Master's Thesis

The Use of Modern Controller Devices at Schools: Game-Based Learning with the Leap Motion

Norbert Spot

Institute for Information Systems and Computer Media (IICM),
Graz University of Technology
A-8010 Graz, Austria

Supervisor: Assoc.Prof. Dipl.-Ing. Dr.techn. Martin Ebner

Graz, November 2014

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, (signature)

Acknowledgments

I would like to express my gratitude to my supervisor Assoc.Prof. Martin Ebner for his support and guidance during the work on this thesis.

I also wish to express my thanks to Silvana Aureli, without her help the small trial at the elementary school would not have been possible.

I am also extremely grateful to my parents, who always support and stand behind me.

Suzy, thank you for always pushing me to do better.

Norbert Spot, November 2014

Abstract

This thesis aims to present the current state of art in the field of *gamification* and *game-based learning*. A prototype is presented, which tries to show a possible way, how new technology devices might improve the learning process. We describe the development process of the application and the device for which it has been developed. Our device of choice is the Leap Motion Controller, a 3D infrared hand and finger-tracking sensor. The tools, which have been used to create the application, are detailed as well. Finally, the findings of a small-scale trial at an elementary school are discussed. The outcomes of the trial show, that kids really like the prototype and they had a lot of fun playing it.

Keywords: Natural User Interface, gamification, game-based learning, Leap Motion, controller, unity3d, hand tracking, tracking device, infrared camera, 3D camera, application

Kurzfassung

Diese Arbeit versucht den aktuellen Stand der Forschung im Bereich von *Gamification* und *Game-Based Learning* zu präsentieren. Mit der Hilfe eines Prototyps soll gezeigt werden, wie neue Technologien den Lernprozess optimieren könnten. Wir beschreiben den gesamten Entwicklungsprozess, so wie das verwendete Gerät. Es handelt sich dabei um ein Leap Motion, welches ein 3D-Infrarot-Sensor besitzt für Hand- und Finger-Tracking. Im Rahmen der Arbeit wird ein 3D-Spiel entwickelt, welches die mathematischen Grundrechnungsarten üben helfen soll. Der Prototyp wird abschließend an einer Volksschule eingesetzt und es zeigt sich, dass die Schülerinnen und Schüler viel Freude bei der Verwendung haben.

Stichwörter: Natural User Interface, Gamification, Game-Based Learning, Leap Motion, Controller, Unity3d, Hand tracking, tracking Gerät, Infrarot Kamera, 3D Kamera,

Table of Contents

1 Introduction

Computers have evolved rapidly and become part of people's everyday life. They spread into nearly every field of industry and entertainment. People use computers almost everywhere, at work, at home, at schools and universities. Usually if people talk about using a computer, we imagine someone sitting at a table, typing on a keyboard, moving around with the mouse or tapping the touchpad and staring at the computers screen. The way we interact with computers has not changed significantly since the 1960's when these peripherals were invented.[1] This is slowly changing though. Today, there are more and more innovative input devices coming out every year. They are different, and they aim to change the way we interact with different kinds of computers. These peripheries though, do not intend to replace the traditional keyboard-mouse setup. They are an addition, useful in diverse applications. Some of them represent a more natural way of human-computer interaction. Utilising such devices, developers can create applications, which instead of getting input by a keyboard and mouse, can track movement in front of the computer's screen, or sense touch. This more natural way of human-computer interaction enables a Natural User Interface (NUI).

The NUI has already become popular in different kinds of mobile devices like smartphones and tablets in the form of multi-touch displays. The NUI is actually one of the reasons why these devices became so popular. They are more natural to use. One can touch the screen to select items, manipulate images or other multimedia content. The real haptic feedback is probably the only thing, which is missing here (yet). However, the evolution of sensing technology has increased in the past few years. Technological advances in computer vision enabled computers to track the movements of the human body.

The goal of this thesis is to make use of a device, which enables a NUI, and show a way how it could possibly improve, and make the learning process more

[1] http://en.wikipedia.org/wiki/History_of_the_graphical_user_interface (last visited 2014.8.7)

interesting, more fun and more natural. Therefore, a prototype of an application has to be developed, which is intuitive enough to be easily used by children in an elementary school. But at the same time, it had to be challenging enough to have an effect on their learning behaviour. For this purpose, different devices were considered, and the one, which has been chosen for this thesis is the *Leap Motion Controller*, announced in 2012 and released in July 2013. With the help of its *Software Development Kit* (SDK) and proper software design, it is capable to provide a really natural *User Interface* (UI).

2 Technology and Schools

Computers and technology are in use in almost every aspect of people's life nowadays. Technology helps people to do their job better, faster and perhaps sometimes easier then before. It is reasonably to believe, that technology should help to improve teaching and learning in schools too. However, simply providing access to teachers and students to technology doesn't necessarily ensure that it will enhance teaching and learning and result in improvements. The rapid growth of technology in every field has helped computers to become an everyday tool for teachers and children at school. Most of the students have now access to computers and the Internet not only at home but also at the school, or even in the classroom. For students without home Internet access, many schools allow students to use computers outside of regular school hours. According to Noeth & Volkov (2004) however, schools may not always use computers in the best way to enhance learning. In their report about the effectiveness of technology in schools they mentioned, that technology may help organise and provide structure for material to students, that it can help students, teachers and parents to interact anytime and anywhere. Students can use computers to simulate, visualise, and interact with scientific structures, processes, and models. Experts believe that to make technology more widely spread and used at schools in different classes, we have to enhance the technology skills of teachers and administrators (Noeth & Volkov, 2004). One of the most critical elements of technology use, is how well are those who use it prepared and how is their skill level. According to Means (2000), teachers should have at least basic technology skills and be able to use technology for their personal productivity and to support learning of a given subject. Teachers should be able to design and adapt learning activities supported by technology. As Noeth & Volkov (2004) state, in the United States, for example, many schools have taken first measures to provide guidelines for how to use educational technology more effectively, and the majority of these schools have developed standards for teachers and administrators that include technology. Technology at schools has first been used as an instructional delivery medium, but it

has become an integral part of the learning environment as whole. Noeth & Volkov (2004) list four distinct purposes how technology is serving at schools:

- To teach, drill, and practice using increasingly sophisticated digital content.
- To provide simulations and real world experiences to develop cognitive thinking and to extend learning.
- To provide access to a wealth of information and enhanced communications through the Internet and other related information technologies.
- As a productivity tool employing application software such as spread sheets, databases, and word processors to manage information, solve problems, and produce sophisticated products.

Despite schools started to utilise computers heavily, the evidences are mixed as to whether overall student performance has increased notably, or the achievement gap has visibly narrowed as a result. Research reviews collected by Noeth & Volkov (2004) for their report have generally concurred, that the use of computers combined with traditional instructions can increase the students learning performance in the traditional curriculum and basic skills area. Furthermore, it produces higher academic achievement in many subject areas then do traditional instructions alone. The reviews reveal, that with the use of computers, students learn more quickly and with greater retention, they like learning with computers, and their attitude towards school and learning in general is positively affected. Dede (2002) mentioned that the important thing about the effectiveness of learning with technology is not how sophisticated the technology itself is, but the way its capability motivates its users.

3 Gamification and Game Based Learning

Over the past few years, the term *gamification* has gained significant attention and has raised a lot of interest in industry, as well as on academic ground. As Huotari (2012) points it out, the term 'Gameification' was first used in 2008 in a blog post[2] by Brett Terrill. He described it as 'taking game mechanics and applying them to other web properties to increase engagement.' The term has finally achieved significant attention in the second half of 2010 and it got its current form 'gamification'. In its meaning, it refers to 'the use of game design elements in a non-game contexts' (Deterding et al., 2011) with the goal of making the given task more engaging, more fun and more motivating. The Oxford Dictionary of English (British English) describes the term as following: 'the application of typical elements of game playing (e.g. point scoring, competition with others, rules of play) to other areas of activity, typically as an online marketing technique to encourage engagement with a product or service: *gamification is exciting because it promises to make the hard stuff in life fun*.' These game elements are typically not the centre of the system, but they tend to motivate the user to actually use it. To say it in an easy language, *gamification* is the use of different achievements, badges, stars, points etc. and various leader boards where one can compare his or her performance with the others. If we use these elements in any context, we *gamify*.

There is a good example to the use of *gamification* coming up in numerous research papers, which describes exactly what the term is about. It is the service called Foursquare. Foursquare is a social network, which enables its users to check-in in different venues, places, parks, cities or even buildings, literally anywhere where an entry has been created for a place, or the user can create a new one based on his or her position. The check-in is only allowed though, when the user actually is at the given place, which is determined by the smartphones GPS (Global Positioning System) data. Being a social network, the user can add friends, and can see where they are,

[2] http://www.bretterrill.com/2008/06/my-coverage-of-lobby-of-social-gaming.html (last visited 2014.10.28)

where they check-in. The check-ins become status updates, where the user can add a short text, add friends he or she is with, and post the status to other social networks like Facebook or Twitter. For every check-in, the user earns points. And here comes *gamification* in. There are bonus points if the user checks in at a place where his or her friends have never been to yet (never been to as never checked in at). Moreover, the user can earn badges for the check-ins. For example he or she earns a badge for checking in at 5 different airports. The badge can be then added to the status so that others can see it too. There are charts presented to the user after some check-ins, which inform about how the users' check-ins perform comparing to his or her friends. This point and badge reward system encourages users to make more and more check-ins. People want their friends to know what a cool place are they at. It may start a competition among some friends, as one wants to do better as the other one, wants to check-in on cooler places, earn fancy badges as first amongst friends. This gamified system makes an otherwise simple product much more fun to use. It motivates it's users to check-in at that other coffee shop, restaurant, train station and so on, to be awarded with a badge or with a higher rank. The motivation to check-in at different places leads to visiting and discovering new places by the user. Furthermore, users cannot only check-in, but they can leave tips on places for others. If someone has tried a new tasty meal at a restaurant for example, he or she can leave a tip, something like a small review on that given restaurants page. These reviews are then shown for other users and so they can get an idea about the given venue, and maybe also decide whether it is worth to go there. These venues can then award their customers who check in at the place with e.g. discounts or other awards. This is another *gamified* marketing step, which motivates the user to use the service. The fact, that Foursquare uses a localisation system to allow users to check-in means, that it has to know the location of places. This led to becoming a large database of different points of interest. Places are shown on a map and people can search for any place Foursquare knows about. Let's say users can find the closest Thai restaurant or burger joint or any other place in Foursquare's database. The service has been launched in 2009 and it quickly became a popular toy for people to show off where they are, and a tool for

people to find interesting places nearby. Up to date, there was over 6 billion check-ins made by more than 50 million users around the world.[3] Foursquare has used proven game mechanics to make their service more engaging. Just like successful commercial big name video games, which attract users because they are fun to play and engaging. We definitely can not state, that Foursquare has become so popular only thanks to its *gamified* system, at least we have not found any scientific proof of that, but it certainly had a huge influence on the services' popularity.

Foursquare has shown, that there might be a place for *gamification* in a lot of different non-game contexts of everyday life. In our work we focus on *gamification* in educational context, where teachers, just like e.g. video games, also try to motivate their students to learn more to perform better. As Magerko et al. (2008), point out: 'Students who are more motivated are more likely to learn'. This motivation can be achieved in different ways, one of which is the use of various learning games in classes.

It might be the same, or similar in other countries too, but in Slovakia, first grade elementary school students do not get marks for their performance. To motivate kids to learn, and to get used to another kind of a reward system, they get stamps if they perform well. Kids will not get a lower mark if their performance does not meet the requirements, they just will not get any stamp, or will get a stamp with a lower rank. These stamps are usually two with some not abusing pictures of animals. If a kid does not get a stamp for its let's say homework, it will motivate him or her to perform better next time to get a reward in the form of a stamp, which it can then show to the parents. This form of motivation appears to work well, and prepares kids for the regular reward system of marks. It is not easy though, to keep up with this motivation, to hold interest in somewhat higher grades.

