, challenges, etc. or a set of such

2. State of the Art

game elements. For example, users of the gamified content get different achievements or badges for working through different parts of the learning content. The collected badges can be viewed and compared with other users to compare the learning progress, which in turn leads to a motivational effect to go on and earn new badges. Gamification does not offer a game cycle, where users interact and change the state of the application, it merely “uses parts of games but is no game” (Kapp et al., 2014, p. 56).

As with games, two types of gamification exist, namely structural and content gamification. Structural gamification is used to enrich content with game elements to guide the user through the learning content without changing the content itself. Content gamification changes the learning content by using animations etc. to create a game-like experience (Kapp et al., 2014).

A gamification ILE should be used to motivate learners in progressing through educational content, an example of this kind of ILE is the website Code School (www.codeschool.com, last visited 23-09-2016). This website offers online programming courses consisting of slides, videos, interactive programming challenges, badges and a scoring system to track the learners progress and therefore is a great example of how to use game-like elements to gamify content.

Simulation

A “simulation is realistic, controlled-risk environment where learners can practice behaviors and experience the impacts of decisions” (Kapp et al., 2014, p. 58). A simulation maps reality in an accurate manner into a virtual game-like environment. It is used for training purpose to remove the real world risks for the trainee. Crashing an airplane in the simulator does not harm the trainee nor the trainer as a crash in the real world would do. The learner is using previous knowledge or learned actions to succeed in the simulation. Failure results in feedback, which in turn causes the user to learn from wrong actions. The main purpose of simulations is therefore the testing of knowledge by challenging the user with exercises.

2. State of the Art

A simulation and a game have many elements in common. Interaction leads to feedback and a change in behavior of the user as stated before as the game cycle. There could be highscores, achievements or multi-player elements as within games. The main difference is the realistic setting of simulations. The more realistic a simulation is the better. Learning to fly an airplane should be as realistic as possible and small mistakes should affect the virtual airplane like a real airplane. Whereas in games flying an airplane is a simple action manageable by every player without further knowledge.

As stated before, simulations should be used to query previous knowledge. Instructors should teach the necessary content to be successful in the simulation and challenge the trainee with different exercises (Kapp et al., 2014).

Classifying the Prototype

According to the above taxonomy, the prototype implemented for this thesis is a testing game. The game queries knowledge, the player already knows or should know. It can further be seen as a matching game with the goal of finding the right answer, or better, solution to the current equation. According to Kapp et al. (2014, p. 49) using “a testing game to teach, the key is to add repetition” and further by “getting an answer wrong, learning the right answer through feedback, and then repeating the process until all the answers are right”, learning can be achieved by using the testing approach. The previous knowledge of the player is a factor for the time being needed to meet the learning goals. Lesser knowledge results in a longer repeat-error-feedback cycle, until the all goals are reached (Kapp et al., 2014). Due to the lack of feedback after wrong calculations, the prototype in its current state is not suitable for a learning-by-testing approach.

2.4. Maker Movement

With today's availability of cheap technology and tools like 3D printers, personal computers (PCs), tiny computers like the Raspberry Pi, the Arduino or even further the Internet of Things (IoT), connecting all sorts of things

2. State of the Art

with the internet, inventing and creating, making, can be done by everyone. Magazines like "Make", FabLabs, conferences and fairs made "Making" popular and more and more people are joining this maker movement. The maker movement stands for creating and developing new things using the before mentioned technologies. This can happen in fabrication labs (FabLabs), Makerspaces or on the desktop at home (Schön et al., 2014). This section outlines the history of the maker movement and shows how making and education can and should be combined to create a new learning experience, not only for children, but students and adults.

2.4.1. History of the Maker Movement

With the invention of microcomputing back in the 1970s, technology hobbyists invented the personal computer and made it popular. Social clubs like the Homebrew Computer Club were founded all around the world to share knowledge, technology and inventions. In 1985 the MIT Media lab was founded with the purpose of learning by doing. The idea was to offer expensive technology to hobbyists and makers to realize their ideas. The lab was a big success and other universities followed the example and created their own labs. Makers graduating from MIT's media lab founded own companies, invented new products to fuel the young maker movement. The biggest invention of the media lab was the invention of new, other labs, to invent new items or tools to be used by makers for creating their products, the so called fabrication laboratories (Libow Martinez & Stager, 2013).

The tools and the space offered by FabLabs, like laser cutters, 3D printers etc., are used to realize and manufacture own products but also create tools to be used to create new products. Gershenfeld (2005) points out that these FabLabs can become smaller, to fit on everyone's desktop. Students working in these labs have invented numerous of personal things and systems on their own, combining different disciplines like technology, arts, etc.

In 2005 the magazine MAKE was published the first time, issued on a two-week-basis. The magazine introduces makers and their projects and guides through the development of own projects. In 2006 the first Maker Faire took place in San Mateo, offering a stage for makers and people interested in making (Schön et al., 2014). Today maker fairs spread around the world,

2. State of the Art

from April 16-17 2016 a maker faire took place in Austria the first time. In his manifesto, Hatch lists nine principles, make, share, give, learn, tool up, play, participate, support and change, defining the maker movement, but he also states, that these principles are there to be altered, rewritten or extended by anyone. "That is the point of making" (Hatch, 2014, p. 2).

With this brief history about making the next section shows the links between making and education, as these two disciplines are connected strongly.

2.4.2. Making in Education

According to Libow Martinez and Stager (2013), Seymour Papert can be seen as the father of today's maker movement. He wanted to revolutionize education in using modern technologies like computers not only for problem solving, as they are and have been used, but for creating some action. In the paper "Twenty Things to Do With a Computer" these previous mentioned actions, like composing music, programming, etc. are described and shown (Solomon & Papert, 1971). But Papert not only showed how to use technology for better educational methods, he was a maker himself. He invented the programming language Logo, to teach kids the concepts of programming. He also worked on the first programmable robotic construction kits for LEGO (Libow Martinez & Stager, 2013). He coined the term "Constructionism" which means that learners gain knowledge by using tools, to realize their ideas (Papert, 1986).

With the invention of computers, the desire of using them, to teach schoolchildren was evident. But the first computers were big machines, programming was done by experts. Adults, apart from experts, did not know how to program these complex computers, so how teach it to children? With the previously mentioned invention of Logo by Seymour Papert, things changed and schools used Logo to teach programming and mathematics to pupils. The same is true for the maker movement. As stated before, FabLabs teach how to use technology to create new tools, technology or products. Policy recognized the importance of making, and curricula of schools were expanded to teach these new working skills, but eventually, in 1999 this skill-based approach was abandoned due to the rapid evolution of technology. The aim

2. State of the Art

is to teach about technology and how it works rather than teaching skills how work with specific items (Blikstein & Krannich, 2013).

But Halverson and Sheridan (2014, p. 498) point out that “learning in making is, emphatically, not interchangeable with schooling”. With the invention of FabLabs and adaptions for schools the learning focus is placed on “principles of engineering, robotics, and design”. Researches show how to rebuild a classroom to make it a makerspace, to enable schoolkids experimenting with tools and technology to create. But there has not been found a consensus yet, how making should fit into existing curricula or if it should replace those curriculas. How should testing and grading look like, as making combines the learning of different skills like mathematics, physics etc., but also leadership (Thompson, 2014). Makers fear, that institutionalize making through school programs “will quash the emergence, creativity, innovation, and entrepreneurial spirit that are hallmarks of the “maker revolution”” (Halverson & Sheridan, 2014, p. 500). According to Halverson and Sheridan (2014) “the great promise of the maker movement in education is to democratize access to the discourses of power that accompany becoming a producer of artifacts, especially when those artifacts use twenty-first-century technologies”. Learning through making has the potential to reach the institutional and policy goals for learning in science, technology, engineering and mathematics (Halverson & Sheridan, 2014).

Project-Based Learning

Closely related to making is the concept of project-based learning (PBL), which was introduced by John Dewey in the 1890s. He noticed the importance to integrate real-world problems and situations into the contents taught in school. Project-based learning evolved and developed over the years and there are exist many definitions and different interpretations of the term (Habok & Nagy, 2016). Markham, Larmer, and Ravitz (2003, p. 4) define PBL as “a systematic teaching method that engages students in learning knowledge and skills through an extended inquiry process structured around complex, authentic questions and carefully designed products and tasks”. PBL has become more popular but faces challenges as how to integrate it to school lessons and how teachers and students should

2. State of the Art

handle this form of schooling. Studies showed that project-based learning results in high engagement of students or pupils and also teaching how to plan, organize a project and communicate with others about aspects of the project. Although project-based learning combines many positive aspects of teaching and learning, it is not suitable for learning basic skills. These basic skills or knowledge like mathematics, writing, reading, etc. needed to realize the project have to be taught beforehand (Markham et al., 2003). This work kind of used an inverse approach to PBL. The described math game was implemented and evaluated in the school. After the evaluation a tinkering project was realized with the evaluating children to create their own cardboard headsets to play the prototype on their own. After the evaluation the pupils were highly motivated and everyone managed to realize the complex tinkering project. Figure 2.5 shows a final cardboard headset, work in progress and a pupil tinkering his cardboard headset.

Even though making has not yet been integrated in school lessons, making with children is becoming more and more popular. There are workshops, maker fairs and open labs where adult instructors are motivating children to create and implement their own ideas. In the book *Making-Aktivitäten mit Kindern und Jugendlichen: Handbuch zum kreativen digitalen Gestalten*, Ebner, Schön, and Narr (2016) defined six principles to differentiate making activities with children from other projects or leisure time activities.

- The children are the main actors of the making process, they develop an idea and create, implement and produce the final item.
- The result of the making process is an actual product, which can be used after the work is finished.
- Making should support the development of creativity of the children.
- To create their products, children are learning how and where they get the necessary skills to successfully implement their ideas on their own. This is called self-organized learning.
- The process of making also leads to communication and knowledge exchange between children working on different products or ideas.
- Making activities, in the best case, are offering possibilities to actively change, improve and shape the world. The focus lies on principles like recycling, sustainability or social engagement.

The handbook introduces important concepts of how to build a maker space

2. State of the Art

for children, which tools to use and how to teach it to the children. It offers a variety of making projects which can be done by the children including road maps for teachers to introduce the projects to the pupils. It also includes the construction plans for the cardboard viewer which was tinkered by the pupils which evaluated the prototype of the implemented math game (Ebner et al., 2016).



Figure 2.5.: The image shows the work in progress and a pupil tinkering his VR headset during special tinkering lessons. The left sub-image shows the different cardboard parts of the headset which were composed to form the final headset shown in the topmost sub-image.

This chapter outlined the history and the state-of-the-art of the technologies used to create the prototype of the implemented math game described in the upcoming chapters of this thesis. The introduction of Google Cardboard, combining making and broad available mobile technology to form a cheap head-mounted display, opens new possibilities to immerse into virtual realities without using expensive, state-of-the art hardware. The interactive

2. State of the Art

learning event “VR-Matherallye” has been classified as a testing game, with respect to the described key concepts of digital game-based learning and gamification. The chapter also discussed the importance and positive effects of project-based learning in schools as well as the possibilities when integrating making in school lessons. The next chapter will discuss and show some examples of existing educational games.

3. Educational Games

The previous chapter outlined and defined the terms gamification, game-based learning and the key elements of these concepts. This chapter of the thesis is about available educational games which can be found on the internet, for mobile devices or are available for the PC or consoles. The first part describes educational browser games which can be found on various website when searching for "educational games". The second part shows educational games available for mobile devices and the final part of this chapter is about adapted PC "AAA"-games and console games to fit the term educational game.

3.1. Educational Flash Games

Flash games are better known as browser games and can be found in various forms all over the internet, they can be played even via social networks and allow to connect with other people. This section will list and describe some sample applications from the website www.learninggamesforkids.com (last visited 23-09-2016), which offers different kinds of educational games for preschool and elementary school kids for free. The site is well structured and lists the different game categories available in the left side menu, example categories are:

- **Health Games** – The games available in this category teach kids about anatomy, allergies or how to stay fit.
- **Vocabulary Games** – Kids learn about the structure of words, vocabularies of foreign languages or about synonyms in the available games.

3. Educational Games

- **Geography Games** – The games teach about the different continents and their countries.
- **Alphabet Games** – This category offers games for every letter of the alphabet.
- **Math Games** – Math games is subdivided in different specializing categories like Addition, Subtraction, 1st grade math, etc.

The educational game implemented for this thesis is a math game and therefore the presented math games are all taken from the “Math Games”-category to compare the ideas and approaches of the different games with the prototype. Four games have been chosen, the first game is about learning addition, the second and third one are about addition and multiplication and the fourth game is a Pacman-like approach to learn the basic calculation operations.

Math Lines 10/20

The aim of this game is to add bubbles with different values together to form a chain summing to 10 or 20 respectively. Bubbles are generated randomly, following a path circling around the center. If the chain of bubbles reaches the end of the path, the game is over. To shorten the chain, the player has to shoot bubbles with different values into the moving chain to create a sub-chain of bubbles summing up to 10/20. Aiming and shooting the new bubbles is done by moving and clicking the mouse. Image 3.1 shows a screenshot of the first level of “Math Lines 10”.

Flash Cards

Flash Cards is available in different forms, “Flash Cards: Add Five” generates exercises where five is added to a random digit, “Flash Cards: Add Double Digits” is about adding double digit values, etc. The Flash Card games are also available to practice the different multiplication tables. The game itself is a simple exercise generator displaying the different calculations following the specific restrictions. The player has to type in the correct result and push the enter button on the keyboard to advance to the next

3. Educational Games



Figure 3.1.: The image displays a screenshot of the addition game “Math Lines 10” where the user has to build sub-chains of sum 10.

calculation. After six correct solved calculations the player advances to the next level. A screenshot of the game is displayed in figure 3.2.

Bus Driver Math

Again this game is available in the addition and multiplication category. The player is the bus driver checking the money he gets from the passengers entering the bus. There are three types of passengers, kids, adults and retirees which have to pay different amounts of money for their bus ticket. The passengers are entering the bus one after another and put money into a cash box. The coins thrown into the box are displayed on the screen and the player has to calculate the sum of the coins and decides if the current passenger has paid enough or not. If all correct paying passengers entered the bus the player advances to the next level. Figure 3.3 shows a screenshot of the first level of the game.

Math Man

The last math game introduced is Math Man, which is a Pacman-like game where the player collects points by eating them with the avatar, looking

3. Educational Games

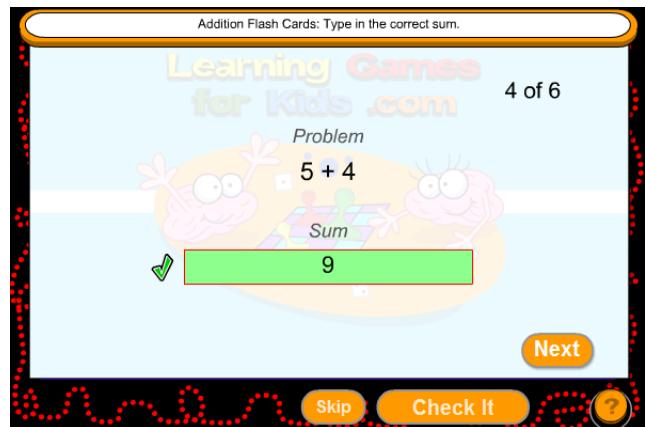


Figure 3.2.: An image of the game Flash Cards: Add Five, were the player has to solve exercises of type $5 + n$, where n is a random digit, by typing in the correct result into the input field.



Figure 3.3.: The screenshot shows a passenger entering the bus and the player has to decide if the paid amount was correct or not.

3. Educational Games

like a circle with a mouth. There are enemies, displayed as ghosts, which have to be eliminated by the player to advance to the next level. Every ghost presents a digit which is the result of a calculation the player has to solve. To display one calculation, the player has to navigate the avatar to a “?”-bubble, which shows a calculation. This exercise has to be solved and the ghost with the calculated result has to be eaten. Navigation of the avatar is done via the cursor keys of the keyboard, a ghost is eliminated by moving the avatar into the ghost. If the player tries to eat the wrong ghost one life is lost, if no life is left, the game ends. The adapted Pacman-math game is shown in figure 3.4.



Figure 3.4.: A screenshot showing the adapted Pacman-game to solve exercises and eliminate the ghosts standing for the correct results by eating them.

With this four games the first section of this chapter is completed. All described games try to teach and help children in learning adding and multiplying, with math man also adding divisions and subtraction exercises. Game one, “Math Lines” and game four, “Math Man”, can be compared with the implemented prototype because they add game-like elements like matching bubbles or moving an avatar through the game world. “Flash Cards” is a simple calculation generator and is not augmented with game elements, the game “Bus Driver Math” shows good looking graphics and adds a kind of story to the game but is in its core elements like “Flash Cards”. Therefore these two games cannot be compared with “VR-Matherallye”. The next section shows and outlines educational games for mobile platforms.