No matter how popular *gamification* has become in the past few years, according to some research studies, it is not the solution to make activities more motivating and engaging. The use of a gamifying concept does not necessarily mean success right away. As Rojas et al. (2013) state 'there are many examples where gamification is ineffective and does not lead to an increase in motivation and

[3] http://foursquare.com/about (last visited 2014.10.9)

ultimately productivity/performance.' This fact was observed by Montola et al. (2009) on a photo sharing service. After adding game design elements in the form of achievements, most of the users disliked this kind of change to the service and showed indifference towards the intervention. According to Montola et al. such bad influence on the users' motivation has been attributed to their past experiences with games in addition to the not ideal design of gamification elements. From this fact we can conclude, that it is not the gamification's fault as a principle, but a bad design strategy. Mollick et al. (2013) stated that a properly designed gamification strategy certainly can add value, and it supports the given activity, which is being gamified. When a gamification strategy fails, the credit for it more than likely goes to the poor or inadequate design decision.

Gamification is something relatively new and it is used in a large number of different contexts. When analysing its implementation it is hard to define those factors, which make gamification to actually work. It is hard to define those factors as well, which need to be adjusted to always get that positive effect of gamification, which we wanted. According to Deterding (2011), this is due to the fact, that gamified systems are kind of hybrid systems. They are neither a pure functional software for example, nor a full-fledged game. Therefore, there are currently no well-established principles for designing them. To standardise the development of a gamified system Rojas et al. (2013) propose a gamification framework inspired by the framework created by the Medical Research Council (2000). The framework of MRC aims to improve health by introducing a system for designing and evaluating complex interventions. It was developed to help replicate and adapt medical interventions in different medical contexts. According to the MRC, their framework 'should not be read as an inflexible to do list'. It should be taken as an advice. Since the creation of the framework, it has been used with success in many different health related fields.

Rojas et al. (2013) propose two types of gamification interventions. The first one is *gamification for development* and the other one *gamification for enhancement*. Gamification for development refers to the use of gamification while creating a new tool or platform to be used for a given purpose. Gamification for enhancement on the other hand, refers to the enhancement of an already well-established activity. The aim of this type of gamification is to make a normal activity more engaging, more fun and

to motivate the person involved performing better and ultimately to the best of his or her ability.

Rojas et al. took the MRC framework, and adapted it in that way that it suits the needs of gamification research. Their final proposed framework consists of four stages:

Stage A (Theory and Modelling): This stage consists of two phases. In the first one the one who uses the framework should decide the type of gamification intervention (e.g. gamification for development, gamification for enhancement), so the purpose of research should be defined. After the definition of the purpose of the work, a literature research is necessary to be able to emphasise why the gamification of the given field would be beneficial. According to Rojas et al. (2013) this stage of research is often overlooked 'due to the fact that investigators believe that given the novelty of the field, newly designed interventions will not have any reference or evidence to be supported by'. Many complex interventions however have not been investigated yet, therefore extending the research to other related fields, where similar interventions may have been used before, makes sense to prevent the development of tools that already exist, or to prevent attempts to enhance activities that are already effective enough, so that gamifying them will not bring any benefits. The modelling process of this stage combines the given context and the evidences found in the first phase to design the best intervention.

Stage B (Piloting): This stage can be divided into more steps which begin with analysing of the feasibility, acceptability and cost-effectiveness of the proposed intervention, and end with a pilot-study which should uncover any methodological issues.

Stage C (Evaluation): In this stage, the effectiveness of the intervention designed in the previous stages is determined. This information is then used to justify the real world implementation of the given intervention. Rojas et al. (2013) have observed that 'research within the gamification field often lacks real justification to implement the designed intervention or created tool in the real world based on objective results'.

Stage D (Implementation): Final stage. In this stage, the intervention developed in the first three stages is implemented in real world. At this stage however,

designers of the gamification intervention should be still careful, as many problems may arise when a system from a controlled environment of an evaluation study is transferred into the real world.

According to Rojas et al. (2013) their proposed framework should be considered when designing a new gamification system either to improve or enhance an existing activity, or developing a gamified application. They believe in the validity of the proposed framework, however it is still difficult to document, evaluate and replicate the outcomes of interventions within different contexts, therefore it cannot be stated that this approach works as expected in every field.

Deterding et al. (2011) in the definition of gamification use the term 'game elements'. What is the definition of *game elements* one cloud ask, or even what the definition of a *game* is. Reeves & Read (2009) identified the 'Ten Ingredients of Great Games' as following: Self representation with avatars; three-dimensional environments; narrative context; feedback; reputations, ranks, and levels; marketplaces and economies; competition under rules that are explicit and enforced; teams; parallel communication systems that can be easily configured; time pressure. All of these elements however, can be found outside of games too. Standing alone, none of them would be identified as gameful or game specific. These are not requirements though. Games differ vastly. Game elements like avatars for example, may be common for action or role-playing games, but not necessarily for other types of games.

By introducing game elements to an activity or a system in any context, we gamify the given context. But as well as there are no methods yet for developing a gamified system, there are also no well-established methods for evaluating, testing their effectiveness. Hamari et al. (2014) attempted to analyse the effectiveness of gamification by going through 24 studies on gamified systems, a majority of which were done in an educational context. They found out, that the majority of studies reported positive influence of gamification on the given context, and while these findings seem to be promising, they are still sceptical and point out, that these results solely rely on user evaluation and some studies lack on control groups, meaning

actual influence on behaviour has not been examined. Hamari et al. (2014) propose that future studies on gamification should be done using more rigorous methods. Given that the majority of studies was conducted in an educational context, it remains unclear how effective gamification is in other contexts. This fact however has been tried to be cleared by Zuckerman et al. (2014), who attempted to evaluate the effectiveness of gamification in the context of physical activity. They have developed a research prototype called 'StepByStep' an accelerometer based application, a step counter, which aims to motivate its users to bring more physical activity, more steps into their everyday life. They state, that presumably, gamification does make physical activity more enjoyable and motivates users to be more active. 'However, due to contradicting findings from prior studies, and lack of systematic research in the field, this assumption cannot be supported by the existing literature.' (Zuckerman et al., 2014)

Barata et al. (2013), in a study, which took five years to accomplish, attempted to explore how gamification could be applied to education to improve student engagement. In their study they gamified a college course by including experience points, levels, badges, challenges and leader boards, typical game elements. The first three years of the study the course was non-gamified. The last two years were gamified. To find out what an impact gamification had on learning experience, they compared data from gamified and non-gamified years using different performance measures. In their study, they aimed to answer questions like '*How did the gamified experiment affect the grades?*' or '*How was student engagement affected by the second gamified edition of the course?*' After asking students for feedback, Barata et al. found out, that the opinions of students were consistent. They found the gamified course more interesting and motivating than other courses, and that it would be a good idea to extend the gamified experience to other courses. Barata et al. state, that students were mildly feeling they were playing a game. That means there is still room for improving the game-like feeling. With the gamified course, students seem to score better on grades, and the differences between their grades seem to decrease. Lecture attendance seems to be unaffected though. The results also showed, that students participated more, and were more active in the forums. They also paid more attention to the lectures' slides, which according to Barata et al. suggests deeper engagement.

Their study also suggests, that even distribution of challenges over the term and their fair rewarding might significantly improve student participation and performance.

Although some studies, due to the lack of real scientific evaluation may be still sceptical about the improvements and the influence of gamification on the fields it is used in, based on the positive outcomes of numerous research studies, and the success of services like Foursquare, we believe that gamification has a great potential and can significantly improve the learning experience, as well as it can bring a benefit in other fields or contexts too.

3.1 Game-Based Learning

When gamification could improve the learning experience introducing game elements into the educational process, how is it with introducing games to the learning process? There are numerous research studies, which refer to the term *game-based learning*. To understand what this term is about and how could this improve or make learning more interesting, we can begin with the definition of the 'classic game model' by Juul (2015): 'A game is a rule-based formal system with a variable and quantifiable outcome, where different outcomes are assigned different values, the player exerts effort in order to influence the outcome, the player feels attached to the outcome, and the consequences of the activity are optional and negotiable.' Previous studies have shown, that playing games may support the development of certain strategies and skills such as problem-solving, decision-making, understanding complex systems, planning or data handling, no matter if they were especially made for educational purposes or not. We have to bear in mind though, that there is a significant difference between games for entertainment and games for learning. As Magerko et al. (2008) point it out, players of entertainment games have the choice to play games which suit their taste, while students who use digital games for learning typically do not have a large number of games they could choose from. Moreover, games made for educational purposes are typically not made by big name game design studios, may have lower quality, thus they may not really be engaging.

Magerko et al. (2008) contend that the need of different game types of different students can be addressed by developing adaptive games, which would adapt to the players' gaming style and learning needs. In their study they present a prototype of such a game, which targets the teaching of microbiology. The prototype is called 'Super Covert Removal of Unwanted Bacteria', or S.C.R.U.B. for short, and it 'is a mini-game that is designed to teach principles about ways of preventing the spread of microbial pathogens.' The prototype game teaches concepts like how to most effectively remove microbes from our hands, how soap, anti-biotic soap and alcohol-based hand sanitisers work on microbes or prevention strategies for avoiding infections. To better suit the needs of different learner types of students, Magerko et al. designed the game the way that players can be assigned a game adaptation in multiple ways. The first one offers an initial customisation screen to the players, who can then choose their preferred style of interface. Another approach asks the players to complete a questionnaire with multiple-choice questions, which according to Magerko et al. are designed 'to help provide evidence of preferred learning and play experiences for the individual player'. Once the player completes the questionnaire, he or she can begin to play. Magerko et al. state that the 'preliminary results from S.C.R.U.B. show promise, but hardly hearken to a rigorous educational evaluation'. They point out that research in games for learning should be treated as rigorously as approaches in more traditional education research. Therefore, they plan to do a serious study on the pedagogical benefits of their prototype.

Numerous studies commonly assume, that computer games are a useful tool for education because students find them motivating. Nicola Whitton (2007) questions this assumption. In her paper, she describes a study on motivational potential of using computer games in education, in this case in higher education. To explore students' motivation for game playing for leisure and study, she carried out a series of in-depth interviews. These were followed by a survey examining students' motivation to play, and to learn with games. The results of the study are particularly interesting and they reflect the findings of our small-scale evaluation of our prototype described in a later chapter. The results show, that it is clearly not the case that all the students find games for learning or game-based learning motivational. However, despite this fact, the perceptions of game-based learning were positive, even from those who considered

themselves non-game players, says Whitton. All the participants reported, that they would consider the idea of learning with games if it would be the most effective way to learn something. According to Whitton (2007), games have the potential to be effective environments for learning, if they are experiential, active, problem-based and collaborative.

Opinions, whether gamification and game-based learning is motivational enough to be scientifically approved, thanks to its positive influence on the learning process, differ. Whitton (2007) points out, that 'simply to rely on the fact that games are motivational is not in itself a sufficient rationale for using a game' for learning. The majority of studies however in the end conclude, that gamification, game-based learning or learning with games does, or they at least say that it might improve the learning process. In the most recent report we found, that Mark Griffiths (2014) claims that 'playing video games is good for your brain', and he explains how. According to him, the purported negative effects of video games like addiction, increased aggression or health consequences like obesity get far more coverage then their benefits. There is however an increasing number of studies which show that games can be integrated into education or which reveal how games can improve reaction times, hand-eye coordination skills or even spatial visualisation ability, such as mentally rotating and manipulating two and three dimensional objects. Griffiths (2014) also refers to a study in the Proceedings of the National Academy of Sciences by Bejjanki et al., which shows that the playing of action video games has a positive influence on performance in perception, attention, and cognition. In a series of experiments on a small number of gamers they report, that gamers of action games were better at perceptual tasks such as pattern discrimination than gamers with less experience with these kind of games. According to Griffiths (2014) 'video games can be fun and stimulating, which means it's easier to maintain a pupil's undivided attention for longer' and that 'because of the excitement, video games may also be a more appealing way of learning than traditional methods for some'. Griffiths (2014) also points out, that thanks to video games there is an opportunity to develop transferable skills, or practice challenging or extraordinary activities such as flight simulators, or simulated operations. Moreover, referring to a study he reveals an

interesting fact about children who played video games following chemotherapy. They need fewer painkillers than others.

Griffiths (2014) states that in addition to their entertainment value, games have great educational potential if they are specifically designed to address a specific problem or teach a specific skill. On the negative consequences of playing he says that these almost always involve excessive game players only, and that there is little evidence of negative health effects on people who are moderate game players.

4 Natural User Interface

Since the first computers, which had a complex interface, and provided only a limited way of interaction in the form of some buttons, flashing lights and making different noises which all had their purpose, the human-computer interaction evolved significantly. The first easier and more comfortable way of human-computer interaction was the *Command Line Interface* (CLI), which enabled users to interact with the computer by typing in commands on a keyboard[4]. This interface is still heavily used though. Mainly on servers and on computers with UNIX-like *Operating Systems* (OS). The interaction with computers however, had to become more intuitive and easy to allow computers to spread out to the masses. The curious nature of the human kind with the help of technological advancement led to the development of the *Graphical User Interface* (GUI)[5]. The GUI with the help of graphical metaphors allowed users the use of complicated applications by exploring what they see on the screen. Britannica states[6]: 'There was no one inventor of the GUI; it evolved with the help of a series of innovators, each improving on a predecessor's work.' The first commercially successful computer with a GUI was the *Macintosh* developed by Apple. It was introduced in 1984. At that time, some critics say it was more suitable for children than for professionals and that the latter would continue to use the old CLI. The GUI of the first *Mac* utilised overlapping windows, rather than tiling the screen, and used a desktop metaphor, in which files looked like pieces of paper, and file directories looked like file folders. The user could delete files and folders by dragging them to a trashcan icon on the screen.

The computer mouse was also introduced with the GUI, which allowed to move a cursor and to point on any item on the screen. Clicking with the mouse on an item in this case performs the command execution. This is the way we interact with

[4] http://en.wikipedia.org/wiki/Command-line_interface (last visited 2014.10.26)
[5] http://en.wikipedia.org/wiki/Graphical_user_interface (last visited 2014.10.26)
[6] http://global.britannica.com/EBchecked/topic/242033/graphical-user-interface-GUI (last visited 2014.10.26)

computers today. However, the development in recent years is directed to a more natural way of using computers, which is called *Natural User Interface* (NUI).

With a *Natural User Interface* often bears in mind the promise of intuitive interactions. We have used the term 'more intuitive' on purpose because when it comes to what it means to be intuitive, the answer is not that straightforward. People who have used a not that well designed motion control interface can tell, that simply to use the motions of the human body to control an interface doesn't necessarily makes it 'intuitive' or 'natural'. As Plemmons & Mandel (2014) point it out in their *Introduction to Motion Control,* when describing an intuitive experience, one often hears that it means that 'one can just sit down and start using it' (Plemmons & Mandel, 2014). They say that 'these interfaces are learnable. They have the right instructions, tutorials, visual affordances, and feedback to help determine how to use it. To be intuitive, an interaction must be learnable.' (Plemmons & Mandel, 2014) One can learn a lot of things though. Even the most complicated things seem learnable to some. Intuitive interaction needs to be more than that. They need to be understandable. 'Understandability is critical to an intuitive experience' (Plemmons & Mandel, 2014). Another thing what we could often hear of an intuitive design is that it's 'mindless'. That means, we don't really think about what we are doing anymore or how we do it. Our actions become a habit over time, part of our routine. Plemmons & Mandel (2014) compare this to when people first learn to drive. It takes a lot of effort to some at the beginning, but it is learnable and understandable, and it becomes a habit. For most daily drivers the act of driving is so intuitive, it almost becomes instinctual. Intuitiveness is subjective, though. What is intuitive to one person may be foreign to another. So the essential part of designing an intuitive interface is first to understand who are we designing for. Beyond intuitiveness, there are a number of other factors that have an influence on the application experience. Like ergonomics and reliable gesture detection can make the experience either to stand out, or burrow it completely.

To sum things up, an intuitive interface is:

- learnable
- understandable
- habitual

In this thesis we will try to conform to these and we will try to create an intuitive natural user interface for our motion-controlled application.

4.1 Multi-touch Interface

Image 1 - Multi touch

The first real application of NUI is represented by the touch interface. This interface evolved to a modern multi-touch interface (illustrated on Image 1) used in today's smartphones and tablets. The first device, which introduced a multi-touch interface, was the *iPhone* by *Apple* released in 2007. 'The device's most revolutionary element was its touch-sensitive multi-sensor interface. The touch screen allowed users to manipulate all programs and telephone functions with their fingertips rather than a stylus or physical keys. This interface — perfected, if not invented, by Apple — recreated a tactile, physical experience; for example, the user could shrink photos with a pinching motion or flip through music albums using a flicking motion.' (Britannica[7]) These predefined motions are called gestures. Devices equipped with multi-touch screens usually support a couple of gestures allowing the user to intuitively do a panning, zoom, rotate, drag objects or flip through pages of documents by flicking. Such gestures are based on natural finger motions giving an increased natural feel to the final interaction. Thus the multi-touch interface represents a more intuitive way of human-computer interaction than using the mouse.

[7] http://www.britannica.com/EBchecked/topic/1326453/iPhone (last visited 2014.10.22)

4.2 Touch-less Interface

Image 2 - Minority Report

With the invention of sensors allowing real-time depth sensing a new kind of interface design (illustrated on Image 2) — only seen in the sci-fi movies so far, like *Minority Report*[8] from 2002 directed by Steven Spielberg — became realisable. This additional depth information allowed computer vision researchers to detect what's going on in front of the computer in *three-dimensional space* (3D). This advantage made the creation of algorithms such as *Skeletal Tracking*, *Face Detection* or *Hand Tracking* easier, enabling computers to understand body movements and leading to the creation of a new way of human-computer interaction called *Touch-less Interface*. The *Skeletal Tracking* is able to track body motion enabling the body language recognition. The *Face Tracking* extends it by the recognition and identification of facial mimics. The *Hand Tracking* enables finger tracking and hand gesture recognition.

[8] http://www.imdb.com/title/tt0181689/ (last visited 2014.10.20)

5 Innovative Input Devices

For the purposes of this thesis, we have considered the use of a couple of devices. The goal was to create an e-learning application, which gets its input from a new, innovative input device. The device should provide an interesting environment, unlike the usual keyboard-mouse or joystick/gamepad setup.

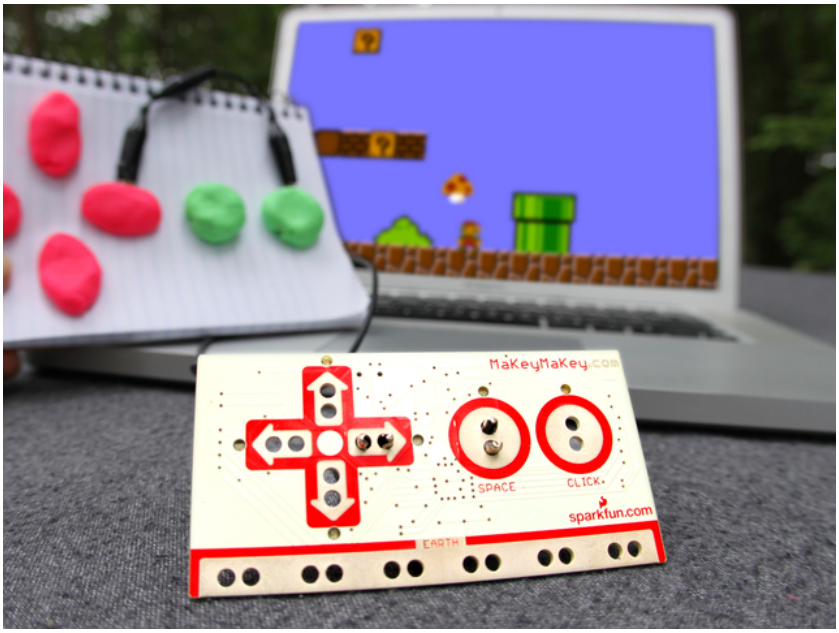

Image 3 - Makey Makey

The first such device on our list was *MaKey MaKey*[9] seen on Image 3. The devices name is a word play from the words 'make' and 'key'. Essentially it's a printed circuit board with a micro controller running Arduino Leonardo firmware. It uses the *Human Interface Device* (HID) protocol to communicate with the computer, and it can send key presses, mouse clicks, and mouse movements. For sensing closed switches on the digital input pins, it's engineers use high resistance switching to make it so that we can close a switch even through materials like the human skin, leaves, and play-doh. That means we can create buttons from play-doh or just even draw a

[9] http://www.makeymakey.com (last visited 2014.9.28)

joystick with a pencil and use our 'do it yourself' controllers to play a game. Or we can load up a piano software and instead of keyboard keys hook up the *MaKey MaKey* to bananas so that they become the piano keys. More in detail, the developers use a pull-up resistor of 22 mega ohms. This technique attracts noise on the input and to save money on hardware, which is a key element to make the device as cost efficient as possible, they use a moving window averager to lowpass the noise in software instead. On the board itself there are six inputs on the front, which can be attached to via alligator clips, soldering to the pads, or any other method. There are another 12 inputs on the back, 6 for keyboard keys, and 6 for mouse motion, which are accessible with jumpers via the female headers, paper clips, or by alligator clips creatively around the headers. By reprogramming the Arduino environment we can use a different set of keys or change the behaviour of the device.

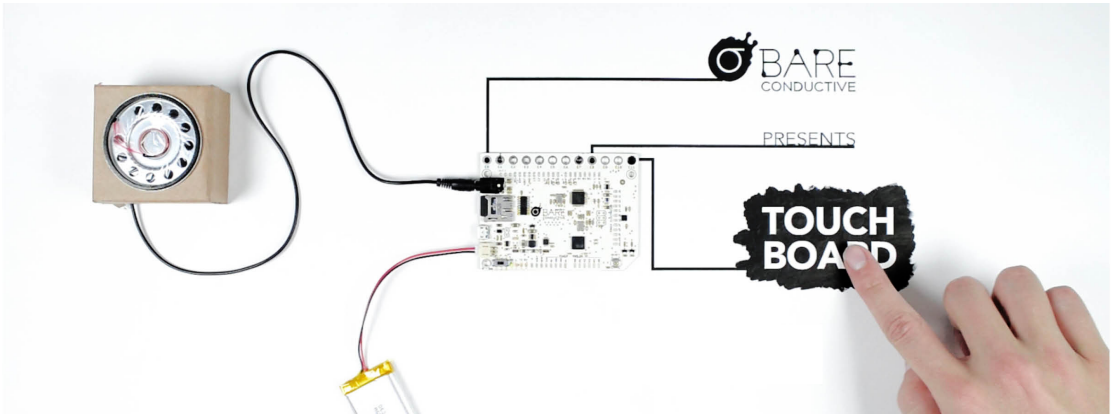

Image 4 - Touch Board from Bare Conductive

Another device on the list was the *Touch Board*[10] from *Bare Conductive* seen on Image 4. It's a similar device to the *MaKey MaKey* capable of turning almost any material or surface into a sensor. It is designed as an easy-to-use platform for a huge range of projects, whether it's painting a light switch on the wall, making a paper piano or creating a custom interactive surface. It plays together with *Bare Conductive's Electric Paint,* which essentially is a non-toxic, air-drying, water-soluble conductive paint. Works great on many materials including paper, plastic,

[10] http://www.bareconductive.com/shop/touch-board/ (last visited 2014.9.28)

textiles and conventional electronics. With the Electric Paint, we can draw our own circuit, our own light switch or keypads and use those as sensors.

5.1 The Leap Motion Controller

Image 5 - Leap Motion

The *Leap Motion Controller* (seen on Image 5) is a consumer-grade sensor developed by Leap Motion, designed to sense natural hand movements. It lets people to use the computer in a whole new way. To point, wave, reach, grab. To pick something up and move it. However, it doesn't replace the keyboard, mouse, stylus, or trackpad. It works with them, and without special adapters. According to the company, with the *Leap Motion*[11] software running, we just need to plug it into the USB port of a computer. But to make use of it, we need to use applications specifically developed for the device. As of September 2013 there were 95 applications available through *Airspace*, the Leap Motion's application store, which has been renamed as Leap Motion App Store[12]. These apps consist of games, educational and scientific apps, as well as apps for music and art.