3. Educational Games

3.2. Mobile Educational Games

Apps for mobile devices are offered in the corresponding app stores, where all kinds of applications can be found. As with the browser games described in the previous section, searching for “educational games” in Google’s play store lists many learning games for children of all ages. According to Adkins (2016) the market for mobile learning applications is growing strong in the next five years due to the acceptance of mobile game-based learning. This section will outline two mobile math learning games tested and played on a mobile phone.

Einmaleins (Multiplication Tables)

This math game offers four different modes to train the different multiplication tables. In the first mode the player has to solve ten exercises in a row to complete the level and the next level is unlocked. The level has a time limit which is decreased if one calculation was wrong, if the exercise was solved correctly, time is added to the clearing time for the level. The second play mode challenges the player with exercises, a correct calculation adds time and the player is rewarded with points, a wrong calculation decreases the time. If the time is up, the current score can be saved. This mode ends if the time is up, unless there is time on the clock, the player is challenged with new exercises. Playing mode three is a multi-player mode, where two players are solving exercises on one device. The fourth mode offers a list with exercises the player can choose from to learn specific multiplication tables. As shown in screenshot 3.5 for every calculation four answers are offered with one being the correct result. The symbols between the equation to solve and the possible answers are jokers, the player can use to ease solving the exercise. The jokers are from left to right: get a new exercise, slow down the time, fifty-fifty-joker and the last one solves the equation correctly.

3. Educational Games



Figure 3.5.: A screenshot of the first playing mode of “Einmaleins”, the green bar at the top displays the remaining time, jokers can be found in the middle of the screen between the equation to solve and the possible answers.

Monster Numbers

In “Monster Numbers” the player has to help squirrel Tob to repair its spaceship to get away from the planet it stranded. The game mixes action elements with math to overcome obstacles and collect the missing parts of the space craft. The story mode is divided into six areas, where every area is subdivided into different levels challenging the player with different exercises. In action levels, Tob runs through the side scrolling world and has to jump over gaps and obstacles by swiping over the screen. In math challenge levels the squirrel faces exercises which have to be solved by the player to get by enemies. Tob’s friends in the game are monsters looking like digits which are helping him to get through the levels, also his enemies are digit-lookalikes. The game adapts its difficulty according to the players age. Figure 3.6 shows an action level of the game, in image 3.7 the math challenges are displayed.

The second game introduced in this section tries to merge gaming for entertainment and learning by using separate action levels and math challenge levels. The first game is a quiz-like game where the player only solves exercises for points or to get to the next challenges. Both games include ele-

3. Educational Games



Figure 3.6.: This screenshot shows the side scrolling levels which are the action parts of the game. The player has to avoid obstacles by jumping over or sliding through beneath them.

ments used by the math game implemented for this thesis but the prototype merges the math challenges with the gameplay in a more fluid way and does not separate action from learning.



Figure 3.7.: In the math parts the player has to tries to solve the exercise correctly to overcome the obstacle.

3.3. Educational Games for PC and Consoles

Educational games are available also for the PC and different consoles, especially Nintendo offers learning applications for its Handheld Nintendo

3. Educational Games

DS consoles. A few PC games have been adapted and published as “Edu”-versions to act as learning games for children and students. Examples are “KerbalEdu” which is a sandbox game-based on a realistic physics model where the players can learn about the connections of different physical laws, “SimCityEDU PollutionChallenge!”, teaches players to build a green city or the educational version of the game “Minecraft”, which will be described in the next passage. This section also describes the game “Dr. Kawashima’s Brain Training” which was developed for the Nintendo DS and published in 2005.

Minecraft: Education Edition

The game “Minecraft” started as an independent game and was released in 2009, it advanced to a big hit and was bought by Microsoft in 2014. The aim of this first person game is to mine resources and craft new things with them to advance in the game. The education edition of Minecraft is a special modded (adapted) version for educational purposes. Teachers and pupils share the same game level and have to solve different exercises together to achieve specific goals. There are different approaches to bring Minecraft to the classroom by creating different challenges for a variety of school subjects (Gallagher, 2015). Figure 3.8 shows a screenshot of the game Minecraft with its minimalistic graphic style.



Figure 3.8.: A screenshot of the game Minecraft showing its minimalistic graphics.

3. Educational Games

Dr. Kawashima's Brain Training

This game was released in 2005 for the Nintendo DS console and offers mini games to train the players brain. A starting session of a few random mini-games determines how old the players game is and compares it with the players real age. Afterwards, the game allows the player to play the different mini-games to train and get better and faster in solving the exercises. This game maybe does not fit into the category educational game but is mentioned here because it was a big success although it was not a classic game where the player enters a virtual world, it only offered the mini-games to train the brain and it showed the progress the player is making while playing and repeating the exercises. Figure 3.9 shows a screenshot of the game, where the player has completed the first mini-games to determine his brain's age.

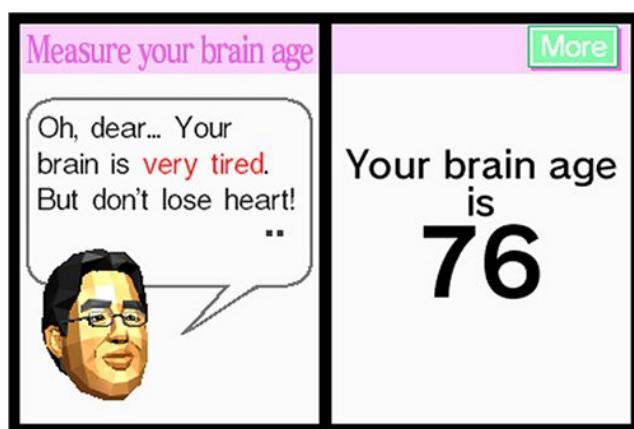


Figure 3.9.: When starting the game the first time, it determines the brain's age by letting the player solve a few random mini-games.

This chapter gave an overview about existing applications and games which can be classified as educational games, where every game has a different approach to teach and engage the player at the same time. It is difficult to create an ILE which satisfies the player's desire to be entertained and integrate the educational content fluidly into the game flow. The introduced games show potential of such learning applications, but it is difficult to integrate them into the classroom. The games have to be highly adaptable

3. Educational Games

to fit the teacher's need and deliver the right content at the right time to the pupils. When looking at the above described games, they are all restricted to test already learned skills which in turn means that they cannot be used in classes to teach new content. The next chapter describes the implemented games and its mechanics in detail.

4. Prototype

The preceding chapters defined and introduced key terms like Virtual Reality, game-based and project-based learning or the maker movement, showed how to create educational applications in theory and listed some sample applications to get a feeling for such applications. The following chapters describe and present the technical part of this thesis, the implemented prototype of the game “VR-Matherallye”, in detail. Section 4.1 outlines the idea and reasons why the game was implemented and the goals it aims to achieve, followed by the description of the gameplay and important game elements in section 4.2. Section 4.3 describes the main parts of the underlying game engine architecture and shows important parts of the implementation and some sample code to ease following the description of processes like initialization or level loading. The second part of the main part deals with the process of evaluating the prototype and the results of this evaluation.

4.1. Idea

As stated in the introducing chapter of this work, with the increasing computational power, their high availability and the variety of embedded sensors, mobile devices are gradually replacing PCs or Laptops in our life. Due to the fact, that children are growing up with mobile technology and are used to play, work and use such devices day-to-day, the idea was born, to implement an educational math game on a mobile phone, to ease accessing the game by using technology known by children and pupils and also ease the distribution of the game.

To create an accessible game the gameplay was kept simple, the player moves through the virtual world and tries to solve exercises by collecting the correct

4. Prototype

result of the current calculation. After solving a couple of exercises the player completes the level and advances to the next one, containing exercises of increased difficulty. To keep pupils engaged and interested in playing the game it was transferred to Virtual Reality by using Google Cardboard, which was introduced in chapter 2. After implementing and evaluating the first version of the game some improvements had to be integrated to keep players motivated and make the game more fun. The second version introduced a time limit, a simple scoring system and power-ups to add more depth to the gameplay. Additional elements like obstacles, for example trees or rocks, the player has to avoid or additional floors in the game level, have been scheduled for the second version but have been delayed to future versions of the game.

As stated earlier in the text, the game is called “VR-Matherallye” and is available for Android devices in the Google Play Store. By using Google Cardboard and the possibility of creating a personal viewer for the game this project successfully combined making with digital game-based learning and Virtual Reality. Figure 5.2 shows a sample of a customized viewer which was used during the evaluation of the game. The next section will explain the gameplay and game elements which were mentioned beforehand.

4.2. Gameplay and Game Elements

How does the navigation of the player’s avatar work? How are levels structured? What items are available and what is their purpose? How are exercises solved, when does the game end? How can the player reach the next level? These questions are answered in the following subsections, where the basic game mechanics are explained and described in detail. The descriptions given do not contain technical details of the different mechanics, which are covered in section 4.3, but can be seen as a manual for the game.

4.2.1. Navigation

An easy navigation and good controls are a key element in successful games, sophisticated control mechanics can decrease the overall gaming experience

4. Prototype

of the player. Due to the lack of input devices for this VR game, a very simple navigation scheme was implemented for the prototype. When a new level is loaded, the player's avatar is placed in the center of the level platform. The camera used to render the scene sits atop the avatar as illustrated in figure 4.1. The avatar moves along the viewing direction with constant speed. By moving around the own axis, the player changes the viewing direction and thus the moving direction of the avatar in the virtual world.

While the avatar is moving through the virtual world, the player is perceiving motion but is not moving himself, which is causing a discrepancy. This discrepancy, called visually induced motion sickness, "typically occurs in the absence of real physical motion, thus, motion sickness in simulators, cinemas, video-games, or Virtual Reality" Keshavarz and Hecht (2014, p. 521) causes symptoms like nausea and dizziness. There have been researches about motion sickness prevention i.e. using pleasant music as countermeasure against visually induced motion sickness Keshavarz and Hecht (2014) or the Entrim 4D motion headset by Samsung, which is "using a combination of algorithms and Galvanic Vestibular Stimulation (GVS), a safe and simple technique that sends specific electric messages to a nerve in the ear, the VR accessory synchronizes your body with changing movements in video content" ("Samsung to Unveil Hum On!, Waffle and Entrim 4D Experimental C-Lab Projects at SXSW 2016," 2016). This headset suggests the brain that the body is moving like the avatar moves in the Virtual Reality and thus prevent motion sickness. To keep the prototype's implementation simple, no technically nor algorithmic methods have been implemented to prevent motion sickness to occur. The induced symptoms could lead to difficulties in keeping the balance while standing and playing the game. The simplest prevention of losing the balance is to sit down on a revolving chair and do the navigation by rotating around the chairs axis.

The evaluation of the game, described in chapter 5, showed that neither motion sickness nor symptoms like nausea or dizziness were a problem for the children testing the game. The previous simple prevention method of using a revolving chair was not needed by any of the pupils.

4. Prototype

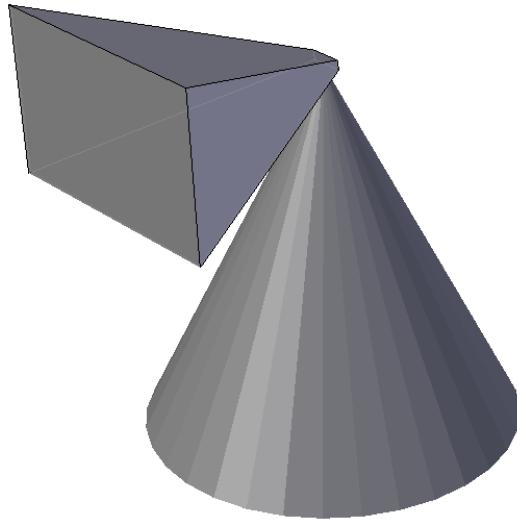


Figure 4.1.: A simple cone model representing the player's avatar. The camera, visualized as the frustum of a pyramid, is placed on top of this cone.

4.2.2. Level Structure and Items

The level structure does not change in any of the five sample levels used to evaluate the game. Every level consists of three key elements, the floor, the player's avatar and collectible items. All game objects except the collectible digits are represented by simple geometric forms to keep the rendering simple and fast and therefore prevent frame rate issues. The following text describes and explains the game objects and their properties.

The avatar of the player is illustrated in figure 4.1, it is the player's representation in the virtual world. The cone shaped avatar always moves into the viewing direction of the player. As stated before, before the game starts, the avatar is placed in the center of the floor. The floor itself, another key element, is represented by a scaled cube and bounds the area the player can navigate his avatar on. Collectible items are divided into two different types, the digits for solving the exercises and pickups acting as power-ups, respectively.

The digits are represented by their corresponding 3D objects and are used to solve the current exercise, the pickups are displayed as green, yellow

4. Prototype

and red boxes. The color of the power-ups visualizes the possible effect of the pickup, where green stands for a positive, yellow for a neutral and red indicates an undesired effect. As the avatar, the pickups and digits are placed onto the floor before the game starts. To summarize the pickups and their effect, the following enumeration lists all power-ups available in the game and a description of their effects, figure 4.2 shows the in-game representation of the pickups.

- **Time Bonus**

Adds thirty seconds to the remaining level time.

- **Speed Up**

Doubles the speed of the avatar for ten seconds. If a second Speed-Up-pickup is collected, while the first speed up is still active, the doubled speed is doubled again.

- **Jump**

The avatar executes a high jump, which can be used to get a better view onto the platform or to jump over pickups or digits the player does not want to collect. The avatar can be navigated while being airborne to prevent it from a jump off the platform, which will end the game immediately.

- **Invert Controls**

The controls are inverted, changing the viewing direction to the right results in the avatar moving left and vice versa. Additionally, the moving direction of the avatar changes from forward to backward. The effect lasts five seconds. This is rather short but as testing showed, inversion of the controls quickly leads to dizziness or motion sickness.

4.2.3. Gameplay

Following the description of the level structure, the game items and the navigation of the avatar, this section explains, how to solve exercises and collect items to accomplish a level in detail. During the loading process of the level the avatar is placed in the center of the platform and the collectible items are distributed randomly on the game floor. More precisely ten digits, nine possibly wrong results and the correct result, and ten pickups are generated every time a level loads. The generation of the pickups is

4. Prototype

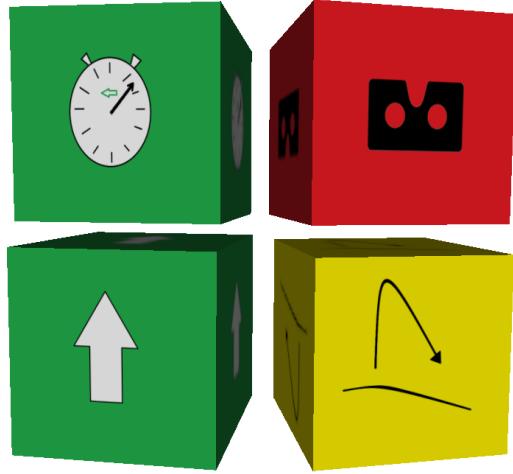


Figure 4.2.: The four different pickups collectible in the game. In the top left corner, the Time-Bonus-pickup, top right the Invert-Controls-pickup, bottom left the Speed-Up-pickup and bottom right the Jump-pickup

completely random, there are no restrictions or rules the pickup creation follows. It is possible that seven time bonuses and three speed-up pickups are created, but no control-inversion or jump-pickup exists. The levels are restricted to ten power-ups, no extra pickups are generated if all pickups have been collected. Due to the time limit of the levels no player collected all power-ups during testing therefore this restriction does not influence the “fun-factor” of the game.

With all the game objects loaded successfully, five exercises or challenges for the current level are generated randomly. The type of the exercises generated depends on the chosen setting from the main menu which is described in section 4.2.4. To start with a moderate difficulty and increase it to a more challenging one, each level defines a range for the results of the generated exercises. As for level one, “Plaza”, the results for the challenges range from 0 to 9 which means the correct result will always be less than ten. Higher levels have other, broader ranges for the results to increase the difficulty. This property also influences the generation of the digits for the level, which are also generated in between the given level range. The ranges for every level can be found in table 4.1. An exercises consists of two digits and either the addition symbol (+) or the multiplication symbol (*) (depending on the

4. Prototype

chosen mode in the main menu) and the equals symbol (=), see listing 4.1 for the regular expression the exercises are created by.

```
1 [1-9]?[0-9](([+[1-9]?[0-9])|(*[1-9]))=
```

Listing 4.1: The rule for exercise generation

After the level loading process is finished, all items have been generated, power-ups and digits are placed onto the platform and the challenges are generated successfully, the avatar starts moving and the clearing time for the level starts to decrease. The clearing times vary from level to level and are chosen tighter for higher levels to gain a second parameter to vary the difficulty of the levels. The remaining time to clear the level can be increased by correctly solving the current challenge or by collecting time-pickups which add 30 seconds of extra time. All collectible items can be “picked” up by navigating the avatar “into” the desired item. It seems like the avatar runs through the object and a sound effect is played, if the item was successfully collected. Three different scenarios are possible when collecting items: a pickup is collected, the player picked the wrong digit or the player solved the current exercise by choosing the correct digit. The following enumeration explains every scenario in detail.