The *Leap Motion Controller* is capable of sensing almost every little move we make with our hands and fingers, or even with tools (pen etc.) in our hand. More precisely, it's 8 cubic feet (cca. 0,23m³) of interactive, three-dimensional space it can observe. The device tracks all 10 fingers up to 1/100th of a millimetre and it tracks

[11] https://www.leapmotion.com/product (last visited 2014.9.14)
[12] https://apps.leapmotion.com (last visited 2014.9.14)

movements at a rate of over 200 frames per second. The company states that it's dramatically more sensitive than existing motion control technology. It has a super-wide 150° field of view and a Z-axis for depth. The effective range of the Leap Motion Controller extends from approximately 25 to 600 millimetres above the device. This range is limited by how the IR light travels through space. Beyond a certain distance, it becomes much harder to detect the hand's position in 3D. The intensity of the LEDs is limited by the maximum current provided by the USB connection. With the applications written for the controller, we can reach out and grab objects. Move them around in 3D.

'The *Leap Motion Controller* introduces a new gesture and position tracking system with sub-millimetre accuracy. In contrast to standard multi-touch solutions, this above-surface sensor is discussed for use in realistic stereo 3D interaction systems, especially concerning direct selection of stereoscopically displayed objects.' (Weichert, 2013) The device is intended to be used with a minimal setup. It is designed for hand gesture and finger movement detection in interactive software applications. The sensor works by projecting infrared (IR) light upwards from the device and detecting reflections using monochromatic infrared cameras.

5.1.1 Inside The Leap Motion

Image 6 - Leap Motion, exploded

The Leap Motion (exploded illustration seen on Image 6) controller with its *Application Programming Interface* (API) delivers positions of predefined objects like fingertips, hands and tools in Cartesian space. The delivered positions are relative to the Leap Motion controller's centre point, which is located at the position of the sensor's infrared emitter in the middle.

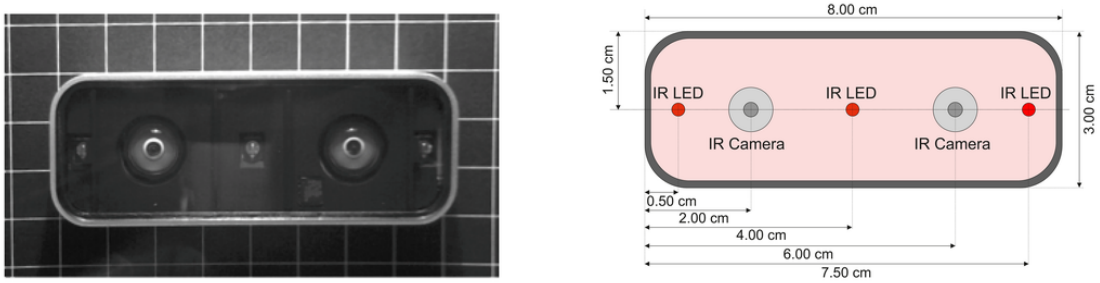

Image 7 - Leap Motion, LEDs and Cameras

As shown on Image 7, the controller has three IR led light sources, or emitters, and two IR CCD cameras. Hence, making it a stereo optic system, so the Leap Motion can be categorised into optical tracking systems based on Stereo Vision. The LEDs emit infrared light with a wavelength of 850 nanometres, which is outside the visible light spectrum. Due to Leap Motion's current patent pending, only insufficient amount of information of its software's geometrical or mathematical frameworks is available. What we know though, is that despite popular misconceptions, the Leap Motion

Controller doesn't generate a depth map – instead it applies advanced algorithms to the raw sensor data.

While the potential of the Leap Motion's technology is great, there are some drawbacks of it. Some reviewers criticised its motion sensitivity and the app control with the device. Particularly the lack of predefined gestures, or set meanings for different motion controls when using the device. This means, there is no standard way of controlling apps. No standardised gestures for given actions. There are different gestures used for the same action, such as item selection, in almost every app. The developers at Leap Motion are aware of some of the issues with the device, and planning solutions which include predefined motions and an improved skeletal tracking function of the hand and fingers. Leap Motion also released a set of Guidelines for menu design, user orientation, user experience and application asset and marketing.

Image 8 - Leap Motion, horizontal palm

Image 9 - Leap Motion, vertical palm

One of the strengths of the Leap Motion Controller is how convenient it is to develop software for the device. It's API delivers pre-processed data so developers don't have to deal with raw data processing to locate hands and fingers. They get precise coordinates in Cartesian space, direction and normal vectors, hands and fingers count and position and more.

Due to the fact that the controller typically lies on a table, or one flat surface in front of the computer's screen and tracks the users hands from below, it works good only so far the palm is in a horizontal position as seen on Image 8. Obviously, if we rotate our palm to vertical position with the fingers above each other as seen on Image 9, the device can't 'see', thus track those fingers from below. They are hiding each

other. The version 2 of Leap Motion SDK tries to fix this, with mixed results yet. Using another Leap Motion controller set up in a vertical direction could eliminate this fact completely, but the use of multiple controllers is not supported yet. Another thing, which could negatively affect the tracking quality, is the light environment of the room the device is used in, and smudges on the surface of the controller. Strong direct light could decrease the precision of the device or even blinding the device's cameras, entirely disabling it's tracking ability. The tracking gets jerky close to the edges of the tracking area.

The current Leap Motion controller is a first generation device and one of the first affordable consumer-grade tracking devices of it's kind. It has its limits, but it gives us a nice preview on what could come next in the NUI field.

5.1.2 Software Development Kit

The *Software Development Kit* (SDK) of the Leap Motion currently officially supports six programming languages, with many more community-created language bindings available. The officially supported languages are: *C# and Unity, C++, Objective-C, Java, Python* and *JavaScript*. The SDK supports all major Operation Systems including Windows 7+, Mac OS X 10.7+ and there is a beta-quality SDK for Linux. The Leap Motion SDK includes the entire library, code and header files required to develop applications and plugins for the tracking device. 'The Leap Motion library is written in C++. Leap Motion also uses SWIG, an open source tool, to generate language bindings for C#, Java, and Python. The SWIG-generated bindings translate calls written in the bound programming language to calls into the base C++ Leap Motion library. Each SWIG binding uses two additional libraries. For JavaScript and web application development, Leap Motion provides a WebSocket server and a client-side JavaScript library.' (http://developer.leapmotion.com) The Leap Motion detects and tracks hands and fingers placed within its field of view (seen on Image 10).

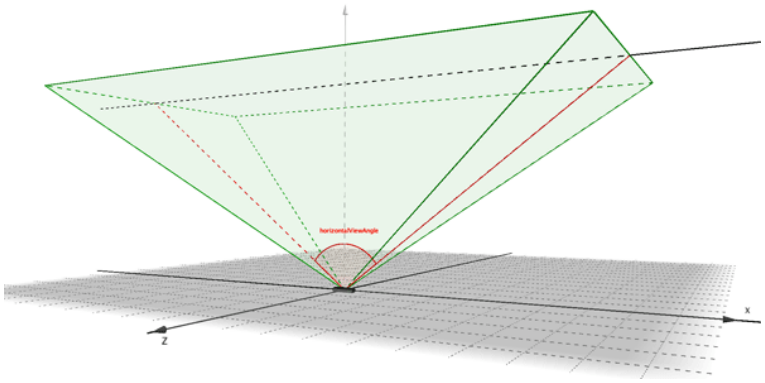

Image 10 - Leap Motion, field of view

It captures this data one *frame* at a time. Applications can use the Leap API to access this data. At the most basic level, the Leap Motion API returns the tracking data in the form of frames. Each of these frame objects carries a list of tracked items such as hands, fingers and tools, as well as objects representing detected gestures and overall motion data. In the version 2 of the SDK there is an Image API that allows access to the raw infrared image data from the sensor. The hierarchy of tracked entities starts with the hand object. The API provides information about the palm position and velocity, direction and normal vectors as well as orthonormal basis of a detected hand, the arm to which the hand is attached, and lists of the fingers and tools associated with the hand. There is an Arm object, which is a bone-like object that provides the length, width, and direction vector and wrist and elbow positions of an arm. When the elbow is not in view, the Leap Motion controller estimates its position based on past observations as well as typical human proportion. With the hand and arm detected, the API provides information about each finger on a hand. Vectors of the position of a fingertip and its velocity, the general direction in which a finger is pointing to and length and width of the fingers are provided by the API. The Bone object provides data about each fingers bones length and width and the joint positions. The finger characteristics are estimated based on recent observations and the anatomical model of the hand, if all or part of a finger is not visible or trackable. Fingers are identified by type name, i.e. *thumb*, *index*, *middle*, *ring*, and *pinky*. Moreover, the API can provide information about a tool held in the hand. A tool is a pencil-like object and it is typically longer, thinner and straighter than a finger. Only thin, cylindrical objects

are tracked as tools. The Tool object provides the same detailed data for the tool as the Finger object for the fingers.

The Leap Motion API measures physical quantities with the following units:

Distance:	millimetres
Time:	microseconds (unless otherwise noted)
Speed:	millimetres/second
Angle:	radians

The Leap Motion Software runs a service on Windows, or as a daemon on Mac OS X and Linux. It connects to the device through the USB bus. Leap-enabled applications access the service to get motion-tracking data. The SDK provides two varieties of API for getting this data: a native interface, and a WebSocket interface. These enable to create applications in several programming languages including JavaScript running in an Internet browser.

The native application interface allows creating foreground, and background Leap-enabled applications. The foreground application receives data from the service as long, as it has the Operating System's focus. When it loses focus, the Leap Motion service stops sending data to it. The Leap-enabled background applications can request permission to receive data even when in the background. Such applications can be used e.g. for mouse control and inputs replacing the touch-panel.

5.1.3 The Skeletal Tracking Model

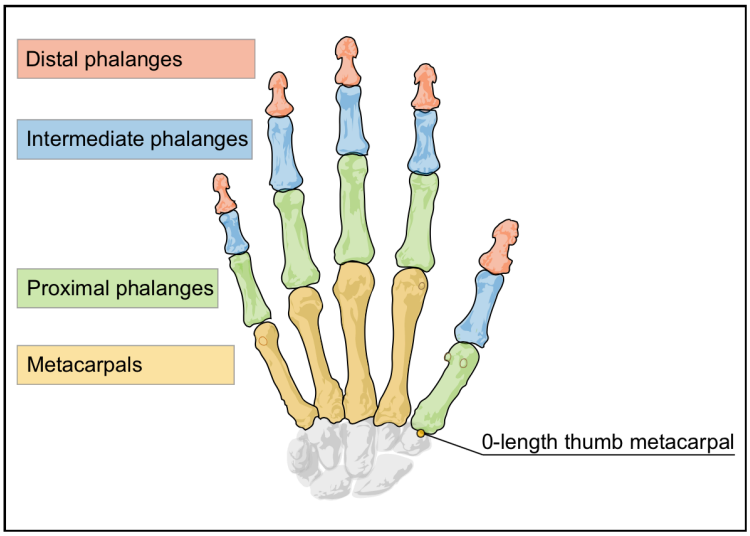

Image 11 - Hand skeleton

The Leap Motion SDK version 2 introduces a new skeletal tracking model that provides additional information about hands and fingers (illustration of a human hand skeleton seen on Image 11) and also improves overall tracking data. The Leap Motion's developer website describes it as following: 'By modelling a human hand, the Leap Motion software can better predict the positions of fingers and hands that are not clearly in view. Five fingers are always present for a hand and hands can often cross over each other and still be tracked. Of course, the controller still needs to be able to see a finger or hand in order to accurately report its position. Keep this in mind when designing the interactions used by your application. Avoid requiring complex hand 'poses' or subtle motions, especially those involving non-extended fingers.' (http://developer.leapmotion.com, last visited 2014.11.28)

'The Leap Motion software uses an internal model of a human hand to provide predictive tracking even when parts of a hand are not visible. The hand model always provides positions for five fingers, although tracking is optimal when the silhouette of a hand and all its fingers are clearly visible. The software uses the visible parts of the hand, its internal model, and past observations to calculate the most likely positions of the parts that are not currently visible. Note that subtle movements of fingers tucked

against the hand or shielded from the Leap Motion sensors are typically not detectable.' (http://developer.leapmotion.com, last visited 2014.11.28)
The skeletal tracking model aims to fix some of the imperfections of the first software version and to provide a more robust tracking solution.

The additions in version 2.0 include

- Reporting of a confidence rating based on the correlation between the internal hand model and the observed data
- Identification of right or left handedness
- Identification of digits
- Reporting of the position and orientation of each finger bone
- Reporting of grip factors indicating whether a user is pinching or grasping
- Reporting of five fingers for each hand
- Reporting whether a finger is extended or not

Perhaps the most significant change for existing applications is the improved persistence of hands and fingers. This should improve the usability of most Leap-enabled applications.