1. A pickup is collected

The simplest of all scenarios, the pickup is applied. The cube representing the pickup is destroyed, the number of available pickups for this level is decremented. The possible effect of each pickup is explained in section 4.2.

2. The wrong result is collected

A sound effect and a message printed on the screen signaling an incorrect calculation. The player’s remaining time is decreased by five seconds, the overall score decreases by ten points. The collected digit is destroyed and a new one is generated randomly according to the range of the current level.

3. The correct result is collected

As with the wrong result, a sound effect and a message indicate the correct calculation. The remaining time is increased by five seconds and additionally 50 points are added to the player’s score. The next

4. Prototype

calculation is displayed on the screen and a digit representing the correct result is generated and placed randomly on the game platform.

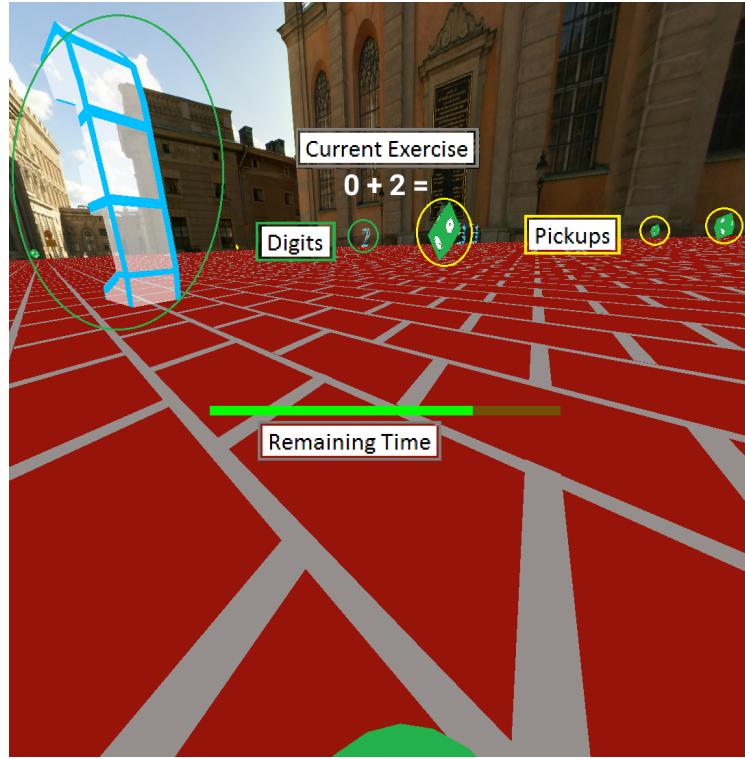


Figure 4.3.: The game screen from the player's point of view. The green circles indicate the digits, the yellow circles are showing pickup items. The bar in the center of the screen represents the remaining time; the current calculation is displayed in the upper half. The green disc at the bottom displays a small fraction of the avatar. The big digit on the left corresponds to an rotated 3, the digits in the game are rotating around their own axis so they can be recognized from every angle.

While looping through the above scenarios the player may solve five exercises within the time limit of the level. If so, the player completed the current level and advances to the next level. The remaining clearing time is added to the player's highscore and the next, more challenging level starts to load. "VR-Matherallye" knows three ending scenarios which are depicted in the following enumeration.

4. Prototype

1. Time limit is exceeded

The remaining time reaches zero and the player was not able to solve five calculations.

2. Falling off the platform

If the avatar is navigated beyond the borders of the game floor, the avatar falls into the void and the game ends immediately, even though time is still remaining.

3. The player completes all challenges of every level

After clearing every of the five sample levels within the time limit by overcoming the five challenges, the player has solved 25 exercises correctly and completes the game.

Regardless of which ending scenario occurred the player's score is shown on the screen. Even though a non-winning scenario is reached the player may have collected some points by calculating correct results or beating a couple of levels. The player can enter a name to save the score, which can be viewed on the "Highscores"-screen of the application to compare the score with other highscores reached by previous playthroughs.

This section showed the rules of the implemented math game and describes all of the game items and mechanics, table 4.1 summarizes the time limits and number ranges for the generated calculations of each level. Figure 4.3 shows a screenshot of the first level of a game in progress. The prototype's human user interface (HUD) is only has two elements, a green bar indicating the remaining clearing time of the level and the current exercise to solve. The screenshot also shows the digits which have to be collected to solve exercises and a time bonus and two speed-up pickups.

No.	Level Name	Time Limit	Range Addition	Multiplication Tables
1	Plaza	60 sec	0 – 9	3
2	City	55 sec	0 – 19	5
3	Jungle	50 sec	0 – 30	6
4	Desert	45 sec	0 – 50	8
5	Mountain	40 sec	0 – 99	7

Table 4.1.: The level properties for the five levels of the prototype.

4. Prototype

4.2.4. Menus and Settings

The preceding sections described the gameplay, the navigation mechanics and how to interact with game items. The game enables the player to change some settings before starting a new game in the main menu. From the main menu, the game can be started, the highscores screen can be opened to show saved highscores from previous games or the application can be closed. This section completes the *Gameplay*-section of this chapter by describing the different adjustable options of the main menu, the highscore list and the “Save HighScore”-screen.

The main menu is always shown first when the app is started to provide the settings for a new game. While the player is adjusting the different options the application loads the game’s assets in the background so the game can be started without loading times. A screenshot of the main menu is shown in figure 4.4, it shows two radio button groups which are used to select the available options. The meaning of all options and their effects on the game is described in the following enumeration.

1. Choose your Level

- Addition only – calculations in every level of the game are additions
- Multiplication only – only calculations according to the levels multiplication table row are generated
- Mix it – additions and multiplications are alternating

2. Are you using Google Cardboard?

- Yes – the game scene is rendered in VR-mode to work with the cardboard headset. Every frame of the game has to be rendered twice, one frame for the left and one for the right eye respectively. See figure 4.12 for a screenshot of a Virtual Reality mode rendered game scene.
- No – the scene is rendered onto the whole display of the mobile device to enable playing without a VR-headset as well. This option enables the player to play without a viewer, therefore only one frame of the scene is rendered. The screenshot of the gameplay 4.3 was captured in non-VR-mode.

4. Prototype

The buttons at the bottom of the main menu, from left to right, start a new game, show the highscores screen or close the app. The highscores screen is shown on the left in figure 4.5, it shows all saved highscores from previous games in a list sorted in descending order of the scores. To create an entry in this list, a name can be saved after a game ends in the before mentioned “Save Highscores”-screen, which is depicted on the right in figure 4.5.

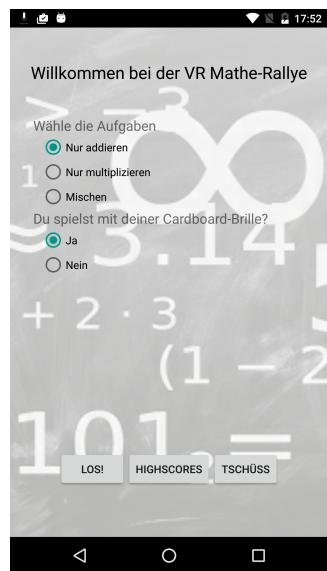


Figure 4.4.: The main menu showed after app startup. Radio buttons in the upper part of the menu are setting game options, the button row in the lower part are used to start a new game, show the highscore list or close the app.

This first part of chapter 4 introduced the key elements of the game, important mechanics to master the levels and challenges and described the different settings available on the main menu of the app. The reader is now ready to play the game and understand all gameplay processes without any further instructions. The next part of this chapter is about important parts of the implementation of the game and the game engine.

4. Prototype

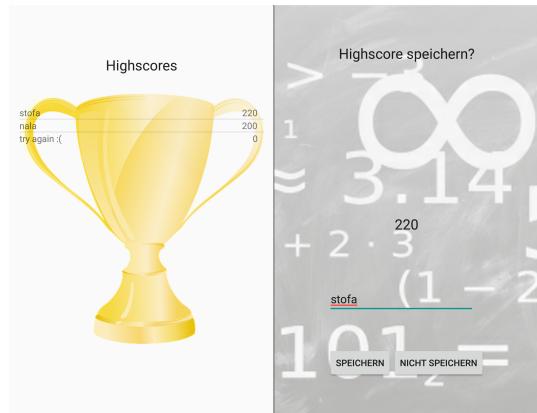


Figure 4.5.: The left part shows the highscores list with some example entries, on the right screen names and corresponding scores can be saved or discarded. The right menu is shown after ending the game either by completing all five levels or in case of losing.

4.3. Implementation

This section is about all technical details and the architecture of the engine which runs the game. The engine's design and structure was inspired by the books *Game Coding Complete* and *Game Programming Patterns* by McShaffry and Graham (2012) and Nystrom (2014). *Android Studio* served as Integrated Development Environment (IDE) for implementation, debugging and testing the source code. The complete engine is written in the *Java* programming language. The reference mobile device, used for the evaluation described in chapter 5, was the *Motorola Nexus 6*. The phones most important specifications are listed in table 4.2 taken from "Motorola Nexus 6 - Full phone specifications" (2000–2016).

The next sections describe the key components of the engine which are the

- Actor-Component-System
- Process System
- Event System
- Rendering System

4. Prototype

and illustrate their connections with the final game prototype. The last section of this chapter is about the heart of the engine, the game logic, containing the main loop and using the different components of the engine to run the game.

Motorola Nexus 6 Specifications	
Dimensions	159.3 x 83 x 10.1 mm (6.27 x 3.27 x 0.40 in)
Display Resolution	5.96" (1440 x 2560 pixels)
Operating System (OS)	Android OS v5.0 (Lollipop) upgraded to v6.0 (Marshmallow)
Chipset	Qualcomm Snapdragon 805
CPU	Quad-core 2.7 GHz Krait 450
GPU	Adreno 420

Table 4.2.: The specifications of the mobile device used to implement, debug and evaluate the game.

4.3.1. Actors and Components

Every element shown on the screen while playing the game is called an actor. The scene of a game can be seen as the stage, where every element on this stage has to play a role and therefore can be compared to actors on a real stage. The implemented engine knows two types of actors, static actors and simple actors. The floor or the sky box for example are static actors because the player cannot interact with them. The avatar, digits and pickups on the other hand are simple actors which can interact with each other. Each actor is a container or wrapper for actor components. These components define properties and different behaviors of one actor, for example the `TransformComponent` holds the actors position and rotation for rendering operations or by adding a `CollisionComponent` the actor will be registered within the collision system of the engine. By integrating this actor-component-system the behavior of actors could be changed at runtime by adding or removing components. Figure 4.6 shows a diagram of the this system with the rectangle on the right representing an actor which holds components B and C and possible other components indicated by the dots. New components from the component pool on the right can be

4. Prototype

added to the actor if they are needed but also components from the actor can be removed if the actor should lose a specific property. One actor can only contain one component of a specific type, it is not possible to add two TransformComponent objects to the actor.

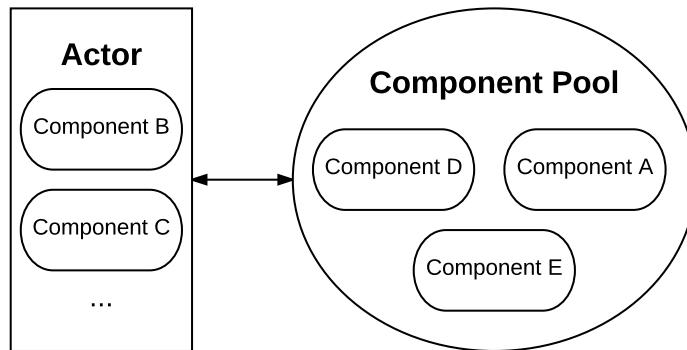


Figure 4.6.: A simple abstraction of the used actor-component-system, with the actor represented by the rectangle on the left and the component pool on the right. Components can be added or removed from the actor which is indicated by the double arrow between the actor and the component pool. Removed components are returned to the pool, one actor can only contain one component of the same type.

Actors are defined in xml-files with the `<Actor>` tag being the root tag of the definition file. Every component can be defined by its own tag, to add a specific component to the defined actor, the `components` tag has to be added as a child node of the actor. By using this design, the data defined in these xml-files defines the behavior of an actor and therefore is called data-driven development. Behavior or properties can be added by changing the components in the actor's definition file and so the engine does not need to be recompiled which in turn saves time when debugging the component system. Listing 4.2 shows the definition file for the avatar in the game. The `type` attribute of the `<Actor>` tag indicates the engine, that the actor being loaded represents the avatar, the `resource` attribute contains the path to the avatars game model. The child tags, `<TransformComponent>`, `<CollisionComponent>` and the `<MeshRenderComponent>` of the root tag represent the avatar's components. Components can have child elements too, to define component-specific properties used for initialization, like the

4. Prototype

<Position> or the <YawPitchRoll> tags.

```
1 <?xml version="1.0" encoding="utf-8"?>
2
3 <Actor type="Player" resource="models/player.obj">
4     <TransformComponent>
5         <Position x="0.0" y="1.0" z="0.0"></Position>
6         <YawPitchRoll x="1.0" y="1.0" z="0.0"></YawPitchRoll>
7     </TransformComponent>
8     <CollisionComponent></CollisionComponent>
9     <MeshRenderComponent>
10        <Color r="0.5" g="0.1" b="0.9" a="1.0"></Color>
11    </MeshRenderComponent>
12 </Actor>
```

Listing 4.2: The definition file for the avatar used in the game.

The different modules of the engine are organized in packages containing all module specific classes. At the end of every section an image of the package is shown to get a feeling of the classes and their connections within such a module. Figure 4.7 shows the package containing the classes needed for the actor creation and initialization. There exists only one Actor class because the actors in the engine act as wrappers for the components and therefore no different actor types are needed. The ActorFactory is used to create actor containing the components defined in the actor definition file.

Figure 4.8 illustrates the components package, containing implementations of the engines basic actor components. Components used in actor definition files have to be registered in the ComponentFactory so the engine can add the desired behavior to the actor. The engine's 6 basic components are listed and described in the following enumeration, the AbstractActorComponent class is the parent class of every component. It defines the basic behavior a component must have to act as actor component, by deriving from this component new components inherit this basic behavior.

- CollisionComponent

Actors containing a collision component are added to the engine's collision system. The avatar, digits and pickups have collision components attached to enable the avatar collecting digits and pickups.

4. Prototype

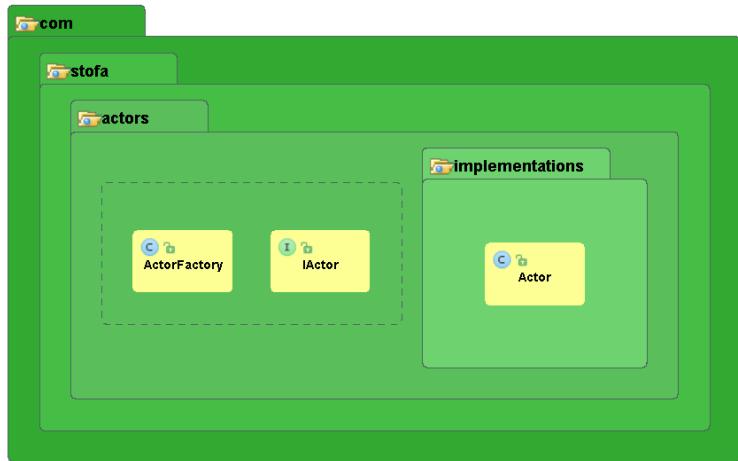


Figure 4.7.: The actor package contains the actor class and the actor factory to create the defined actors.

- **AbstractRenderComponent**

To enable a game object to be rendered, a render component for the game object has to be implemented. This specific render component has to inherit the basic render behavior from the `AbstractRenderComponent` to draw the new game object into the scene.

- **PropertiesComponent**

This component is used by digits and contains the value of the digits. This is needed to check if the player solved the current exercise correctly, when the avatar collects a digit.

- **TransformComponent**

As mentioned before, it is used to query the position and the rotation of an actor.

- **AnimationComponent**

Digits, the avatar and the pickups are animated gain a more dynamic scene.

This section explained the flexible actor-component-system used by the engine to augment actors with new properties or behavior, the next section will explain the process system of the engine in detail. For further study, the commented source code of the `AbstractActorComponent` can be found in listing A.1 of the appendix.

4. Prototype



Figure 4.8.: Components are created via the factory during the level loading process. Different component implementations are shown in the implementation package. All components have to inherit from the *AbstractActorComponent*-base class.

4.3.2. Processes

A process is a sequence of actions, which are executed for a specified time or until some condition is met to end the execution. The time span between the rendering of frame A and the next frame B is used by the engine to update states or check for events which may have occurred. With the implementation of processes in the game engine, a concurrent-like behavior can be reached by updating different processes within this time span. For instance animations can be mapped to processes by defining the length and the different steps of the animation within an animation process. When this process is started, the steps for the animation are executed whenever the process is updated until the animation has reached the last step, which in turn ends the process. By animating game objects with processes, which are updated between the rendering process, the player gets the feeling that all objects are animated at the same time, although the processes are updated sequential between the rendering of frames A and B.

To update and manage the processes running during the game, the engine uses the *ProcessManager*. This manager keeps a list of active processes which are updated one after another between the frames. The actions taking

4. Prototype

place between the rendering of two frames are displayed in figure 4.9, to update the processes the method `onUpdate` in the `ProcessManager` is called.

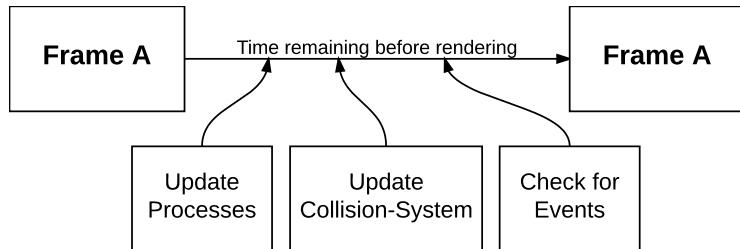


Figure 4.9.: The image shows the three actions which are executed by the engines main loop to update the processes, the collision system and check for events which have to be handled before the next frame.

All processes attached to the `ProcessManager` have to inherit from `AbstractProcess` which defines the basic behavior and the states a process can have which are listed in the following enumeration.

- **Uninitialized** – The process has been created but not yet initialized. Initialization will be done in the next `onUpdate` call of the `ProcessManager`.
- **Running** – This state indicates the process is running and will be updated by the `ProcessManager`.
- **Paused** – A paused process will not be updated but stays in the active process list until the state changes to running.
- **Succeeded** – The process has successfully ended and will be removed from the active process list.
- **Failed** – An error occurred during the process update step, the process is removed from the process list.
- **Aborted** – The engine aborted the process and removes it from the list.

Two example processes running while playing “VR-Matherallye” are the `DelayedEventTriggerProcess` which is initialized with a timestamp and an event which will be triggered when the timestamp is reached or the `RunProcess` responsible for moving the avatar in the viewing direction of

4. Prototype

the player. Figure 4.10 shows the processes package containing all relevant classes to manage the processes in the engine. The RunProcess is not shown in this figure because it is not a core process but a game specific process. The source code of the ProcessManager and the AbstractProcess are shown in listings A.2 and A.3 in the appendix. The upcoming section will explain and describe the important event system of the engine.

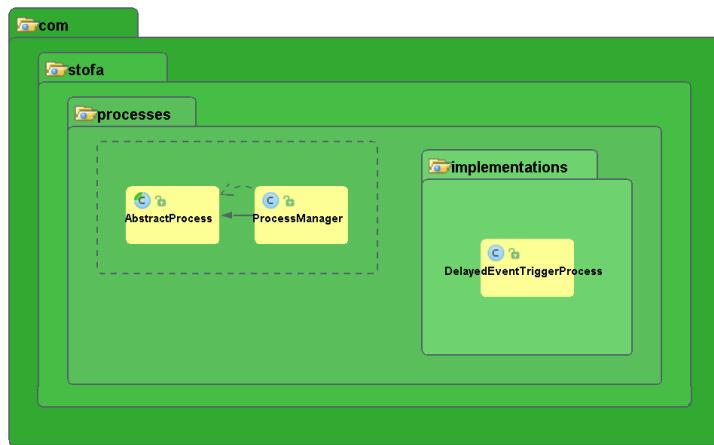


Figure 4.10.: The process manager schedules the attached processes, all attachable processes have to inherit from `AbstractProcess`. An example of an implemented process is the `DelayedEventTriggerProcess` in the implementation package.

4.3.3. Events

Every game engine needs an event system which enables the communication between engine systems without coupling all systems tightly. The event system is, metaphorically speaking, the glue that keeps the different systems of the engine stick together. An example will illustrate this concept in more detail. When the player picks up a digit to solve an exercise, the collision system reports a collision by creating a `CollisionEvent`. The event system distributes this event to all systems of the engine, which have registered to listen for `CollisionEvents`. In the example case, the audio system, the rendering system and the game logic are listening to this kind of event. The audio system plays a sound effect to indicate, that a digit was collected. The

4. Prototype

logic checks if the digit, the player picked up, solves the current exercise or a wrong digit was picked. The rendering system removes the digit from the screen, because picking up an item destroys the collected object. With the help of the event pattern, the collision system can communicate with the audio system, the renderer and the game logic, even though none of these modules hold an instance of the collision system. If another system wants to react to collision events, it only has to register for this event in the event manager. With this decoupling, the complete collision module could be rebuilt or exchanged by another system without touching all other modules.

As with the `ProcessManager`, the `EventManager` is responsible for checking its event queue for events and to trigger them if available. The `EventManager` is also updated between the rendering of two frames as can be seen in figure 4.9. Components of the engine can register as listeners to specific events by calling `addListener`, only components implementing the `IListenable` interface can be added as listeners. When the game is running, events occurring during playing are queued in the event queue, during the rendering of two frames the `EventManager` checks the queue, removes the first event, checks for registered listeners for the specific event and calls the `invoke` method of the listening component with the event data to be processed. This is done until the queue is empty.

```
1 /**
2  * This interface has to be implemented by all classes which want
3  * to receive event notifications
4  */
5 public interface IListenable {
6     /**
7      * Handler for triggered event.
8      *
9      * @param eventData the event data of the triggered event
10     */
11    void invoke(IEventData eventData);
12 }
```

Listing 4.3: The interface implemented by components which can be registered as listeners.

The `EventManager` also offers the possibility to immediately trigger an event

4. Prototype

without queuing it. This function is necessary for events occurring in the initialization and starting states of the engine, for instance when loading a level the starting scene graph is built by triggering events when a new SceneNode should be added to the scene. If these events are queued in the event queue, the renderer will not be able to render the scene because the EventManager checks for available events only when the game logic is in running state.

The EventManager only queues events inheriting from the base event class AbstractEventData, listeners are only registered if they implement the IListenerable interface, as stated before. The simple interface consisting of a single method invoke taking event data as the only parameter is shown in listing 4.3. The implementations of the AbstractEventData and the EventManager classes are shown in listings A.5 and A.4 respectively. Figure 4.11 illustrates the engine's event system and its classes. Terms like scene and scene nodes have been used in this section and will be described in detail in the next section which is about the rendering system.

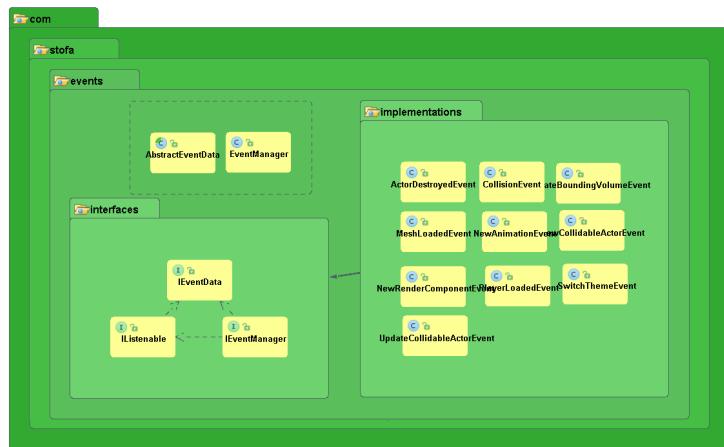


Figure 4.11.: The event package of the engine. The EventManager registers listeners and distributes the events. All events inherit from the base event AbstractEventData. The implementation-package contains some events used by the engine.

4. Prototype

4.3.4. Rendering System

When starting a new game, the game engine starts running and updates the states of game objects like the avatar of the player, the digits, the pickups and also manages the exercises the player has to solve. To visualize the different game objects in their current states the engine uses the rendering system. The rendering system is responsible for drawing the frames of the game displaying the game scene on the screen of the mobile device to enable the user to interact with objects or react to certain game situations. Basically the rendering system is creating a two-dimensional image from the three-dimensional scene containing all game objects in their current states. A rendering system is used by nearly every game engine with engines of text adventures being an exception. In modern engines, rendering systems are highly optimized to render detailed scenes including a big amount of objects in different render passes to create realistic virtual worlds. The methods to render the frames for the implemented prototype were kept at a basic level due to the usage of simple game objects consisting of a few vertices. The renderer contains two basic classes, the Scene and the SceneNode, which are responsible for the proper rendering of a frame and are described in the following text.

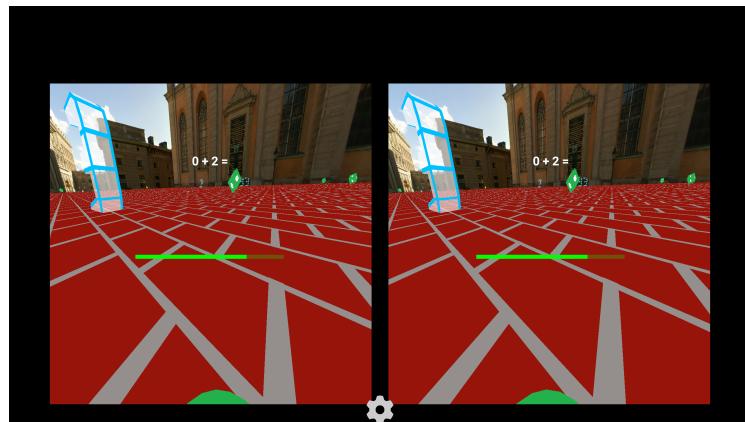


Figure 4.12.: An image of the rendered scene when playing with the cardboard VR headset. Two separate images of the scene have to be rendered, one image for the left eye and one for the right respectively. The left and right image differ slightly because the images are rendered with respect to the eye distance.

4. Prototype

The Scene class contains the scene graph, a tree-like data structure holding all kinds of different SceneNodes, which in turn represent the game objects and their current states, the LightManager, which contains all light sources, and it keeps track of the transformation matrix stack, which is important to render the game objects properly. It also offers methods to add or remove nodes to and from the scene respectively and to start the rendering process. When the engine calls the `onRender` method of the Scene, the scene graph gets traversed depth-first starting with the `RootNode`. Figure 4.13 shows a simplified scene graph to illustrated the tree-like structure.

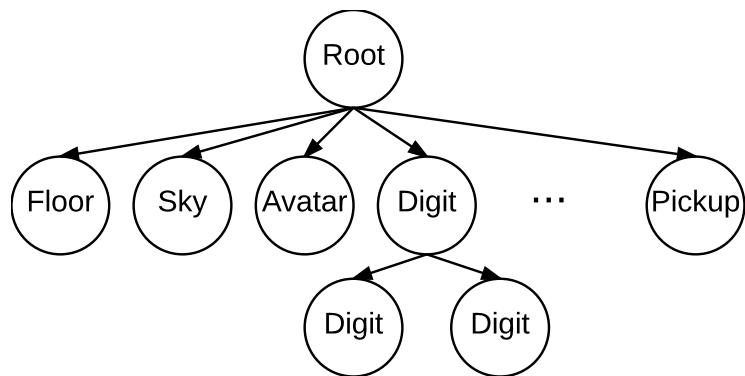


Figure 4.13.: The scene graph constructed at the level loading process. Each node in the tree contains a game object which is rendered when a rendering process is triggered by the engine to draw the next frame.

“VR-Matherallye” is a Virtual Reality game, therefore the scene graph has to be rendered twice, for the left and the right eye respectively, to create two create two images of the scene which take the distance from one eye to the other in account. By using the viewer and “merging” the two images together, the feeling of depth and presence in the virtual world is generated. Figure 4.12 shows the two images rendered for every frame by the rendering system.

The `SceneNode` class is the base class for the different types of nodes, the engine uses to build up the tree. Every node type has a different function but inherits the basic functionality from `SceneNode`. The following enumeration lists and describes the most important node types of the rendering system.

- `CameraNode` – This node represents the camera which sits atop the

4. Prototype

avatar, see figure 4.1. The camera properties define how to render the game objects. The avatar and the camera can be seen as the photographer with its camera taking a picture of a scene. This picture represents the frame rendered to the screen.

- LightNode – Adds a light source to the scene to enable the rendering of shadows and highlights on game objects. These nodes are managed by the LightManager but are not used in the current version of the prototype.
- SkyBoxNode – This node is used to draw the sky and the surrounding environment of the floor. It is a big cube enclosing all game objects with the walls of the cube textured by images. Texturing the walls, metaphorically speaking, is the process of sticking a poster onto a wall to cover the empty or white space of the wall with a colorful picture.
- MeshNode – A MeshNode contains the game objects like the floor, the avatar, etc. When the method `onRender` of this node is called, the object within the node is rendered onto the screen.
- RootNode – As the name states, it is the root of the scene graph and has a special purpose, it manages the render passes to draw a frame. The rendering of a frame in modern engines is not accomplished in one render pass but in a couple of different passes to create different kinds of effects. This prototype is capable of rendering more passes but in fact only one render pass is needed to draw a scene.

4. Prototype

```

1  private int getLod(Matrix4f V, Scene scene) {
2      Matrix4f model =
3          m_Parent.getSceneNodeProperties().getToWorld();
4      Vector3f p1 = new Vector3f(m_MinCoords.x, o.of,
5          m_MinCoords.z);
6
7      p1.mul(model);
8      Vector3f camPos = new Vector3f(V.m30, V.m31, V.m32);
9      p1.sub(camPos);
10
11     // calculate the distance from camera to game item
12     double distance = Math.sqrt(Math.pow(p1.x, 2.0) +
13         Math.pow(p1.z, 2.0));
14
15     // the returned integer determines the divisor
16     // for the triangle count
17     if (distance > 120.0)
18         return m_TrianglesCount;
19     else if (distance > 100.0)
20         return 3;
21     else if (distance > 70.0)
22         return 2;
23
24     return 1;
25 }
```

Listing 4.4: The level of detail calculation in the MeshNode. Bigger distances lead to a bigger divisor and the rendered object is drawn with fewer triangles.

Nodes can be created or destroyed on the fly by the game engine if certain events occur during the game is running. To speed up the rendering process two optimizations have been implemented, a rudimentary frustum culling function and a primitive level of detail (LOD) display of game objects. Frustum culling ensures that only objects which are visible by the camera are rendered. Pickups or digits behind the user are not rendered to free resources for objects which are in the player's field of view (FOV). The level of detail implementation checks the distance of the avatar to drawn game objects. If the distance is bigger than the specified thresholds the amount of rendered triangles of the corresponding item is decreased which again increases rendering speed. If an object is far away, the player is able to distinguish between a digit or a pickup and therefore this primitive function does not affect the flow of play. Listing 4.4 depicts the source code

4. Prototype

of the LOD function, listing 4.5 shows the code of the frustum culling in the `isVisible` method of the `MeshNode`. The method uses the frustum culler offered by the scene to check if one of the vertices creating the bounding rectangle, a rectangle defining the base area of the game item, is inside screen space. If none of the four points lies in a space visible for the player the object is not drawn.

```
1  public boolean isVisible(Scene scene) {
2      if (m_Properties.getRenderPass() == RenderPass.STATIC) {
3          return true;
4      }
5      Matrix4f top = scene.getTopMatrix();
6      Vector3f p1 = new Vector3f(m_MinCoords.x, 0.0f,
7          m_MinCoords.z);
8      Vector3f p2 = new Vector3f(m_MaxCoords.x, 0.0f,
9          m_MinCoords.z);
10     Vector3f p3 = new Vector3f(m_MaxCoords.x, 0.0f,
11         m_MaxCoords.z);
12     Vector3f p4 = new Vector3f(m_MinCoords.x, 0.0f,
13         m_MaxCoords.z);
14
15     p1.mul(top); p2.mul(top); p3.mul(top); p4.mul(top);
16
17     if (scene.getFrustumCuller().isPointInsideFrustum(p1))
18         return true;
19     if (scene.getFrustumCuller().isPointInsideFrustum(p2))
20         return true;
21     if (scene.getFrustumCuller().isPointInsideFrustum(p3))
22         return true;
23     if (scene.getFrustumCuller().isPointInsideFrustum(p4))
24         return true;
25
26     return false;
27 }
```

Listing 4.5: The `MeshNode` method `isVisible` is using the frustum culler to determine if the object should be rendered or not because it is not in the field of view of the player.

To summarize this section, the important parts of the rendering system are illustrated in figure 4.14, to give an overview about the classes contained in the renderer package. In the appendix commented sources of the `SceneNode`

4. Prototype

and a shortened version of the Scene class are listed in listings A.6 and A.7 respectively. The last section of the chapter illustrates the game logic of the engine.