5.1.4 Gestures and Motion

The Leap Motion software recognises certain movement patterns as gestures, which are observed for each finger or tool individually. The software reports each recognised gesture frame by frame like the finger tracking data. Gestures aren't recognised by default though. Developers have to enable gesture recognition in their software for each gesture they intend to use.

The following gestures are recognised by the current software version:

- Circle

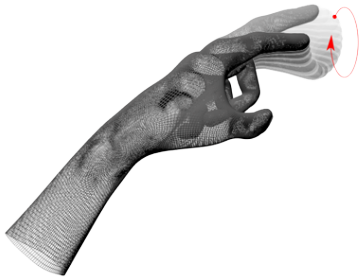

Image 12 - Circle gesture

- Swipe

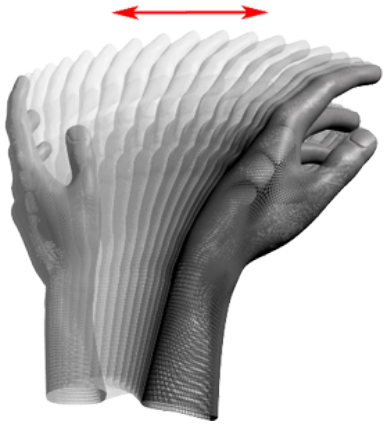

Image 13 - Swipe gesture

- Key Tap

Image 14 - Key tap gesture

- Screen Tap

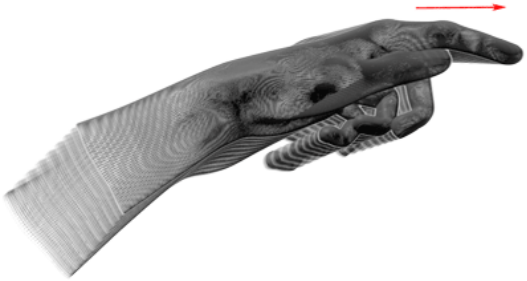

Image 15 - Screen tap gesture

The Leap Motion software is capable of recognising certain types of motion. ‘Motions are estimates of the basic types of movements inherent in the change of a user’s hands over a period of time.’ (http://developer.leapmotion.com, last visited 2014.11.28) Motions include *scale*, *rotation*, and *translation.* The motion type recognition is described in Table 1.

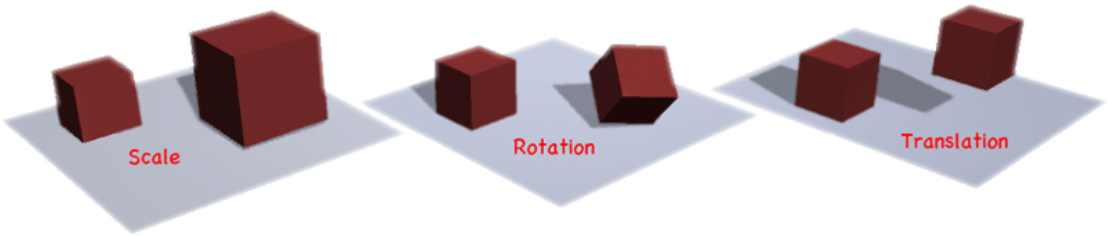

Image 16 - Scale, rotation, translation

They are computed between two frames. Developers can use the reported motion data to design interactions between applications. Instead of tracking the change in position of individual fingers across several frames, they can use the scale factor computed between two frames to let the user to change the size of an object.

Table 1 - Leap Motion - motion type recognition

Motion Type	Frame	Hand
Scale	Frame scaling reflects the motion of scene objects toward or away from each other. For example, one hand moves closer to the other.	Hand scaling reflects the change in finger spread.
Rotation	Frame rotation reflects differential movement of objects within the scene. For example, one hand up and the other down.	Hand rotation reflects change in the orientation of a single hand.
Translation	Frame translation reflects the average change in position of all objects in the scene. For example, both hands move to the left, up, or forward.	Hand translation reflects the change in position of that hand.

5.1.5 Precision of the Leap Motion Controller

Since our hand's and finger's natural movements are really fine, it is important for such a device like the Leap Motion controller to be capable to track and detect these fine movements. The Leap Motion controller was designed with precision in mind. According to the manufacturer, the controller is able to recognise movements with an accuracy of up to *0,01 mm*.

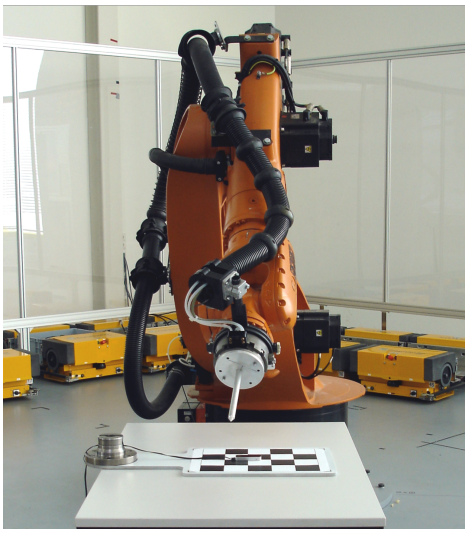

Image 17 - Precision testing

A research group presented the first study of the Leap Motion controller done on an early developer device in 2013. Their main focus of attention was on the evaluation of the accuracy and repeatability. Considering the fact, that the accuracy attainable by the human hand is on average around *0,4 mm*, they drafted an experimental setup

accordingly. This consisted of an industrial robot with a reference pen as seen on Image 17, which allowed a position accuracy of *0,2 mm*. While analysing the accuracy of a static setup, there was no observable influence of the radius of the reference pen upon the accuracy and the deviation between a desired 3D position and the measured positions was less than *0,2 mm*. The evaluation of dynamic situations, where the tip of the robot was moved to different coordinate positions showed, that under real conditions it was not possible to achieve the theoretical accuracy of *0,01 mm*. But the obtained overall average accuracy of *0,7 mm* means, that the Leap Motion is more accurate than comparable controllers on the market today.

5.1.6 Leap Motion Showcase

Since it's introduction in 2012, the Leap Motion controller has been used in various fields for many purposes and enabled developers to create engaging interfaces for different environments. Controlling a robotic arm with your hand movements, creating a 'touch-less touch-screen' or even giving deaf people a voice by translating sign language real-time. A particularly interesting field is the use of the controller in schools. As Chatzopoulos (2013) points out in his article 'How to use Leap Motion in the classroom', according to his trials the Leap Motion can make an excellent teaching companion. With the use of proper software, the device can elevate almost any lesson to a highly enjoyable experience. Furthermore, it appears that the Leap Motion controller has a great potential in the field of special education especially with autistic and developmentally delayed students and it can help students in the lower grades of elementary schools to develop eye-hand coordination skills. The device gives a new meaning to one of the most popular trends in education, gamification. With the controller in use, learning literally becomes a game. It's kinaesthetic approach turns every activity into an enjoyable, interactive learning experience. Whilst according to some, the use of such devices is distracting, it can be observed that with the Leap Motion i.e. the students' interest in the lesson becomes more intense and it keeps them engaged and motivated for longer. Moreover, students are

more likely to retain what they learn, and they easier apply the new information on everyday situations. (Chatzopoulos, 2013)

The list of possible use cases doesn't stop here. Veterinary surgeon *Charles Kuntz* (Kuntz, 2013) used the device in August 2013 to assess a CT scan with the medical imaging program called Osirix while scrubbed into surgery, all without the need of removing his sterile gloves. This is believed to be the first reported use of the controller in this application.

Image 18 - DexType keyboard for Leap Motion

With a virtual keyboard like *DexType*[13] we can fully take advantage of the touch-less natural interface created by the Leap Motion. We can type with ten or just two fingers (one on each hand) literally in the air (as seen on Image 18) on a virtual keyboard, the keys are aligned in a line at the bottom of the screen. Being touch-less it means no touching of a surface is necessary. There is no need to touch anything with i.e. dirty hands in a car service or diagnostic centre or in a sterile environment like an operating room where surgical operations are carried out.

Another tool, which makes the touch-less use of a computer a reality, is the *BetterTouchTool*[14]. This utility enables users to control the mouse by pointing with their finger to the screen. The mouse moves to that position where the users fingers direction vector points to. The tool enables i.e. scrolling on a web page with the hands 'in the air' and using gestures for various features like turning the volume up or down by circling with a finger clock- or counter clockwise. Mac OS X uses multi-touch

[13] http://www.assistivetechnologyblog.com/2013/07/dextype-virtual-keyboard-for-leap.html (last visited 2014.10.14)
[14] https://airspace.leapmotion.com/apps/bettertouchtool/osx (last visited 2014.10.14)

gestures to scroll and zoom, to swipe between pages or full-screen apps, to show the notification centre or the desktop and more. These can be replaced by gestures performed with the fingers and hands in the air. From now on, one only needs a swipe in the air above the sensor to turn the page of an e-book or to go to the next slide in a presentation. The list of possibilities is almost endless.

Image 19 - Oculus Rift DK2 with Leap Motion

One of the pioneers in *Virtual Reality* is the company called *Oculus* with their headset *Oculus Rift* seen on Image 19 with an attached Leap Motion. The *Oculus Rift* is a virtual reality headset which creates a stereoscopic 3D, approximately 100° field of view with excellent depth, scale and parallax. It uses a custom real time tracking technology to provide low latency 360° head tracking allowing users to seamlessly look around in the virtual world just as they would in real life. To achieve a high quality, more natural and comfortable 3D experience it presents unique and parallel images for each eye. This is the same way the eyes perceive images in the real world. A VR headset provides the image only (newly also sound), it's a substitute to the monitor. That means to control the environment or a player in a game, the use of some other input device like mouse, keyboard or gamepad is necessary. The use of the Leap Motion controller in a VR experience is particularly interesting. Leap Motion co-

founder *David Holz* writes in a blog post[15]: 'If virtual reality is to be anything like actual reality, we believe that fast, accurate, and robust hand tracking will be absolutely essential.' The drawback of wearing a VR headset is that the user can't see anything from the real world. There are gloves, which provide hand tracking for VR, but their use is cumbersome. With the help of the *Leap Motion* developers can create a VR experience in which the player can actually see his hands and use them to interact with virtual objects and all this naturally without using some ugly gloves. To help developers to explore this paradigm, Leap Motion released a mount for it's controller which can be used to mount the device on the front face of a VR headset, so that the tracking follows the users wherever they look at.

Image 20 - Parrot AR Drone 2.0

Drones are getting more and more popular amongst hobbyists as well as professional photo and videographers. There are hobby drones like the *AR Drone 2.0* (seen on Image 20) from *Parrot* which provide an API, which can be used by developers to control the drone. In a blogpost[16] published on Leap Motion's website, *Daniel Liebeskind* describes how he created a motion-controlled interface to control the drone with the Leap Motion. He has used Node.js as server infrastructure for directing communication between the drone and the Leap Motion. Node.js is a platform built on Google's open source JavaScript engine called V8, which is written in C++ and is used in Google's browser Google Chrome. It's been designed to allow

[15] http://blog.leapmotion.com/leap-motion-sets-a-course-for-vr/ (last visited 2014.10.15)
[16] http://blog.leapmotion.com/the-beginning-of-a-drone-revolution/ (last visited 2014.10.15)

building of fast, scalable network applications. It uses an event-driven, non-blocking input/output model that makes it lightweight and efficient. To translate hand motion and coordinates into control commands for the drone, *Liebeskind* has used Leap.js, which is the Leap Motion JavaScript SDK. He uses Leap Motion's X, Y and Z axis hand position tracking to control the drone's up/down, forward/backward and right/left actions. To take off, land or rotate the drone, he has used gestures. Gestures can be used to make the drone flip over, or to make it do other tricks as well. *Liebeskind* published his project on GitHub so everyone can try it and contribute.