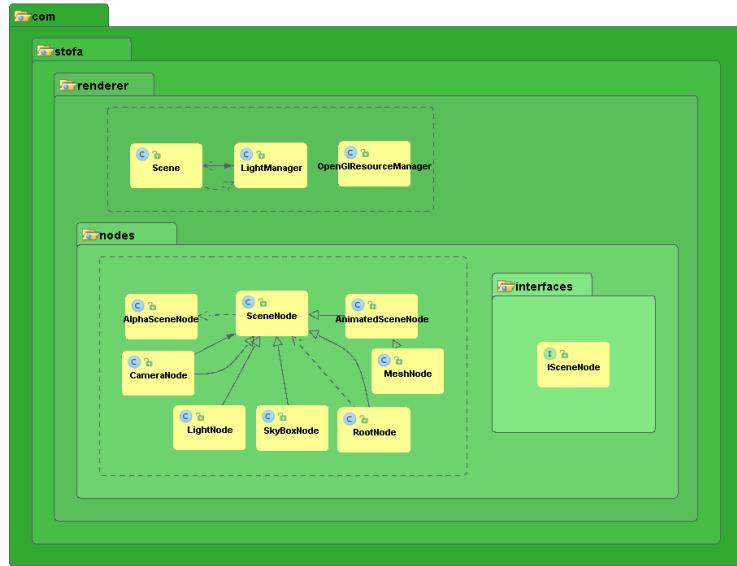


Figure 4.14.: The render package of the engine and its most important classes. The Scene-class manages the scene graph and its nodes. The different kinds of nodes needed to render the scene can be found in the nodes-package. The MeshNode is used the most and is responsible for drawing the game objects of the scene.

4.3.5. The Game Logic

The game logic is the heart of the game engine, it initializes all engine sub-systems in the correct order, reads the level definition files and starts the process of creating the actors with their components and the scene nodes for the scene graph. It also generates the pickups, digits and calculations needed for the loaded level. If everything has been initialized, the engine starts the main game loop, which is running until the player ends, loses or discards the game. The logic itself is a state machine, which means the behavior changes according to the current state the logic is in. The following enumeration lists and describes the possible states of the logic. Figure 4.15

4. Prototype

displays the states of the logic and the possible transitions between the different states.

- **INITIALIZING** – As stated beforehand, the engines sub-systems like the process manager, the event manager or the collision system are initialized. The RunProcess moving the avatar through the virtual world is created and attached to the process manager. With all steps completed successfully, the state is switched to the next state **LOADING_GAME_ENVIRONMENT**.
- **LOADING_GAME_ENVIRONMENT** – With all the sub-systems up and running, the first level of the game is loaded. After successfully loading the level the ten pickups and digits are loaded, the RunProcess is started and the background music is loaded into the audio system. The logic switches its state to **RUNNING**.
- **RUNNING** – The **RUNNING** state is the main state of the game loop, the logic stays in the running state until the level is finished successfully by solving all challenges or the player loses. During the **RUNNING** state the clearing for the level decreases, the process manager, the event manager and the collision system are updated and a new frame is drawn, as illustrated in figure 4.9. From the **RUNNING** state two state changes are possible, back to the **LOADING_GAME_ENVIRONMENT** state if the level is cleared by the player or to the **GAME_OVER** state, if the player could not finish the level or all levels have been finished successfully.
- **GAME_OVER** – If the game is over, the sounds played by the sound system are stopped, the RunProcess is aborted, the engine shuts down all sub-systems and the current score of the player is sent to the “Save Highscore” screen. After saving or discarding the score, the application displays the main menu and the player could start a new game or check the highscores.

When the logic has reached the **LOADING_GAME_ENVIRONMENT** state a new level is loaded. Loading a level is achieved by parsing the level definition file which is again, as with the actor definition files, an xml-file. The root tag of a level defining file is the **World** tag, actors which are part of the world or level follow as child tags. The level definition file knows three types of child tags, **StaticActors**, **Actors** and **Properties**. The tags

4. Prototype

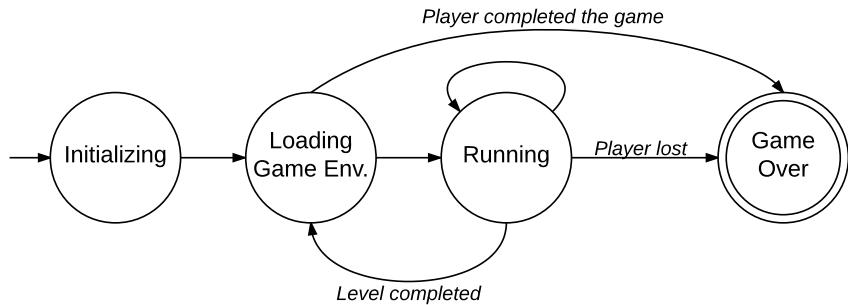


Figure 4.15.: The game logic displayed as state machine with the possible transitions indicated by arrows. A switch from “running” to “loading game environment” is performed to load the next level. If no new level can be loaded, the player has completed the game. When the clearing time of the level reaches zero or the avatar falls off the platform the player loses. In any case, the last state is always the “game over” state.

are described in the following itemization.

- **StaticActors** – All actors defined within this tag are treated as static actors by the engine. In “VR-Matherallye” only two kinds of static actors are known, the platform and the sky box.
- **Actors** – The game items like the digits and the pickups are actors, they are animated and the avatar can interact with them. The avatar itself also is an actor which is moving through the virtual world. All actors define within the Actors tag indicate that they have a special purpose in the game.
- **Properties** – The child tags of this tag define the level properties like the time limit, the bounds for calculation results and the number of digits which should be rendered in the level.

Listing 4.6 shows the definition file for level one, “Plaza”, of the game. The properties of the game are defined, the clearing time is 60 seconds, the results of the additions generated do not exceed nine, if the game runs in multiplication-mode, only multiplications with three are generated. The maximum amount of generated digits is set to ten, which means that there are always ten digits present in the world with one being the correct result. The only actor define within the Actors tag is the avatar because the pickups