Image 21 - Painting Heineken bottles with the Leap Motion

The Leap Motion has been used to create a lot of different and interesting applications and experiences, even not really tech focused ones. For the fourth annual *ArtRio Festival* in Rio de Janeiro, Heineken, the Dutch brewing company approached *Eduardo Abdou's* Interactive Lab about building a dynamic experience for the festival. The company wanted something involving art and technology. Abdou's choice fell on the Leap Motion, which he has used before, but this was a good opportunity to develop something challenging with it. The idea was to make people be able to design their own Heineken bottles by waving their hands in the air as seen on Image 21. Abdou's team executed the project in just a month using an *Arduino*, *Unity3D,* and *Leap Motion* to control the motors, relays and solenoids. The Leap Motion blog claims that the installation was a real success. People created more than 1200 Heineken bottles with a unique design. Abdou says that he believes that 'what

made this project engaging was precisely the sensitivity and accuracy of the sensor' and that 'people were impressed that they were able to operate the machine without touching any buttons or physical control.' (Mitchell, 2014)

Image 22 - Cyber Science - Motion

The most common use case of the *Leap Motion* is however, as a substitute of a game controller, for playing games. There are already numerous 3D games available on *Leap Motions'* App Store. There are also some educational apps. We have had a look on some of the best-reviewed ones. Among educational applications, *Cyber Science - Motion* (screenshot on Image 22) is one of the most popular ones. In this 3D application the user can explore, dissect and assemble a human skull. The makers describe their application with the following words: 'Unparalleled interaction allows users to experience the subtle and complex spatial relationships of human anatomy first-hand. Watch as game mechanics and rich content combine to provide the building blocks necessary for vital scientific understanding.'[17] The app provides two play modes. The first one, 'Dissection' allows players to dissect and reassemble a skull at their own pace to explore the skull's construction. The other one is called 'Assembly', in which players can test their skull re-construction skills against the clock in different challenges.

[17] https://apps.leapmotion.com/apps/cyber-science-motion/osx (last visited 2014.11.5)

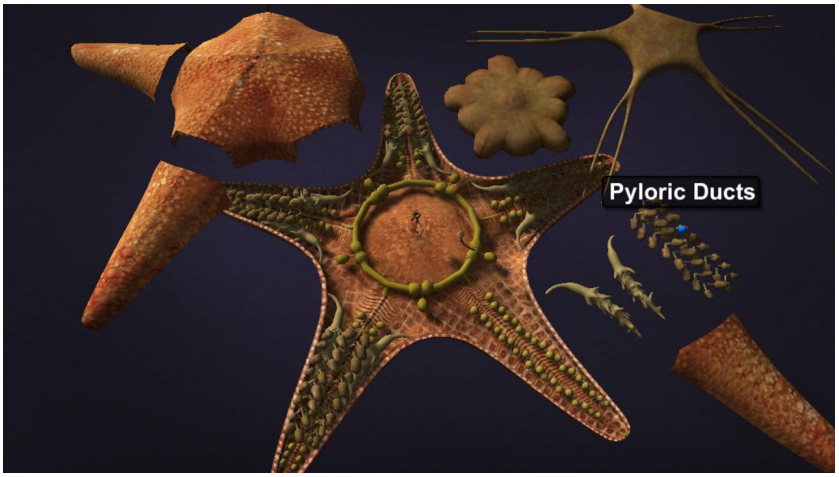

Image 23 - Cyber Science - Motion: Zoology

Another application from the same developers is *Cyber Science - Motion: Zoology*.[18] This app (screenshot on Image 23) provides the same features but this time on 6 realistic zoology models:

- Tarantula
- Rhinoceros Beetle
- Caterpillar
- Butterfly
- Earthworm
- Starfish

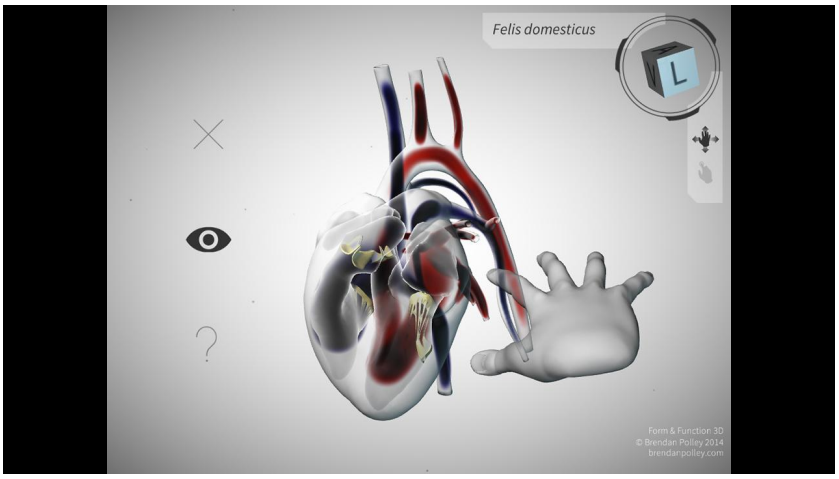

Image 24 - Form and Function 3D

[18] https://apps.leapmotion.com/apps/cyber-science-motion-zoology/osx (last visited 2014.11.5)

A similar application is called *Form and Function 3D* (screenshot on Image 24) described by the developers as: 'an educational tool meant to return a sense of space to learning comparative anatomy.'[19] In this app, players can compare the hearts of three different animals:

- Shark
- Salamander
- Cat

In *Solar Walk - 3D Solar System model*[20] (screenshot on Image 25) players can navigate through space and time to explore our Solar System, learn about the planets, observe them in close-up, learn their trajectories, structure, history of exploration and more.

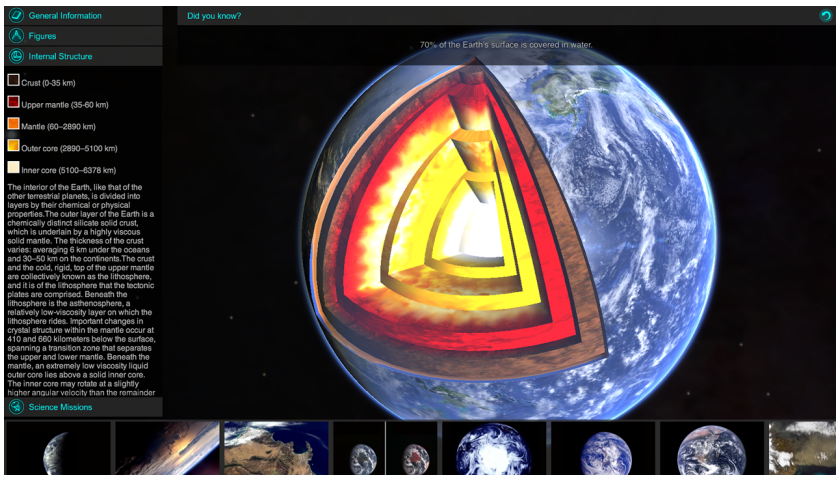

Image 25 - Solar Walk - 3D Solar System model

[19] https://apps.leapmotion.com/apps/form-and-function-3d/osx (last visited 2014.11.5)
[20] https://apps.leapmotion.com/apps/solar-walk-3d-solar-system-model/osx (last visited 2014.11.5)

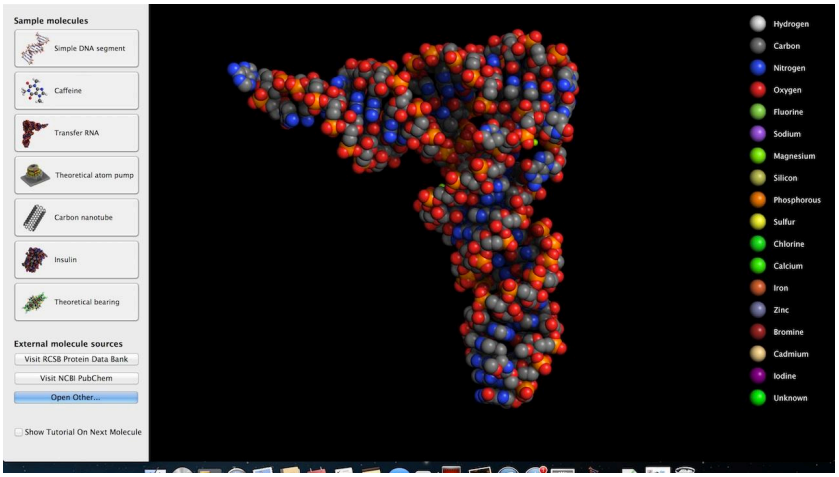

Image 26 - Molecules

Molecules[21] (screenshot on Image 26) is another educational app which focuses on molecules and their 3D visualisation. It is essentially a file reader, which renders the 3D representation of molecules stored in various file formats and allows its users to manipulate them with the help of *Leap Motion*. The *Leap Motion* enables a faster and better manipulation with the three dimensional molecules than a standard 2D input of a mouse.

Image 27 - Sortee

[21] https://apps.leapmotion.com/apps/molecules/osx (last visited 2014.11.5)

Another quite popular educational app is *Sortee*[22]. Sortee (screenshot on Image 27) is a puzzle game, which tests the users' perception of everyday objects. Again, best described with the developers' words: 'All kinds of junk clutters up our lives. In Sortee, you sort these ordinary items into categories. Sounds easy, right? Well, what colour is an apple: red or green? How many legs does a seahorse have? Is an igloo cold, or round, or somebody's home -- or all three? When you start to think about it, even the simplest objects are defined by more than meets the eye.' As objects appear on the screen, the player has to flick the object with a swipe of the finger into the right bin, and they only have a few seconds to sort it out which bin is the right one. The makers of the game say, that only players with the sharpest brains and quickest fingers can succeed.

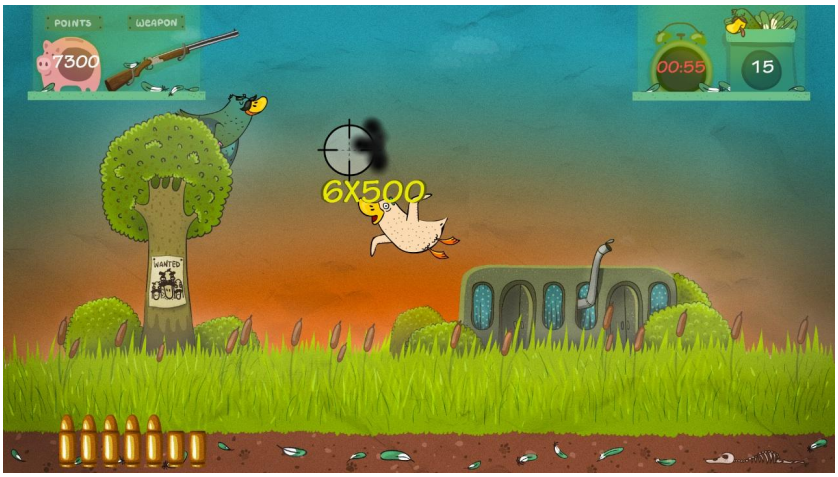

Image 28 - Duck-n-Kill

Apart from educational games and other kinds of applications and utilities, the *Leap Motion* is also about casual games. Among all the entertainment games there is one suitable for older kids only (16+), which nicely integrates the precision of the controller as well as its gesture recognition capability. *Duck-n-Kill*[23] (screenshot on Image 28) is a shooter game in which players shoot flying ducks with their fingers as the weapon. Combine aiming with the finger, shooting with a short fast swipe up,

[22] https://apps.leapmotion.com/apps/sortee/osx (last visited 2014.11.5)
[23] https://apps.leapmotion.com/apps/duck-n-kill/osx (last visited 2014.11.5)

reloading with a circle gesture and a classic duck hunting game steps into a higher level of modern entertaining.

This closes our list of interesting, innovative use cases and games with the *Leap Motion* controller. In this chapter, we wanted to showcase some of the interesting ideas and projects people have created with the controller to get a feeling of what the device in smart hands is capable of. There are lots of possible uses of the controller and with some creativity, one can create really engaging experiences.

6 Implementation

As written in the introduction, the goal of this thesis is to make use of a modern, innovative controller and to create an e-learning application suitable for children in the elementary school. This application will be a game in our case, which should be easy to use, but the gameplay itself should be challenging and informative so that it has a didactic effect. That means, children should be able to learn by playing the game. Furthermore, the game should look nice and it should be also fun to play so that kids should not get bored easily.