4. Prototype

and digits for this level are generated dynamically after the level file was parsed.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <World>
3     <StaticActors>
4         <Actor resource="actors/skybox.xml"></Actor>
5         <Actor resource="actors/floor.xml"></Actor>
6     </StaticActors>
7     <Actors>
8         <Actor resource="actors/player.xml"></Actor>
9     </Actors>
10    <Properties>
11        <TimeLimit limit="60"></TimeLimit>
12        <AdditionBound from="0" to="9"></AdditionBound>
13        <MultiplicationTables row="3"></MultiplicationTables>
14        <DigitNumber number="10"></DigitNumber>
15    </Properties>
16 </World>
```

Listing 4.6: The level definition file of the first level in the game.

The following passage describes the instructions executed when a level definition file is parsed by the game logic. At first, the static actors are loaded, in the implemented prototype there is no difference in loading a static actor and a regular actor like the avatar. The `<actor>` tag is parsed, the `resource` attribute of the tag points to the actor definition file, listing 4.2 shows the avatar definition. As described in the actor-component-system section, an actor object is created and its components are added. This is done for every actor in the level textxml-file to create and initialize all actors for the current level. After creating all actors, the level specific properties and rules are parsed. If the process ended successfully the logic generates the exercises and pickups according to the level properties and the logic state changes to RUNNING. A simplified flow diagram of this process is displayed in figure 4.16.

This chapter described the game “VR-Matherallye” and its gameplay including game items, game flow and how to navigate through the virtual world. The second part gave some insights into the underlying game engine, which was created from scratch to develop the game. The most significant sub-systems of the engine and their connections among each other were

4. Prototype

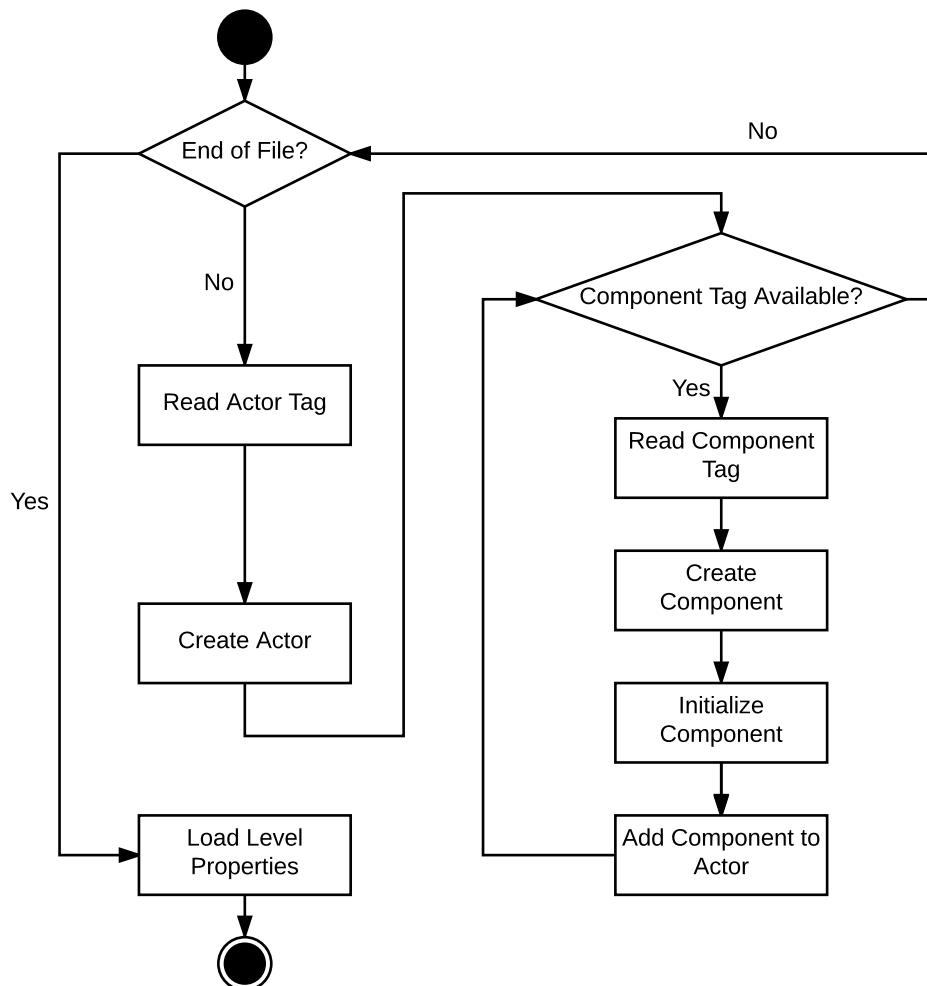


Figure 4.16.: A simplified flow diagram of the function used to load and initialize a level and its actors. Two while loops are needed to initialize a level properly. The first loop iterates through actor tags in the level definition file. If an actor is created, the second loop iterates over the components defined in the actor's definition file, creates and initializes the component and adds it to the actor created beforehand. After all actors have been created the level specific properties are loaded and the level is loaded successfully.

4. Prototype

introduced and the game logic including its states and their transitions were described. The next chapter is about the evaluation of the game by pupils of a school class and its results.

5. Evaluation

The following chapter describes the evaluation process and its results in detail. This chapter is structured in two subsections, the first section describes the evaluation setup and process. Section 5.2 shows the results, issues and conspicuousnesses observed while the pupils were testing the game with the VR headset mounted and finally draws a conclusion about the evaluation of the prototype.



Figure 5.1.: The image shows the 14 pupils who evaluated the prototype.

5. Evaluation

5.1. Evaluation Setup

The evaluation took place on the 18th of December in the “Neue Mittelschule (NMS) Fröbel” (secondary school) and lasted about two hours. The class, taught by Mrs. Silvana Aureli, counts 15 pupils, with one child missing on the evaluation day. The proportion of boys and girls was almost equal, the children were aged 12 to 13. Figure 5.1 shows a group picture of the class taken after the evaluation. After a short introduction of the game, about the evaluation and their tasks, the pupils were split into groups of three and one group of two. The evaluation took place in a separate room with only the current testing group present. Every group ran through a four-step-process to test and evaluate the game. Figure 5.2 shows an image of the cardboard headset used by every group to evaluate the prototype.



Figure 5.2.: The picture shows the cardboard headset used to evaluate the prototype. The pupils tinkered their own headsets after the evaluation and therefore one headset was provided and used to evaluate the game.

1. Introduction

The evaluating group of pupils got a brief introduction about the game and its mechanics. Neither of the children in the group had played

5. Evaluation

the game before nor had experience with VR headsets or VR games. Therefore the children got instructions on how to use the cardboard VR headset, used for the evaluation, how to navigate through the virtual world and what the aim of the game is.

2. First playthrough

After introducing the game and the VR headset, the first child of the group started playing the game for the first time. Every child had the chance to play until it failed or completed the game. Each child of the testing group played the game once and after that first playthrough, possible questions were answered, some advice was given to avoid common mistakes made in the first iteration of the test.

3. Second playthrough

The second playthrough did not differ from the first, the children played the game again, considering the tips given after the first playthrough.

4. Evaluation

The evaluation of the game consisted of five statements about the game. The current evaluation group had to rate the statements by marking them with one of four differently tempered smileys shown in figure 5.3. One statement had to be rated with one smiley by the whole group, not child by child. This “cut-off”-method was used to gain some extra information about the feelings of the children during their discussion on how to rate the current statement. This approach has already been used in previous studies (Spitzer & Ebner, 2015).

The evaluation process lasted approximately two hours. Each of the five evaluation groups ran through the above described evaluation process and rated five evaluation statements. Two months after the evaluation took place, the pupils created their own VR-headsets in six school lessons. The construction manual needed to tinker the headsets and the bi-convex lenses to create the stereoscopic vision of the 3-dimensional world created by the prototype, were offered by the Graz University of Technology. According to the teachers supervising the tinkering lessons, all pupils enjoyed creating their own personalized viewers. The results of the evaluation and the ratings of the statement are presented in the next section.

5. Evaluation

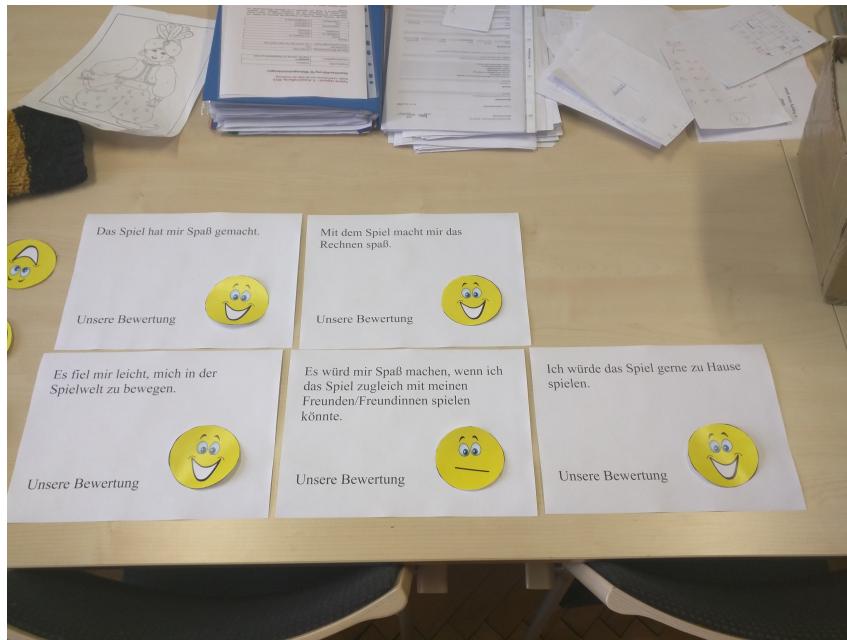


Figure 5.3.: The image shows the statements and the smileys used to rate each statement.

5.2. Results of the Evaluation

As described in point 4 of the evaluation process, each of the evaluating groups rated five statements by marking them with smileys. The statements were read out loud with additional explanations about each statement's intentions, to be sure every child understands the meaning of every statement, begin able to rate them properly. To get a measurable result, every type of smiley is assigned a score from -2 to +2. The scores are added up group by group with the sum being the final score of the statement.

The meanings and scores of the different smileys can be found in table 5.1. The following sections describe each statement, the group-specific ratings and a brief interpretation of each result. In parentheses below each statement, the original German statement, as used during the evaluation, is shown. Observations made during the playthrough-steps of the evaluation process and discussions, on how to rate the statements, respectively, are described and interpreted afterwards.

5. Evaluation

Smileys and their meaning		Points
	The evaluating group totally agrees with the statement of the question	2
	The group was not fully convinced but positive about the statement	1
	The children were skeptical about the statement, but did not completely disagree	-1
	The group totally disagrees with the statement	-2

Table 5.1.: The four different smileys used to rate the statements of the evaluation.

5.2.1. Statement 1

“The game was fun.”
 (Das Spiel hat mir Spaß gemacht.)

Results and Interpretation

Table 5.2 shows the final results for statement one. The groups all agreed on statement one and rated it with the highest score. The children liked to immerse into the virtual world created by the game and navigate through it. It was an overwhelming experience for the children to be part of a virtual world. The fact that no child of the evaluating groups had experience with Virtual Reality certainly had a big influence on the rating.

5. Evaluation

Group No.	Points
1	2
2	2
3	2
4	2
5	2
Result	10

Table 5.2.: Final ratings of statement 1.

5.2.2. Statement 2

“Calculating was fun using the game.”
(Mit dem Spiel hat mir das Rechnen spaß gemacht.)

Results and Interpretation

Statement two had the same rating as statement one, shown in table 5.3. The intention of this statement was to check, if the children like solving math exercises while playing a computer game. On the one hand the high rating could mean that all evaluating children liked the game-based approach of learning math, on the other hand, while discussing in the group, some pupils pointed out that the calculations were too easy. The reason for the easy exercises is, that the prototype’s difficulty was designed for pupils attending primary school. Maybe the rating of statement two could have been worse by generating calculations with a higher difficulty. A few children also stated that they did not try to solve the exercises but they liked exploring the virtual world. Again the effect of diving into a virtual world via the VR headset could have influenced the ratings of this statement.

5. Evaluation

Group No.	Points
1	2
2	2
3	2
4	2
5	2
Result	10

Table 5.3.: Final ratings of statement 2.

5.2.3. Statement 3

“It was easy to navigate through the virtual world.”

(Es fiel mir leicht, mich in der Spielwelt zu bewegen.)

Results and Interpretation

A good navigation mechanic is an important requirement to create a good user experience. The intention of Statement three was to evaluate the implemented control mechanics of the game. With a final score of seven points, all groups rated the game’s navigation positive, although three of the five groups did not rate the statement with the full score. Playing a computer game always implies the use of an input device like a game controller, a keyboard or the fingers touching the screen of a mobile device, to control and navigate the game’s avatar. Since the evaluating pupils were not used to VR-Games nor VR-Headsets, navigating by moving the whole body took some time to get used to. All evaluating children managed to navigate through the world and play the game properly on their second playthrough, therefore, the navigation mechanics can be considered a success, table 5.4 shows the ratings and the final score of statement three.

5. Evaluation

Group No.	Points
1	1
2	2
3	1
4	2
5	1
Result	7

Table 5.4.: Final ratings of statement 3.

5.2.4. Statement 4

“It would be fun, playing the game together with my friends.”

(Es würde mir Spaß machen, wenn ich das Spiel zugleich mit meinen Freunden/Freundinnen spielen könnte.)

Results and Interpretation

The aim of statement four was to evaluate the need for a multi-player mode. The final score of seven points (shown in table 5.5) indicates that multiple players playing the game together in the same virtual world would be a good extension to the current prototype of the game. Surprisingly one group did not like the idea of playing against other players and rated the statement with -1 points, but a multi-player mode being a good idea was the general opinion.

5. Evaluation

Group No.	Points
1	-1
2	2
3	2
4	2
5	2
Result	7

Table 5.5.: Final ratings of statement 4.

5.2.5. Statement 5

“I would play the game at home.”
 (Ich würde das Spiel gerne zu Hause spielen.)

Results and Interpretation

The intention of this statement was to find out, if the evaluating group members would download the game and play it at home with their own VR-headset. All groups top-rated this statement which indicates the success of the combination of game-based learning in a virtual world combined with making. The Virtual Reality experience motivated the children to tinker their own VR-Headset, which was done in six lessons. For the sake of completeness, table 5.6 shows the final results of statement five.

Group No.	Points
1	2
2	2
3	2
4	2
5	2
Result	10

Table 5.6.: Final ratings of statement 5.

5. Evaluation

5.2.6. Observations during Evaluation

Following the results of the evaluation, this section describes some observations made during the evaluation process and gives some explanations about their potential causes. The following enumeration does not follow a specific order.

1. The evaluating person begins to walk forward.

While playing the game some pupils began to walk into their viewing direction, this could be caused on the one hand, by the desire to move the avatar faster into its moving direction, or on the other hand by the discrepancy caused by moving within the virtual world without moving one's body.

2. The testing pupils did not see the current calculation to solve.

This observation was mostly made during the first playthrough. When the evaluating person was told to look up a little to see the current exercise the person lifted the head which moves the game's camera upward, causing the calculation to move upward as well. The eyes have to be moved upwards to see the exercise without changing the camera angle in the game. This could take some time to get used to it.

3. Bending the whole body or tilting the head to the left or right was used to navigate the avatar.

Due to the new way of navigation without input devices and only a brief introduction to the new navigation mechanics some pupils needed some extra time to get used to navigation of the avatar and tried different motions to control the avatar. After some short instructions during playing the game, the pupils internalized the navigation process.

4. The first playthrough during the evaluation process was worse than the second playthrough.

The first playthrough sometimes lasted only a few seconds because the avatar fell off the game platform due to bad navigation. If this was the case, the child was instructed again and could start the playthrough one more time which resulted in a far better first playthrough. The cause of this problem is the start of the game, because the avatar moves forward immediately after the game starts. Until the evaluating person

5. Evaluation

has adjusted the VR-Headset the avatar may have moved close to one edge of the platform.

5. Motion sickness did not occur.

This was a very positive observation because previous, adult, test persons could not play the game for a long time due to motion sickness.

6. During the evaluation-step there was little discussion between the group members.

The intention of forming groups to evaluate the game was to observe the discussion during rating of the statements. Unfortunately, there was little discussion between the group members, almost every group had some kind of leader, who took the rating-smileys and distributed them with the other group members agreeing in most of the times.

This chapter explained the evaluation process and the rating of the evaluation. The results of the evaluation have been very positive and they suggest that there is a high potential in game-based learning combined with making when used for educational purpose. The next section concludes the thesis and discusses issues which should be tackled when extending the prototype and outlines potential improvements to the current prototype.

6. Discussion and Conclusion

This is the concluding chapter of the thesis, it will discuss issues and possible solutions of the implemented prototype and outlines future research directions. It will also answer the research questions from the introductory chapter of this thesis. This chapter contains three sections, the first section carries a discussion about the results of the prototype's evaluation, the second section is about future research direction and the third section finally concludes this thesis.

6.1. Discussion

The discussion section is divided into three subsections, the first part is about the gameplay of the prototype, discussing the positive and negative aspects and how to improve the game flow. The second section lists problems and mistakes, which occurred during the evaluation process and gives suggestions on how to improve it for further evaluations.

6.1.1. Gameplay

The game design and the navigation mechanics were held simple, to create a beginner friendly, Virtual Reality experience. Navigating by changing the viewing direction is an easy way to control the avatar but was hard to explain, while introducing the game's mechanics to the pupils. After a first hands-on and a few minutes with the game, all pupils got used to it. The game objects could clearly be recognized and the children understood how to solve calculations and use the pickups to their advantages. Changing the level after three correct solved exercise also added to the fun factor of the

6. Discussion and Conclusion

game and motivated the pupils in the group, to solve more exercises than the previous group member to advance to the next level.

Some pupils pointed out, that calculations are too easy to solve, but as stated in the previous chapter, the prototype was implemented for pupils of primary schools. With a short interval given to bound the results of additions, sometimes preceding exercises result in the same solution. A better Random Number Generator (RNG) has to be implemented, to solve this problem and to get a better distribution of the exercises. Another problem is a drop of the frame rate, which occurs when too much objects have to be rendered. This could also be prevented by a more advanced RNG distributing the created objects equally over the gaming platform to avoid placing too many digits or pickups on the same area of the platform. But a better distribution will not completely fix this problem. If the avatar is positioned near a boundary of the game floor and the player changes the viewing direction, to get back into the middle of the platform, the player gains a big field of view, containing many game objects, which in turn have to be rendered and a frame rate drop could occur. The engine has to be extended by more advanced rendering patterns to get a steady frame rate in all possible situations. Positively the frame rate problem either did not occur during the evaluation or it did not disturb the game experience for the pupils although it occurred. If the prototype is extended or revised these problems have to be addressed. The current calculation to solve, displayed in the upper area of the screen was not seen by all pupils at once. A better way to display text onto the screen should be implemented to make the player see the calculations immediately without any help.

The combination of the simple game mechanics and the new experience of emerging into a virtual world via a VR-Headset was an exciting experience for the pupils, there were no complaints about the simplistic graphics or missing realism, which is standard in AAA-games (spoken “triple A”, standing for high budget video games) on the new console generations or the PC nowadays. As suggested by Kapp et al. (2014), the prototype should be extended to offer a story and story-telling to provide better engagement and long-term motivation. A feedback system and better evaluation of the learn progress should be implemented to visualize the learning progress of the player, which has a motivating effect too. Adding new challenges like boss-fights, a multi-player mode, enabling comparison with other human

6. Discussion and Conclusion

players, may increase engagement as well as adding connection to social media. Showing the learning progress to friends via social media could lead to engaging friends to replay the game and progress further in the game.

6.1.2. Evaluation

The four step evaluation process, as described in chapter 5, turns out to be a good method to gain information about the prototype. The evaluating group size of three pupils per group and one group of two was a good decision. The groups were big enough to have the attention of every group member while introducing the game and the headset and to deal with the limited time for the whole evaluation process.

A problem of multiple groups was the introduction of the game for every group. Future groups may have gotten a more detailed introduction due to mistakes made by previous groups. Also two playthroughs have not been planned, but after the first playthrough of group one, a second playthrough, after a second briefing, seemed to be necessary and it showed that every group needed this extra briefing and a second playthrough.

Evaluating the game via rating statements using smileys was very intuitive for the children. Every statement was read out loud and additional information was given to be sure, every statement is understood by every child. The intention of three group members having only one vote for a statement was, to observe the pupils, while discussing how to rate the statement and gain some additional information about the thoughts and feelings of the children. Unfortunately, as stated in the evaluation chapter, every group had some kind of leader, distributing the smileys at first. The other group members agreed in most cases without having a discussion about the rating. Rating the statements by pupil and a little interview on the one hand could have resulted in some additional or more precise information but on the other hand would have exceeded the time limit for this evaluation. Distributing smileys to every child in the group could be a second way to improve the rating process. Now each child gets the chance to rate each statement and by different ratings of the same statement could result in a discussion between the pupils.⁰¹

6. Discussion and Conclusion

Despite these shortcomings, the evaluation process itself was good enough to gain some vital information and results, to build atop for improvements and extensions of the prototype. It clearly showed, that the pupils are willing to use and play educational games and enjoyed the game-based approach. Table 6.1 summarizes the known issues and suggested solutions as listed in the above text.

No.	Issue	Solution
1	Same solution for subsequent exercises	Add a more advanced random number generator.
2	Possible drops of the frame rate resulting in jerking and thus could result in motion sickness.	Implement a more advance rendering algorithm including better Level-Of-Detail (LOD) mechanics.
3	Current exercise is difficult to see.	Implement a better Human User Device (HUD) and find better ways to position text elements on screen.
4	No story was added to the game.	Add a compelling story for a higher level of engagement.
5	Common game elements are missing	Add well-known game elements like a multi-player mode, enemies, boss fights, etc.
6	Introduction of the game to every single evaluating group was biased.	Create a script to follow when introducing the game to the evaluating groups.
7	No discussion during the rating of the statements.	Rating the statements by pupil including a little interview.

Table 6.1.: A list of the issues and suggested solutions identified during the evaluation and testing of the prototype.

6. Discussion and Conclusion

6.2. Future Research Directions

The evaluation of the prototype in the school showed that there is a high potential in educational games. Children are used to playing games and the combination of playing and learning is a great way to break up old structures of today's teaching. A more advanced prototype could be used for testing pupils or to submit results of a homework. All pupils get the same exercises and play the game at home. Their results are submitted as homework. Another step would be the integration into lessons, to make the game a tool for the teachers, to teach new content to pupils. For that purpose, a good feedback system to better evaluate and visualize the learning progress of every pupil, has to be implemented. Also further testing of the prototype has to be done to fix unknown flaws in the gameplay.

Using the prototype during school lessons and integrating such an educational medium into learning scenarios in school demands some initial training for teachers to use this new schooling method properly. It should be evaluated how such training for a teaching staff should look like and if it demands explicit training lessons for teachers or if precise guidelines, on how to integrate and use such media in school, will be sufficient. The aforementioned integration of story elements should be evaluated as well. How could adding a story be done and how will the story elements fit with the math content tested by the current prototype. How can challenges of the game mask mathematical calculations and integrate them into the gameplay to blend learning and playing without being recognized by the player.

The time period between the evaluation and the tinkering lessons should be better timed for future experiments and evaluations. After the first hands-on pupils are motivated and engaged to create their own headsets to immediately play the game. By delaying the tinkering lessons too long the motivational effect could suffer. Further research should investigate how the possibility of creating a personalized head-mounted display fosters the creativity and tinkering skills of the pupils.

6. Discussion and Conclusion

6.3. Conclusion

The aim of this thesis was to evaluate the need for educational computer games in combination with making. The game mechanics and implementation details have been described in the previous chapters of this thesis, the making-part consisted of tinkering a VR-Headset made of cardboard. The game was evaluated by 14 pupils with one headset and a testing device. After the evaluation, the children tinkered their own VR-Headset with the help of their teachers to play Virtual Reality games. In six lessons the pupils created their own, personalized Virtual Reality viewers. The teachers mentioned the positive attitude of the pupils during the tinkering lessons and were amazed by the high motivated children, although it was no easy task to tinker the cardboard headsets. The lessons were a big success and the pupils were happy with their headsets. Figure 6.1 shows examples of the headsets created by the pupils. (RQ1)



Figure 6.1.: The headsets tinkered by the pupils in extra school lessons to play the prototype and other VR games.

Although the game was kept very simple in its presentation due to performance bottlenecks, resulting from mobile hardware and the simple render-

6. Discussion and Conclusion

ing system, the pupils were excited to play the game. Emerging into the virtual world via the cardboard headset was an overwhelming experience, because none of the evaluating children used a VR-Headset ever. The results of the evaluation have been positive throughout the evaluation process and none of the pupils felt motion sickness nor criticized the game or its mechanics. Navigation within the virtual world needed some time, so the children got used to it, but the learning curve was steep and every child was able to solve exercises and therefore was motivated to play on. (RQ2)

The results of the evaluation clearly show, that educational games in combination with making can be a great success. The prototype created an extended learning experience for the pupils. Using their mathematical knowledge to advance in the game motivated the pupils to play on and try again after failing to finish a level or the game. By using and playing "VR-Matherallye", the process of repetition to tighten the mathematical skills of the children is added a fun and motivational component, leading to long-term engagement. To improve the learning experience, the mathematical content must be highly adapted to the level of education of the users. Additionally, a compelling story should be added and the user interface must be improved to keep the pupils or players motivated and to reduce the time of teacher introductions. (RQ3)

The impact of a better looking, realistic 3D game using educational elements and integrating one or more new elements, mentioned in the recommendations of the previous section, could have an even greater impact and show great potential to be used in schools and for education of pupils. Maybe the positive and educational effects of learning by doing and game-based learning will revolutionize learning as we know it today.

Bibliography

- Abt, C. C. (1987). *Serious games*. Lanham, MD: University Press of America.
- Adkins, S. S. (2016, August). *The 2016-2012 Worldwide Self-paced eLearning Market: Global eLearning Market in Steep Decline*. Ambient Insight. Retrieved September 16, 2016, from http://www.ambientinsight.com/Resources/Documents/AmbientInsight-The%202016-2021-Worldwide_Self-paced%20eLearning_Market.pdf
- Ali, N., Ullah, S., Rabbi, I., & Alam, A. (2014). The Effect of Multimodal Virtual Chemistry Laboratory on Students' Learning Improvement. In L. T. de Paolis & A. Mongelli (Eds.), *Augmented and Virtual Reality* (Vol. 8853, pp. 65–76). Lecture Notes in Computer Science. Cham: Springer International Publishing. doi:10.1007/978-3-319-13969-2_5
- Blikstein, P. & Krannich, D. (2013). The makers' movement and FabLabs in education. In J. P. Hourcade, N. Sawhney, & E. Reardon (Eds.), *Proceedings of the 12th International Conference on Interaction Design and Children* (p. 613). doi:10.1145/2485760.2485884
- Breuer, J. (2011). *Spielend lernen? eine bestandsaufnahme zum (digital) game-based learning*. Landesanstalt für Medien NRW. Retrieved May 4, 2016, from http://lfmpublikationen.lfm-nrw.de/index.php?view=product-detail&product_id=190
- Burdea, G. & Coiffet, P. (2003). *Virtual reality technology*. Academic Search Complete. Wiley. Retrieved April 16, 2016, from <https://books.google.ca/books?id=oxWgPZbcz4AC>
- Cruz-Neira, C., Sandin, D. J., & DeFanti, T. A. (1993). Surround-screen projection-based virtual reality: the design and implementation of the CAVE. In M. C. Whitton (Ed.), *SIGGRAPH '93 Proceedings of the 20th annual conference on Computer graphics and interactive techniques* (pp. 135–142). New York: ACM. doi:10.1145/166117.166134

Bibliography

- Daineko, Y., Ipalakova, M., Dmitriev, V., Giyenko, A., & Rakhimzhanova, N. (2015). 3D Physics Virtual Laboratory as a Teaching Platform. In L. T. de Paolis & A. Mongelli (Eds.), *Augmented and Virtual Reality* (Vol. 9254, pp. 458–466). Lecture Notes in Computer Science. Cham: Springer International Publishing. doi:10.1007/978-3-319-22888-4-34
- de Paolis, L. T., Ricciardi, F., & Giuliani, F. (2014). Development of a Serious Game for Laparoscopic Suture Training. In L. T. de Paolis & A. Mongelli (Eds.), *Augmented and Virtual Reality* (Vol. 8853, pp. 90–102). Lecture Notes in Computer Science. Cham: Springer International Publishing. doi:10.1007/978-3-319-13969-2-7
- Deterding, S., Dixon, D., Khaled, R., & Nacke, L. (2011). From game design elements to gamefulness. In A. Lungmayr (Ed.), *Proceedings of the 15th International Academic MindTrek Conference Envisioning Future Media Environments* (pp. 9–15). New York, NY: ACM. doi:10.1145/2181037.2181040
- Ebner, M., Schön, S., & Narr, K. (2016). *Making-Aktivitäten mit Kindern und Jugendlichen: Handbuch zum kreativen digitalen Gestalten* (1. Auflage). Norderstedt: Books on Demand.
- Entertainment Software Association. (2005). Essential Facts about the Computer and Video Game Industry. Retrieved May 4, 2016, from <http://www.tntg.org/documents/gamefacts.pdf>
- Entertainment Software Association. (2015). Essential Facts about the Computer and Video Game Industry. Retrieved May 4, 2016, from <http://www.theesa.com/about-esas/esa-annual-report/>
- Evens, E. D. (2007). A brief history of (virtual) reality. Retrieved May 4, 2016, from <http://kunochan.com/?p=33>
- Gallagher, C. (2015). *Minecraft in the classroom: Ideas, inspiration, and student projects for teachers* (Online-Ausg). San Francisco: Peachpit Press.
- Garris, R., Ahlers, R., & Driskell, J. E. (2002). Games, Motivation, and Learning: A Research and Practice Model. *Simulation & Gaming*, 33(4), 441–467. doi:10.1177/1046878102238607
- Gartner Press Release. (2016). Worldwide Device Shipments to Grow 19 Percent in 2016, While End-User Spending to Decline for the First Time. Retrieved August 21, 2016, from <http://www.gartner.com/newsroom/id/3187134>
- Gershenfeld, N. (2005). *Fab: The coming revolution on your desktop - from personal computers to personal fabrication*. New York, NY: Basic Books.

Bibliography

- Grimus, M. & Ebner, M. (2014). Learning with Mobile Devices Perceptions of Students and Teachers at Lower Secondary Schools in Austria. In J. Viteli & M. Leikomaa (Eds.), *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2014* (pp. 1665–1674). AACE.
- Habok, A. & Nagy, J. (2016). In-service teachers' perceptions of project-based learning. *Springerplus*, 5(83). doi:10.1186/s40064-016-1725-4
- Halverson, E. R. & Sheridan, K. (2014). The Maker Movement in Education. *Harvard Educational Review*, 84(4), 495–504. doi:10.17763/haer.84.4.34j1g68140382063
- Hatch, M. (2014). *The maker movement manifesto: Rules for innovation in the new world of crafters, hackers, and tinkerers*. New York: McGraw-Hill Education.
- Heilig, M. L. (1962). Sensorama simulator. US Patent 3,050,870. Google Patents. Retrieved April 19, 2016, from <https://www.google.com/patents/US3050870>
- Kapp, K. M. (2012). *The gamification of learning and instruction: Game-based methods and strategies for training and education*. Pfeiffer essential resources for training and HR professionals. San Francisco, CA: Pfeiffer.
- Kapp, K. M., Blair, L., & Mesch, R. (2014). *The gamification of learning and instruction fieldbook: Ideas into practice*. San Francisco, CA: Wiley.
- Kelly, K. (2016). The untold story of magi leap, the world's mos secretive startup. Retrieved April 19, 2016, from <http://www.wired.com/2016/04/magic-leap-vr/>
- Keshavarz, B. & Hecht, H. (2014). Pleasant music as a countermeasure against visually induced motion sickness. *Applied ergonomics*, 45(3), 521–527. doi:10.1016/j.apergo.2013.07.009
- Kuusisto, F. (2015). Vr head-mounted displays. *XRDS: Crossroads, The ACM Magazine for Students*, 22(1), 65. doi:10.1145/2837754
- Lardinois, F. (2014, June). The story behind google's cardboard project. Retrieved August 21, 2016, from <http://techcrunch.com/2014/06/26/the-story-behind-googles-cardboard-project/>
- Libow Martinez, S. & Stager, G. (2013). *Invent to learn: Making, tinkering, and engineering in the classroom*. Torrance, Calif.: Constructing Modern Knowledge Press.
- Lippman, A. (1980). Movie-maps: An application of the optical videodisc to computer graphics. In J. J. Thomas (Ed.), *Proceedings of the 7th annual*

Bibliography

- conference on Computer graphics and interactive techniques* (pp. 32–42). New York, NY: ACM. doi:10.1145/800250.807465
- Lo Presti, G., Freschi, C., Sinceri, S., Morelli, G., Ferrari, M., & Ferrari, V. (2014). Virtual Reality Surgical Navigation System for Holmium Laser Enucleation of the Prostate. In L. T. de Paolis & A. Mongelli (Eds.), *Augmented and Virtual Reality* (Vol. 8853, pp. 79–89). Lecture Notes in Computer Science. Cham: Springer International Publishing. doi:10.1007/978-3-319-13969-2_6
- Markham, T., Larmer, J., & Ravitz, J. L. (2003). *Project based learning handbook: A guide to standards-focused project based learning for middle and high school teachers* (2nd ed.). Novato, CA.: Buck Institute for Education.
- McShaffry, M. & Graham, D. (2012). *Game coding complete* (Fourth Edition). Course Technology, Cengage Learning.
- Metz, C. (2015, January). The inside story of google's bizarre plunge into vr. Retrieved April 22, 2016, from <http://www.wired.com/2015/06/inside-story-googles-unlikely-leap-cardboard-vr/>
- Nystrom, R. (2014). *Game programming patterns* (First Edition). Genever Benning.
- Motorola Nexus 6 - Full phone specifications. (2000–2016). Retrieved March 16, 2016, from http://www.gsmarena.com/motorola_nexus_6-6604.php
- Cardboard - Google VR. (2016). Retrieved September 1, 2016, from <https://vr.google.com/cardboard>
- Daydream - Google VR. (2016). Retrieved September 1, 2016, from <https://vr.google.com/daydream>
- Get Cardboard - Google VR. (2016). Retrieved September 1, 2016, from <https://vr.google.com/cardboard/get-cardboard>
- Samsung to Unveil Hum On!, Waffle and Entrim 4D Experimental C-Lab Projects at SXSW 2016. (2016). Retrieved September 5, 2016, from <https://news.samsung.com/global/samsung-to-unveil-hum-on-waffle-and-entrим-4d-experimental-c-lab-projects-at-sxsw-2016>
- Papert, S. (1986). *Constructionism: A New Opportunity for Elementary Science Education*.
- Prensky, M. (2007). *Digital game-based learning* (Paaragon House ed.). St. Paul, Minn.: Paragon House.
- Ricciardi, F., Pastorelli, E., de Paolis, L. T., & Herrmann, H. (2015). Scalable Medical Viewer for Virtual Reality Environments. In L. T. de Paolis &

Bibliography

- A. Mongelli (Eds.), *Augmented and Virtual Reality* (Vol. 9254, pp. 233–243). Lecture Notes in Computer Science. Cham: Springer International Publishing. doi:10.1007/978-3-319-22888-4_17
- Schön, S., Ebner, M., & Kumar, S. (2014). The Maker Movement. Implications of new digital gadgets, fabrication tools and spaces for creative learning and teaching. Retrieved May 6, 2016, from <http://www.openeducationeuropa.eu/de/article/The-Maker-Movement.-Implications-of-new-digital-gadgets,-fabrication-tools-and-spaces-for-creative-learning-and-teaching>
- Solomon, C. & Papert, S. (1971). Twenty Things to Do With a Computer. Retrieved May 6, 2016, from <http://www.stager.org/articles/twentythings.pdf>
- Spitzer, M. & Ebner, M. (2015). Collaborative Learning Through Drawing on iPads. In S. Carliner, C. Fulford, & N. Ostashevski (Eds.), *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications* (pp. 633–641).
- Sportillo, D., Avveduto, G., Tecchia, F., & Carrozzino, M. (2015). Training in VR: A Preliminary Study on Learning Assembly/Disassembly Sequences. In L. T. de Paolis & A. Mongelli (Eds.), *Augmented and Virtual Reality* (Vol. 9254, pp. 332–343). Lecture Notes in Computer Science. Cham: Springer International Publishing. doi:10.1007/978-3-319-22888-4_24
- Stanković, S. (2015). Virtual Reality and Virtual Environments in 10 Lectures. In A. C. Bovik (Ed.), *Synthesis Lectures on Image, Video, and Multimedia Processing* (Vol. 8, pp. 1–197). Morgan & Claypool Publishers. doi:10.2200/S00671ED1V01Y201509IVM019
- Sutherland, I. E. (1968). A head-mounted three dimensional display. In AFIPS (Ed.), *Proceedings of the December 9–11, 1968, fall joint computer conference, part I* (pp. 757–764). doi:10.1145/1476589.1476686
- Thompson, G. (2014). The Maker Movement Conquers the Classroom. *THE Journal (Transforming Education through Technology)*. Retrieved May 6, 2016, from <https://thejournal.com/Articles/2014/04/30/The-Maker-Movement-Conquers-the-Classroom.aspx?Page=2>

Appendix

Appendix A.

Sources

A.1. Actor-Component-System

AbstractActorComponent