When designing for a motion controlled experience one of the first things one has to think about is in what environment will the application be used. People are quite adept at using their hands but the movements and gestures are sometimes subtle, sometimes not that precise and we cannot hold our hands perfectly still. And if it comes to holding the arms for extended periods of time, we do not really like that. Depending on the application we have to consider different ergonomic factors. In our case the environment will be a classroom and the kids will use the application while standing or sitting at a desk. The interactions will be designed with rest periods so that the hands and arms don't get tired that fast.

We have decided to create a game which helps children to practice addition and subtraction between 1 and 100, and which makes them more interesting to get to know the small multiplication chart and its practice. All this split into more levels with variable difficulty.

The game is placed into a 3D cartoon-like virtual world. In this world there are colourful balloons floating in the air with numbers on them. The player gets a mathematical exercise to solve. By solving we mean, the player solves the exercise and he has to destroy the balloon, which carries the result of the exercise. So that the game is more challenging, the player has to solve the exercises under time pressure. Sometimes the player even has to navigate through the world and search for the right balloon. These are spread around and and they might be hidden behind trees or bushes. If the player pops the right balloon, he/she earns points and can continue to the next level. If he/she does not manage to solve the exercise in time or tries to

destroy a balloon, which does not carry the result to the exercise, he/she loses one of the lives. At the beginning the player owns three hearts - lives. If the player manages to go through all the levels, or loses all the lives the game is ended and the score is presented.

The power of *Leap Motion* is unleashed when used in a 3D environment. To create our 3D world we have used *Unity,* a 3D game development ecosystem which is a powerful rendering engine fully integrated with a complete set of intuitive tools and rapid workflows to create interactive 3D and 2D content.

The following chapters will describe the tools and techniques we have used to create our game prototype.

6.1 Unity3D

Image 29 - Unity

Unity is a full-featured 3D and 2D game development environment currently available for Windows and Mac OS X. It allows game designers to develop games in a highly visual approach. What makes it even more interesting, particularly for game developers-beginners, is it's *Asset Store* with thousands of ready to use assets and a huge knowledge-sharing community. Unity features an advanced game editor with a powerful toolset in which developers can create their games. It allows importing all

the assets and arranging them, this is where scenes with terrain, lights, audio, characters, physics and more are built. Interaction can be added through scripts right in the Editor (Image 29). We can attach scripts and properties to game objects just by clicking on the game object and assigning the desired property. The effects of these properties can be tested and edited simultaneously by play testing and editing the game. This encourages the designer to experiment with different game world settings, making the development process fun and easier. There are different windows, or views in the Editor, which contain the tools and workflows. These views are driven by Unity's own GUI scripting and thus they are fully customisable and can be extended with new functionality. One can benefit of Unity's features when prototyping a concept of an application. The relatively easy to use Editor makes it a smooth and fast process. We can even hear from some developers that Unity's is only good for prototyping, because it's that easy to use. They are not taking the software seriously. Although it's a fully capable game engine and editor, some may refer to it as an easy to use tool for beginners. This is only emphasised by the fact that there is a free version of Unity too, although it's a rather cut-down edition of the Pro version. Due to that the free Unity application is so accessible to anybody, there are far more beginners on it than on other platforms, producing lesser quality games or applications, overshadowing the AAA titles made with Unity. While it is not ideal for the software's reputation, we would rather take Unity's user-friendly nature as a plus then a contra feature. Professional tools do not always have to be too complex and complicated. The software is highly capable and tries to offload a lot of hassle from developer's hands providing advanced physics and interactions so it might not be that well optimised for all types of games, but a lot depends on the game itself we want to make.

Unity's asset handling has been created with ease of use in mind. It supports a wide range of industry tools. It can efficiently import models, textures, audio, scripts, sprites and other assets into the project. Moreover, we can simply copy our assets to the projects folder and Unity will automatically import those into the project. Apart from the GUI scripting, Unity provides an extension API for creating plugins and extensions. This API enables us to extend e.g. the import process of all assets, giving us full control over how the assets are being imported.

Unity Editor - which we have described previously - is, as its name suggests, an editor in which we can build up our whole game world. It provides some 3D primitives to work with, but it's not a 3D modelling application. However, Unity can import models, bones and animation from nearly any 3D modelling application like Maya, 3ds Max, Cinema 4D, Cheetah3D, Blender and more. If it comes to text and fonts, Unity can handle the rendering of Unicode TrueType Fonts perfectly. Speaking of rendering, Unity supports the Windows DirectX 11 graphics API that enables us to work with more complex shaders and adding more detail to the 3D world's models and environment. Since Unity is a multi-platform solution, it supports OpenGL too. It comes with one hundred built-in shaders ranging from the simplest diffuse or glossy shaders, to more advanced ones like self illuminated and more. We can still write our own shaders though, or even shop in the Asset Store for hundreds of ready to use materials and textures. With the direct access to the Graphics and GL classes in Unity, we can bypass the rendering pipeline and create our own specific effects. The Graphics class is the raw interface to the drawing functions and the GL class is the low-level graphics library similar to OpenGL's immediate mode. Unity also supports OpenGL ES, which enables the use of full shaders on mobile devices, with its own optimiser for GLSL shaders that yields a 2-3x fill-rate increase. Unity provides advanced tools for rendering, lighting, special effects, audio, materials, terrains, physics and artificial intelligence. The tools deliver a reliable performance, smooth frame rates and great player experience across all the supported platforms. Unity's precomputed Occlusion Calling solution, which was developed together with Umbra Software, ensures that only those objects are rendered which the camera can see at the moment. This technology works on mobile devices, on the web and consoles with minimal processing overhead. Unity can identify what is visible and what is not by auto-generating data from the scene's static geometry in a format, which can be accessed during runtime effectively.

The Unity Editor provides tools to create and work with terrain models. We can simply carve, raise and lower sweeping and mountainous terrains. The grass, trees and bushes as well as rocks also make a part of the terrain model. Unity has an integrated Terrain Engine, which includes a tree-authoring tool, an ideal option to use

to create e.g. jungles. We can add branches and leafs with real-time preview in the editor. All these advanced features are packed into Unity Editor.

Image 30 - A virtual world created in Unity

If we create our virtual world for example as seen on Image 30 with a diverse terrain, forests and bushes and majestic rocks it will still lack something. It will feel raw and not real until we add physics. The PR text on Unity's website[24] describes it's physics capabilities as following: 'Unity contains powerful 3D physics engine NVIDIA® PhysX® Physics. Create immersive and visceral scenes with clothes and hair that blow in the wind; tires that screech and burn; walls that crumble; glass that shatters, and weapons that inflict major damage!' When designing characters and their movements, Unity features a ragdoll wizard that lets us to easily implement a full ragdoll from an animated character.

[24] http://unity3d.com/unity/quality/physics (last visited 2014.11.7)

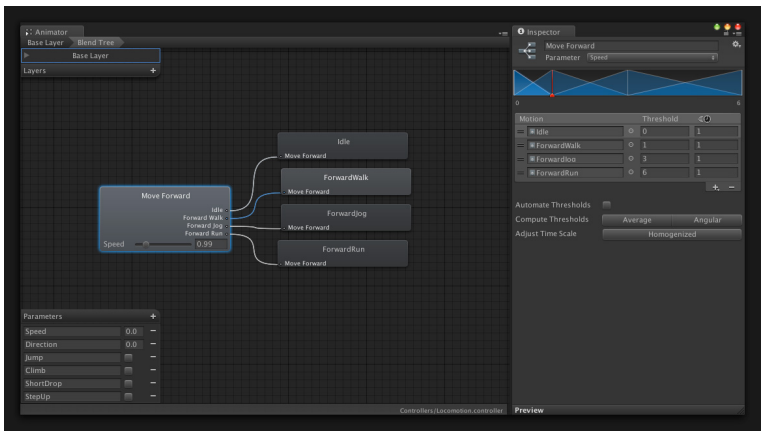

Image 31 - Mecanim

The powerful and flexible animation system called Mecanim seen on Image 31, helps to bring human and other characters to life with smooth, natural and fluid motion. Mecanim is fully integrated in Unity's engine. It enables to animate the light intensity, to blend shapes, or animate sprites. From inside the animation playback, we can call scripts with the help of AnimationEvents, which makes the animation engine even more powerful and flexible. To make a gorgeous game with characters, with todays processing power of computers and even mobile devices, to just texture the character models is not enough anymore. To add more detail and realness we need clothes. Unity has two types of cloth materials built in which are fully physically simulated and they fully interact with the rest of the environment.

Image 32 - Unity, racing game

When working on a racing game as the one on Image 32, we need physics and forces to apply to the car. The car's model has a mass, a rigidbody, which can receive forces and torque to make the object move realistically without scripting anything. There might be different forces applying to the car when accelerating, braking and drifting or colliding with something. To simulate the traction model of a real car tire for example, Unity has a dedicated Wheel Collider. It has built-in collision detection as well as wheel physics and slip-based tire friction model. It can be used for other

objects too, but this collider was created specifically with tires in mind. There are more types of colliders in Unity and they make an essential part of every project. We will give more detail on colliders later.

No strategy or RPG works without some level of artificial intelligence. Unity features a high-performance path-finding and crowd simulation. With automatic navigation mesh generation we can quickly bring the scene to life. Navigation meshes are used at runtime for path finding and they describe the boundaries of any navigable space in the game. With the support of navigation mesh obstacles, our agents are able to react to changing, dynamic environments.

That Unity is multi-platform has already been mentioned. We have not meant the fact that the Editor is available for more platforms, but that it is possible to build a 2D or 3D game, a 3D experience, and have it published to all the major global platforms. Developers have complete control to create and deploy an application on almost any screen. Unity allows publishing on mobile devices, desktops, consoles and even the web. Here is a list of supported platforms by Unity the date of this writing (October 2014), to show how long the list really is:

- iOS
- Android
- Windows Phone 8
- BlackBerry 10
- Windows
- Mac
- Linux
- Unity Web Player
- PlayStation 3
- PlayStation 4
- PlayStation Vita
- Xbox One
- Xbox 360
- Wii U

Unity makes sure that our code works seamlessly on all supported platforms. The team works close with hardware manufacturers to make sure that games work with a solid performance on any of the supported platforms, and that they run on old devices as well as on current flagships. With the *Unity Remote* feature, we can test our game live on any target device. It really is a useful feature if we are developing for a mobile device or console. The *Unity Profiler* is a tool, which we can use for optimising our game on the given platform. The *Profiler* pinpoints the parts of the game with the

highest performance impact to help us optimise our code. We can profile the game during development, or when deployed on a target device. The *Profiler* is however, only available in the Pro version of Unity.

Since Unity introduced its iPhone deployment in 2008, it has become the world's favourite mobile game engine providing easy development and high-end graphics without overloading the mobile device's GPUs. There are numerous chart-topping games in the mobile app stores like Apple's *App Store* or Google's *Play Store*, and since Unity introduced its Windows Phone and BlackBerry support there is a huge number of games made with Unity being ported to these platforms. Unity is not only popular on mobile devices or desktops, there are numerous titles made and published for their *Web Player* too. The *Unity Web Player* makes it possible to play games, or other applications made with Unity, in a regular web browser. Since its launch in 2005 the *Unity Web Player* has been installed on hundreds of millions of machines worldwide.[25]

6.1.1 Unity Scripting

The essential part of every game or other application is scripting. Yes, our entire game world can be built in *Unity Editor*. It will look good, and we can hit play and test whether it runs and works as expected. But without scripting nothing will happen, obviously. Scripting is that part of the development where life is given to characters, where interaction and controls are added. The scripts run our application under the hood. In *Unity*, the behaviour of *GameObjects* is controlled by the components attached to them, but to implement our own gameplay features scripts can be written for different events, which control what should happen when e.g. our character enters some area, where a *collider* as a *trigger* was set. Almost everyone has ever played a computer game, we all know that if a checkpoint is reached, our progress is saved at that point, if our car leaves the track, it slows down, if a target is

[25] http://unity3d.com/unity/multiplatform/web (last visited 2014.11.28)

hit something happens again. All this and many more is done by scripting. Such scripts might be lightweight, with only a few lines of code controlling some basic behaviour or respond to user input, or they can contain extended functions with advanced algorithms controlling physics, or some other complicated parts of the game. We can write our scripts in one of the supported languages, and *Unity* will make sure, that they run as they should on all of the supported platforms. Unity as the date of this writing supports three programming languages natively:

C# - an industry standard language similar to C++ or Java

UnityScript - basically JavaScript, it's a language specifically designed for use with Unity

Boo - a .NET language which syntax is similar to Python

To help with scripting, Unity features fully integrated scripting and debugging with the *Integrated Development Environment* (IDE) called MonoDevelop (Mono). Mono is a cross platform open source implementation of Microsoft's .NET framework, and one of the worlds leading programming environment. By default, Unity will open scripts in Mono's editor MonoDevelop, but any other editor can be used, this can be set up in the *External Tools* panel in Unity's preferences. When a script file is created, one can start writing functions for rotating, scaling and moving objects, which usually take just a single line of code, as well as for duplicating, changing or removing properties. Everything can be referenced directly, by name or hierarchy, tags, proximity, or even touch.