```
1 /**
2  * Base class for all actor components.
3 */
4 public abstract class AbstractActorComponent implements
5     IActorComponent {
6     protected Actor m_Owner;
7
8     /**
9      * Sets the component owner.
10     *
11     * @param owner the owner of the component
12     */
13     public void setOwner(Actor owner) {
14         m_Owner = owner;
15     }
16
17     @Override
18     abstract public boolean init(XmlPullParser parser);
19
20     @Override
21     public void postInit() { }
22
23     @Override
24     public void onChanged() { }
```

Appendix A. Sources

```
24
25     @Override
26     public void update(int deltaMs) { }
27
28     /**
29      * Gets the component ID.
30      *
31      * @return the component ID
32      */
33     public Integer getId() {
34         return getIdFromName(getName());
35     }
36
37     /**
38      * Gets the component name.
39      *
40      * @return the component name
41      */
42     public abstract String getName();
43
44     /**
45      * Converts the given component name into the corresponding
46      * component ID.
47      *
48      * @param name the component name
49      * @return the component ID
50      */
51     public static Integer getIdFromName(String name) {
52         return name.hashCode();
53     }
```

Listing A.1: The `AbstractActorComponent` defining the basic properties and methods for every specific actor component.

Appendix A. Sources

A.2. Process System

ProcessManager

```
1 /**
2  * Singleton class for managing, updating and maintaining all
3  * processes in the engine.
4 */
5 public class ProcessManager {
6     /**
7      * The class name.
8      */
9     public static final String TAG =
10        ProcessManager.class.getName();
11
12    /**
13     * Gets the instance of the process manager.
14     *
15     * @return the process manager
16     */
17    public static ProcessManager getInstance() {
18        return m_Instance;
19    }
20
21    /**
22     * Constructor, creates the process manager.
23     */
24    private ProcessManager() {
25        m_ProcessList = new ArrayList<>();
26    }
27
28    /**
29     * Updates all processes currently in process list.
30     *
31     * @param deltaMs the time passed since last update
32     * @return the number of successful and aborted/failed
33     *         processes combined in an integer, where
34     *         the first 16 bits hold the successful count and
            the least 16 bits the failed count
```

Appendix A. Sources

```
35  /*
36  *public int updateProcesses (long deltaMs) {
37  *    int currentProcess = 0;
38  *    short successCount = 0;
39  *    short failCount = 0;
40  *
41  *    try {
42  *        while (currentProcess < m_ProcessList.size ()) {
43  *            AbstractProcess p =
44  *                m_ProcessList.get (currentProcess);
45  *
46  *            if (p.getState () ==
47  *                AbstractProcess.State.UNINITIALIZED)
48  *                p.onInit ();
49  *
50  *            if (p.getState () == AbstractProcess.State.RUNNING)
51  *                p.onUpdate (deltaMs);
52  *
53  *            if (p.isDead ()) {
54  *                switch (p.getState ()) {
55  *                    case SUCCEEDED:
56  *                        p.onSuccess ();
57  *                        AbstractProcess child =
58  *                            p.removeChild ();
59  *
60  *                        if (child != null) {
61  *                            attachProcess (child);
62  *                        } else {
63  *                            ++successCount;
64  *                        }
65  *                        break;
66  *
67  *                    case FAILED:
68  *                        p.onFail ();
69  *                        ++failCount;
70  *                        break;
71  *
72  *                    case ABORTED:
73  *                        p.onAbort ();
74  *                        ++failCount;
75  *                        break;
76  *                }
77  *                m_ProcessList.remove (p);
78  *            } else {
```

Appendix A. Sources

```
76             ++currentProcess;
77         }
78     }
79
80     return ((successCount << 16) | failCount);
81 } catch (ConcurrentModificationException ccModEx) {
82     Logger.getInstance().error(TAG, "Concurrent
83         Modification Exception");
84     return -1;
85 }
86
87 /**
88 * Gets the number of processes currently in the process list.
89 *
90 * @return the number of processes
91 */
92 public int getProcessCount() {
93     return m_ProcessList.size();
94 }
95
96 /**
97 * Aborts all processes and empties the process list.
98 *
99 * @param immediate processes are aborted immediately if true
100 */
101 public void abortAllProcesses(boolean immediate) {
102     Iterator it = m_ProcessList.iterator();
103
104     while (it.hasNext()) {
105         AbstractProcess p = (AbstractProcess)it.next();
106
107         if(immediate)
108             p.abort();
109     }
110 }
111
112 /**
113 * Attaches a new process.
114 *
115 * @param p the process to attach
116 */
117 public void attachProcess(AbstractProcess p) {
118     Logger.getInstance().info(TAG, "Attaching new Process " +
```

Appendix A. Sources

```
119     p.toString());
120     m_ProcessList.add(p);
121 }
122 public void reset() {
123     m_ProcessList.clear();
124 }
125 }
```

Listing A.2: The ProcessManager responsible for updating and managing the processes running during playing.

AbstractProcess

```
1 /**
2  * Abstract class which defines the must have behavior of a
3  * process running in the engine.
4 */
5 public abstract class AbstractProcess {
6     private State m_State;
7     private AbstractProcess m_ChildProcess;
8
9     /**
10      * The states of a process.
11      */
12     public enum State {
13         UNINITIALIZED,
14         REMOVED,
15         RUNNING,
16         PAUSED,
17         SUCCEEDED,
18         FAILED,
19         ABORTED
20     }
21
22     /**
23      * Constructor, sets the state to uninitialized : {@link
24      * State#UNINITIALIZED}.
25      */
26     public AbstractProcess() {
27         m_State = State.UNINITIALIZED;
28         m_ChildProcess = null;
29     }
30 }
```

Appendix A. Sources

```
29  /**
30   * Gets the current state.
31   *
32   * @return the state
33   */
34  public State getState() {
35      return m_State;
36  }
37
38 /**
39  * Checks if the process is alive.
40  *
41  * @return true if the process is alive, false otherwise
42  */
43  public boolean isAlive() {
44      return (m_State == State.RUNNING || m_State ==
45          State.PAUSED);
46  }
47
48 /**
49  * Checks if the process is dead.
50  *
51  * @return true if the process is dead, false otherwise
52  */
53  public boolean isDead() {
54      return (m_State == State.SUCCEEDED || m_State ==
55          State.FAILED || m_State == State.ABORTED);
56  }
57
58 /**
59  * Checks if the process was removed.
60  *
61  * @return true if the process was removed, false otherwise
62  */
63  public boolean isRemoved() {
64      return (m_State == State.REMOVED);
65  }
66
67 /**
68  * Checks if the process is paused.
69  *
70  * @return true if the process is paused, false otherwise.
71  */
72  public boolean isPaused() {
```

Appendix A. Sources

```
71     return (m_State == State.PAUSED);
72 }
73 /**
74 * Sets the state of the process to succeeded if its state is
75     running or paused.
76 */
77 public void succeed() {
78     if (m_State == State.RUNNING || m_State == State.PAUSED) {
79         m_State = State.SUCCEEDED;
80     }
81 }
82 /**
83 * Aborts the process and sets its state to aborted.
84 */
85 public void abort() {
86     m_State = State.ABORTED;
87 }
88 /**
89 * Sets the state of the process to failed if its running or
90     paused.
91 */
92 public void fail() {
93     if (m_State == State.RUNNING || m_State == State.PAUSED) {
94         m_State = State.FAILED;
95     }
96 }
97 /**
98 * Pauses the process.
99 */
100 public void pause() {
101     if (m_State == State.RUNNING)
102         m_State = State.PAUSED;
103 }
104 /**
105 * Resumes the process.
106 */
107 public void resume() {
108     if (m_State == State.PAUSED)
109         m_State = State.RUNNING;
```

Appendix A. Sources

```
113     }
114
115     /**
116      * Sets the state of the process to running.
117      */
118     protected void onInit() {
119         m_State = State.RUNNING;
120     }
121
122     /**
123      * Attaches a child process to this process.
124      *
125      * @param child the child process
126      */
127     public void attachChild(AbstractProcess child) {
128         m_ChildProcess = child;
129     }
130
131     /**
132      * Removes the child process.
133      *
134      * @return the child process or null if no child process is
135      * attached
136      */
137     public AbstractProcess removeChild() {
138         if (m_ChildProcess != null) {
139             AbstractProcess tmp = m_ChildProcess;
140             m_ChildProcess = null;
141             return tmp;
142         }
143         return null;
144     }
145     /**
146      * Updates the process.
147      *
148      * @param deltaMs the time passed since the start of the
149      * process
150      */
151     abstract public void onUpdate(long deltaMs);
152
153     /**
154      * Called when the process ended successful.
155      */
```

Appendix A. Sources

```
155     abstract public void    onSuccess();
156
157     /**
158     * Called when the process failed.
159     */
160     abstract public void    onFail();
161
162     /**
163     * Called when the process was aborted.
164     */
165     abstract public void    onAbort();
166
167     /**
168     * Prints process informations.
169     *
170     * @return the process information.
171     */
172     @Override
173     abstract public      String toString();
174 }
```

Listing A.3: The base class for every process able to run in the engines process system.

A.3. Event System

EventManager

```
1 /**
2  * Singleton class, implements the event manager interface and
3  * acts as the global event manager
4  */
5 public class EventManager implements IEventManager {
6     /**
7     * The class name.
8     */
9     public static final String TAG = EventManager.class.getName();
10
11    private static EventManager m_Instance = new EventManager();
12
13    LinkedList<IEventData>          m_EventQueue;
14    Map<UUID, List<IListenable>>   m_EventListenerMap;
15
```

Appendix A. Sources

```
16  /**
17  * Constructor.
18  */
19  private EventManager() {
20      m_EventQueue      = new LinkedList<>();
21      m_EventListenerMap = new HashMap<>();
22  }
23
24 /**
25 * Gets the instance of the event manager.
26 *
27 * @return the event manager
28 */
29 public static EventManager getInstance() {
30     return m_Instance;
31 }
32
33 @Override
34 public boolean addListener(IListenable component, UUID
35 eventType) {
36     List check = m_EventListenerMap.get(eventType);
37
38     if (check == null) {
39         check = new ArrayList();
40         check.add(component);
41         m_EventListenerMap.put(eventType, check);
42     } else {
43         ArrayList<IListenable> listeners = new
44             ArrayList<>(check);
45         if (listeners.contains(component)) {
46             Logger.getInstance().warning(TAG, "Attempting to
47                 double-register a listener!");
48             return false;
49         }
50         listeners.add(component);
51         m_EventListenerMap.put(eventType, listeners);
52     }
53     return true;
54 }
55
56 @Override
57 public boolean removeListener(IListenable component, UUID
58 eventType) {
59     List check = m_EventListenerMap.get(eventType);
```

Appendix A. Sources

```
56     boolean success = false;
57
58     if (check == null) {
59         Logger.getInstance().warning(TAG, "Event type not
60             available!");
61         return success;
62     }
63     ArrayList<IListenable> listeners = new ArrayList<>(check);
64     for (Iterator<IListenable> it = listeners.iterator();
65         it.hasNext(); ) {
66         IListenable listener = it.next();
67
68         if (component == listener) {
69             listeners.remove(listener);
70             success = true;
71             break;
72         }
73     }
74     m_EventListenerMap.put(eventType, listeners);
75     return success;
76 }
77
78 @Override
79 public boolean triggerEvent(IEventData eventData) {
80     boolean processed = false;
81     List check = m_EventListenerMap.get(
82         eventData.getEventType());
83
84     if (check == null) {
85         Logger.getInstance().warning(TAG, "Event type not
86             available!");
87         return processed;
88     }
89     ArrayList<IListenable> listeners = new ArrayList<>(check);
90     for (Iterator<IListenable> it = listeners.iterator();
91         it.hasNext(); ) {
92         IListenable listener = it.next();
93         listener.invoke(eventData);
94         processed = true;
95     }
96     return processed;
97 }
98
99 @Override
```

Appendix A. Sources

```
95  public boolean queueEvent(IEventData eventData) {
96      List check = m_EventListenerMap.get(
97          eventData.getEventType());
98
99      if (check == null) {
100         Logger.getInstance().warning(TAG, "Event type not
101             available!");
102         return false;
103     }
104
105     @Override
106     public boolean abortEvent(IEventData eventData, boolean
107         allOfType) {
108         boolean success = false;
109         ArrayList<IListenable> listeners = new
110             ArrayList<>(m_EventListenerMap.get(
111                 eventData.getEventType()));
112
113         if (listeners == null) {
114             Logger.getInstance().warning(TAG, "No events in for
115                 the specified event type in queue!");
116             return success;
117         }
118
119         int i = 0;
120         synchronized (m_EventQueue) {
121             while (i < m_EventQueue.size()) {
122                 IEventData queuedEvent = m_EventQueue.get(i);
123                 if (queuedEvent.getEventType() ==
124                     eventData.getEventType()) {
125                     m_EventQueue.remove(queuedEvent);
126                     success = true;
127
128                     if (!allOfType)
129                         break;
130                 }
131             }
132         }
133         return success;
134     }
135
136     @Override
```

Appendix A. Sources

```
132     public boolean updateEvents() {
133         boolean updated = true;
134         IEventData event = m_EventQueue.poll();
135
136         while (event != null) {
137             ArrayList<IListenable> listeners = new
138                 ArrayList<>(m_EventListenerMap.get(
139                     event.getEventType()));
140
141             for (int i = 0; i < listeners.size(); ++i) {
142                 IListenable listener = listeners.get(i);
143                 listener.invoke(event);
144             }
145             event = m_EventQueue.poll();
146         }
147
148         /**
149          * Resets the event listeners and the event queue.
150         */
151         public void reset() {
152             m_EventQueue.clear();
153             m_EventListenerMap.clear();
154         }
155 }
```

Listing A.4: The source code of the event manager.

AbstractEventData

```
1 /**
2  * Base event implementing the event data interface.
3  * <p>
4  * All events used in the engine must inherit from the abstract
5  * base event.
6  */
7 public abstract class AbstractEventData implements IEventData {
8     private float m_TimeStamp;
9
10    /**
11     * Default Constructor.
12     */
```

Appendix A. Sources

```
13  public AbstractEventData() {
14      m_TimeStamp = 0;
15  }
16
17  /**
18  * Constructor.
19  *
20  * @param timeStamp
21  */
22  public AbstractEventData(float timeStamp) {
23      m_TimeStamp = timeStamp;
24  }
25
26  /**
27  * Gets the event type.
28  *
29  * @return the event type (UUID)
30  */
31  @Override
32  abstract public UUID getEventType();
33
34  /**
35  * Gets the time stamp of the event.
36  *
37  * @return the time stamp
38  */
39  @Override
40  public float getTimeStamp() {
41      return m_TimeStamp;
42  }
43
44  /**
45  * Gets the name of the event.
46  *
47  * @return the event name
48  */
49  @Override
50  public String getName() {
51      return "BaseEvent";
52  }
53 }
```

Listing A.5: The base class every event has to inherit from to fit into the event system.

A.4. Rendering System

SceneNode

```

1 /**
2 * Basic Scene Node.
3 */
4 public class SceneNode implements ISceneNode {
5     protected ArrayList<ISceneNode> m_Children;
6     ISceneNode m_Parent;
7     SceneNodeProperties m_Properties;
8
9 /**
10 * Constructors removed for illustration
11 */
12
13 /**
14 * Sets the parent of this node.
15 *
16 * @param parent the parent node
17 */
18 public void setParent(ISceneNode parent) {
19     m_Parent = parent;
20 }
21
22 @Override
23 public SceneNodeProperties getSceneNodeProperties() {
24     return new SceneNodeProperties(m_Properties);
25 }
26
27 @Override
28 public void setTransform(Matrix4f toWorld, Matrix4f
29     fromWorld) {
30     m_Properties.setTransform(toWorld, fromWorld);
31 }
32
33 @Override
34 public int onUpdate(Scene scene, long elapsedMs) {
35     Iterator<ISceneNode> it = m_Children.iterator();
36
37     while (it.hasNext()) {
38         ISceneNode next = it.next();
39         next.onUpdate(scene, elapsedMs);

```

Appendix A. Sources

```
39         }
40     return 1;
41 }
42
43 @Override
44 public boolean isVisible(Scene scene) {
45     boolean visible = false;
46
47     for (ISceneNode node : m_Children) {
48         if (node.isVisible(scene)) {
49             visible = true;
50             break;
51         }
52     }
53     return visible;
54 }
55
56 @Override
57 public int renderChildren(Scene scene) {
58     Iterator<ISceneNode> it = m_Children.iterator();
59
60     while (it.hasNext()) {
61         ISceneNode next = it.next();
62
63         if (next.preRender(scene) == 1) {
64             if (next.isVisible(scene)) {
65                 double alpha = next.getSceneNodeProperties()
66                     .getMaterial().getAlpha();
67
68                 if (alpha == 1.0) {
69                     next.render(scene);
70                 } else {
71                     Matrix4f transformation =
72                         scene.getTopMatrix();
73                     Vector4f worldPos = new Vector4f();
74                     transformation.getColumn(3, worldPos);
75                     Matrix4f camFromWorld =
76                         scene.getCamera()
77                             .getSceneNodeProperties()
78                             .getFromWorld();
79                     Vector4f screenPos = new Vector4f();
80                     screenPos.mul(camFromWorld);
81
82                     AlphaSceneNode alphaNode = new
```

Appendix A. Sources

```
79             AlphaSceneNode( next , transformation ,
80                         screenPos .z );
81             scene .addAlphaSceneNode(alphaNode) ;
82         }
83     }
84     next .renderChildren( scene ) ;
85 }
86 return 1;
87 }
88
89 @Override
90 public int render(Scene scene) {
91     return 1;
92 }
93
94 @Override
95 public int postRender(Scene scene) {
96     scene .popMatrix();
97     return 1;
98 }
99
100 @Override
101 public boolean addChild(ISceneNode child) {
102     if (child == null)
103         return false;
104
105     if (!m_Children.add(child))
106         return false;
107
108     ((SceneNode) child) .setParent( this );
109
110     return true;
111 }
112
113 @Override
114 public boolean removeChild(Integer actorId) {
115     Iterator<ISceneNode> it = m_Children.iterator();
116     boolean childRemoved = false;
117
118     while (it.hasNext() && !childRemoved) {
119         ISceneNode next = it .next();
120         if (next .getSceneNodeProperties() .getActorId () ==
```

Appendix A. Sources

```
121         actorId) {
122             next.removeChild(actorId);
123             it.remove();
124             childRemoved = true;
125         }
126     return childRemoved;
127 }
128 /**
129 * Sets the node's visibility.
130 * <p>
131 * Setting a node visible or invisible also affects the nodes
132 * children.
133 * </p>
134 *
135 * @param visible <code>true</code> if the node and its
136 * children should be visible, <code>false</code> otherwise
137 */
138 public void setVisible(boolean visible) {
139     for (ISceneNode child : m_Children) {
140         SceneNode node = (SceneNode)child;
141         node.setVisible(visible);
142     }
143 }
```

Listing A.6: A shortened source code of base class for every node the scene graph can contain.

Scene

```
1 /**
2 * The Scene class.
3 */
4 public class Scene implements ILISTenable {
5     ISceneNode m_Root;
6     CameraNode m_Camera;
7     MatrixStack           m_MatrixStack;
8     List<AlphaSceneNode> m_AlphaSceneNodes;
9     Map<Integer, ISceneNode> m_ActorMap;
10    LightManager          m_LightManager;
11    FrustumCuller         m_FrustumCuller;
12    ITheme                m_CurrentTheme;
```

Appendix A. Sources

```
13
14  /**
15   * Default Constructor.
16   */
17  public Scene() {
18      m_Root          = new RootNode();
19      m_MatrixStack   = new MatrixStack(50);
20      m_LightManager  = new LightManager();
21      m_ActorMap      = new HashMap<>();
22      m_FrustumCuller = new FrustumCuller();
23      m_CurrentTheme  = new StandardTheme();
24
25      // add listeners to event manager
26      IEventManager eventManager = EventManager.getInstance();
27      eventManager.addListener(this,
28          NewRenderComponentEvent.EVENT_ID);
28      eventManager.addListener(this,
29          ActorDestroyedEvent.EVENT_ID);
29      eventManager.addListener(this,
30          DisableActorEvent.EVENT_ID);
30      eventManager.addListener(this, EnableActorEvent.EVENT_ID);
31      eventManager.addListener(this,
31          NewAnimationEvent.EVENT_ID);
32      eventManager.addListener(this, SwitchThemeEvent.EVENT_ID);
33  }
34
35  /**
36   * Getter and Setters removed for illustration
37   */
38
39  /**
40   * Renders the scene with all children
41   *
42   * @return <code>1</code> if successful, <code>0</code> else
43   */
44  public int onRender() {
45      if (m_Root != null && m_Camera != null) {
46          m_Camera.setView(this);
47          m_LightManager.calculateLighting(this);
48
49          if (m_Root.preRender(this) == 1) {
50              m_Root.render(this);
51              m_Root.renderChildren(this);
52              m_Root.postRender(this);
53      }
54  }
```

Appendix A. Sources

```
53         }
54     }
55
56     renderAlphaPass();
57
58     return 1;
59 }
60
61 /**
62 * Adds a child with <code>actorId</code> to the scene.
63 *
64 * @param actorId the actor id of the child
65 * @param child the child to be added
66 * @return <code>true</code> if the child was added,
67 *         <code>false</code> otherwise
68 */
69 public boolean addChild(Integer actorId, ISceneNode child) {
70     if (actorId != Constants.INVALID_ID) {
71         mActorMap.put(actorId, child);
72     }
73     if (child instanceof LightNode) {
74         if (mLightManager.getLightCount() + 1 <
75             Constants.MAX_LIGHTS_SUPPORTED)
76             mLightManager.addLight(child);
77     }
78     return mRoot.addChild(child);
79 }
80
81 /**
82 * Removes the actor with the given actor id from the scene.
83 *
84 * @param actorId the actor id
85 * @return <code>true</code> if the actor was removed,
86 *         <code>false</code> otherwise
87 */
88 public boolean removeChild(Integer actorId) {
89     if (actorId == Constants.INVALID_ID)
90         return false;
91
92     ISceneNode child = findActor(actorId);
93
94     if (child instanceof LightNode)
95         mLightManager.removeLight(child);
```

Appendix A. Sources

```
94     m_ActorMap.remove(actorId);  
95  
96     return m_Root.removeChild(actorId);  
97 }  
98  
99 /**  
100 * Pushes a new world matrix onto the matrix stack.  
101 *  
102 * <p>  
103 * The given <code>toWorld</code> matrix is pushed onto the  
104 * stack and multiplied  
105 * with the previous top matrix of the stack to gain correct  
106 * transformation matrices for  
107 * a specific node in the scene.  
108 * </p>  
109 *  
110 * @param toWorld the new to-world transformation matrix  
111 */  
112 public void pushAndSetMatrix(Matrix4f toWorld) {  
113     m_MatrixStack.pushMatrix();  
114     m_MatrixStack.multMatrix(toWorld);  
115 }  
116  
117 /**  
118 * Pops the current transformation matrix from the stack.  
119 */  
120 public void popMatrix() {  
121     m_MatrixStack.popMatrix();  
122 }  
123  
124 @Override  
125 public void invoke(IEventData eventData) {  
126     /**  
127         * Event handling was removed for illustration  
128     */  
129 }
```

Listing A.7: The shortened Scene class used to traverse the scene graph and render the scene.