Every basic script file in C# in Unity looks something like this in Listing 1:

```
using UnityEngine;
using System.Collections;

public class NewBehaviourScript : MonoBehaviour {

        // Use this for initialization
        void Start () {

        }

        // Update is called once per frame
        void Update () {

        }
}
```

Listing 1 - Unity, basic script

Every script, which is a Component of a GameObject, has to be a subclass of the built-in class called MonoBehaviour to make a connection with the internal workings of Unity. To enable the script component to be attached to a GameObject the name of the class and the file name must be the same. The first two functions defined in the class are important. The *Update* function is called every frame and this is the place to add code for e.g. movement, triggering actions, responding to user input, or anything that needs to be handled over time during gameplay. Before any game action starts, it's often handy to be able to setup our variables, get or find objects or read preferences first. The *Start* function is called before gameplay, before the first *Update* gets called, and it's the ideal place to do any initialisation.

It was not the first time above, that we have mentioned a *GameObject* or a *Component*. According to the Unity Documentation: 'GameObjects are the fundamental objects in Unity that represent characters, props and scenery. They do not accomplish much in themselves but they act as containers for Components, which implement the real functionality. For example, a Light object is created by attaching a Light component to a GameObject.' (http://docs.unity3d.com/, last visited 2014.11.16) In fact every object in our application is a GameObject. But without any touch, these are only like empty containers. With the right properties a character can be made of these empty boxes, or a physics-driven car, an environment or even a firework. Depending on what kind of an object we want or need to create, different combinations of components can be added. One of the most important components is the Transform component. GameObjects always have a Transform component attached, which is unremovable and defines the object's scale, rotation and position in 3D space. The other components that give the object it's functionality can be added from a script or in the Editor. There are also many pre-built GameObjects ready to use like primitive shapes, Cameras, lights and more.

A script gets called, or is activated only after it has been attached to a GameObject. This can be accomplished by dragging the script to the desired GameObject in the Editor. Once the script is attached, it appears between the GameObject's components and it will start working if we run our game or application. Unity however, creates a more sophisticated connection between scripts and GameObjects than only attaching and listing them. After selecting a given

GameObject, editable fields and values of components can be seen in the Inspector. Unity enables us to add such an editable property to a script component by simply creating a public variable in our script. This public variable appears automatically as an editable property of the script in the Inspector. Moreover, Unity will let us to change the value of such a variable while our game is running. This is especially useful while debugging, as we can see the effects of the changes real time.

6.1.2 Unity Showcase

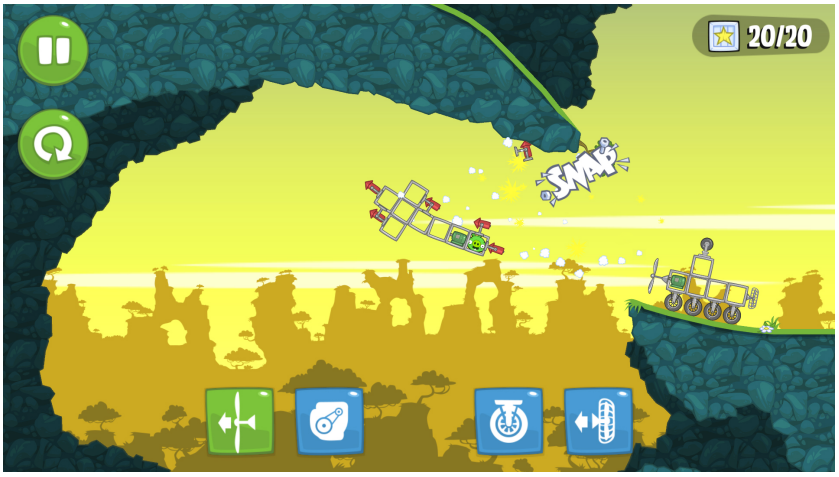

Image 33 - Bad Piggies

With all the capabilities described in the previous section, we can imagine that Unity will not brake us in the development. There are numerous really high quality game titles that have been developed in Unity. One of the world's most successful games for mobile devices, and actually one of the first big hits is Rovio's[26] 'launch a birdie, hit a piggie' game *Angry Birds*. There have already been a few sequels of the original title and they have been ported to numerous platforms. One of the recent sequels comes with a little different point of view. There are no slingshots anymore and the birds were also left out of the game. The game is called *Bad Piggies* and like in Angry Birds, one of the key elements of the game is physics. The player needs to

[26] http://www.rovio.com (last visited 2014.11.28)

set some pigs in motion by building different things like cars or flying vehicles with balloons or rockets as seen on Image 33. The developers at Rovio turned to Unity to prototyping the game and it did fit so well, that they went to production as well. The game was a success, and also thanks to Unity it was made available for iOS, Android, Mac, Windows and BlackBerry.

Another remarkable game, which was developed in Unity, is a mind-boggling puzzle game *Monument Valley*. The description from the makers on the Store describes it as 'a surreal exploration through fantastical architecture and impossible geometry'[27]. In the game we guide our character through mysterious monuments like the one seen on Image 34, uncovering hidden paths and unfolding optical illusions. It plays with our imagination, helps to improve geometric vision, explains perspective and presents optical illusions. The game is touch based, and although we play it flat in 2D, the game is almost entirely 3D. Monument Valley is an *Apple Design Awards*[28] winner game available for iOS and Android. Apple Design Awards is given to developers, who combined design and technology in creative, compelling, and powerful ways.

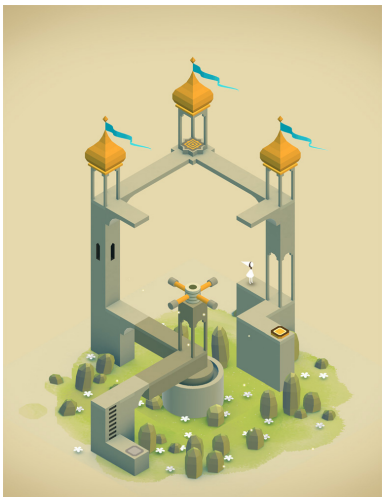

Image 34 - Monument Valley

Puzzles are great for improving our mental skills. They can help to head-count faster or to improve our memory skills. One of such puzzles is another *Apple Design Awards* winner game called *Threes*[29] seen on Image 35. The idea of the game is to collect points by moving the cardboard with our finger and merging pairs of cards. We start with a '1' and '2' by merging them we get a '3' and from now

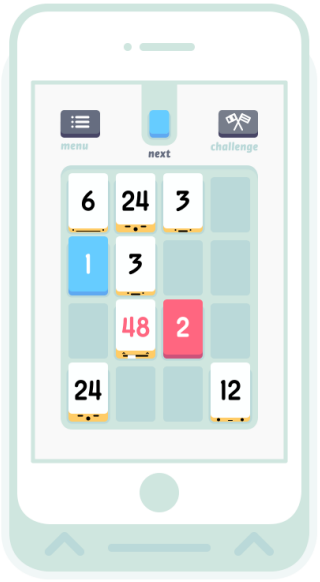

Image 35 - Threes

[27] https://itunes.apple.com/us/app/monument-valley/id728293409 (last visited 2014.11.13)
[28] https://developer.apple.com/design/awards/ (last visited 2014.11.13)
[29] http://asherv.com/threes/ (last visited 2014.11.13)

on we only can merge with the number's matching double. The makers say that *Threes* is a tiny puzzle, which will 'grow your mind beyond imagination'. In other words, playing it might be good for our brain. This has not been scientifically tested yet, though. *Threes* has also been developed with Unity and is also available on multiple platforms.

These games show the power of Unity and what can be done with it. Not only in 3D, but in 2D too.

6.2 Leap Motion and Unity

Unity for any of the supported platforms is available to download on their website[30]. There is a 30-day trial of the Pro version available, and you can use Unity's standard free version after the trial period. The Leap Motion SDK is available on Leap Motion's developer website[31]. There is a download for each of the supported platforms, and the package includes all the library, code and header files required to develop Leap-enabled applications and plug-ins. The *C#* class definitions to use with Unity 3D are provided as a .NET 3.5 library. The library is called *LeapCSharp.NET3.5.dll* the date of this writing. The library loads *libCSharp.dylib* on Mac, *LeapCSharp.dll* on Windows, or *libLeapCSharp.so on* Linux. These intermediate libraries then load *libLeap.dylib*, *Leap.dll*, or *libLeap.so* depending on platform. In the free version of Unity, we will need to copy these three files into our projects folder.

The application for this thesis has been developed on a Mac with Unity for Mac and the Leap Motion SDK for Mac. Therefore, all the following writing will describe the project setup along with a small tutorial for the Mac running OS X. However, the process on other platforms is very similar and the source-code of the application itself written in *C#* is the same on each platform. The Leap Motion SDK

[30] http://unity3d.com/unity/download (last visited 2014.11.13)
[31] https://developer.leapmotion.com (last visited 2014.11.13)

provides a Unity plugin to use with the Pro version of Unity, but we will explain the steps which should be taken to get the SDK work with the free version of Unity.

As Unity supports *C#* and the Leap Motion SDK for Unity works with *C#* only, we choose this language for our application. Although it's mainly a Windows specific language, it was a good opportunity to learn it.

6.2.1 How to Get Started

In this small tutorial we will explain the steps needed to get the Leap Motion SDK work with Unity's free version. To get started, run Unity and create a new clean project.

1. Navigate to the project's root folder in Finder
2. Open the *LeapSDK* folder in the package downloaded from the Leap Motion website, and navigate to the *lib* folder, we need to copy two files from here to our project's root folder
3. Copy the following two libraries to the project's root folder:
    - *libLeapCSharp.dylib*
    - *libLeap.dylib*
4. Open the project's *Assets* folder and copy the following file to the root of the *Assets* folder:
    - *LeapCSharp.NET.3.5.dll*

Now, there should be all set to use the Leap Motion SDK in the application. To try this out, go to the project in Unity. We are going to add an empty GameObject with a script component, and within this script we will use the Leap SDK to write out the number of hands visible by the sensor.

1. With the empty project open in Unity, add an empty GameObject to the scene. Click on *GameObject* in the menubar, and click *Create Empty.* Unity will add an empty GameObject into the middle of the scene and it will be preselected.

2. Now click to the *Add Component* button
3. In the drop-down go to *New Script*, name the script *Sample* and choose CSharp as language
4. Double click the script's name in the Inspector. It will open in MonoDevelop.
5. Copy and Paste the following code from Listing 2, the explanation is below

```
using UnityEngine;
using System.Collections;
using Leap;

public class Sample : MonoBehaviour {
	Controller controller;

	void Start ()
	{
		controller = new Controller();
		Debug.Log("I am alive!");
	}

	void Update ()
	{
		Frame frame = controller.Frame();

		string handsCountStr = string.Format("Number if hands I see right now: {0}",
frame.Hands.Count);

		Debug.Log(handsCountStr);
	}
}
```

Listing 2 - Unity with Leap Motion, basic script

To use the Leap Motion API, we need to use the *Leap* namespace. As we have said before, the *Start* function is a good place for initialization so we create our controller object here, and we write a line to the debug log that will make us sure, that our script is up and running. The *Debug.Log* is Unity's equivalent to the default *Console.WriteLine* call, if we want to log something in Unity we always use this method. The *Controller* class is the main interface to the Leap Motion and to get frames of tracking data and configuration information this object is accessed. In the *Update* function, which is called once every frame, we poll the controller for it's frame, which holds the tracking data for the current frame. The Frame object stores a history of the last 60 frames. These can be accessed by calling the frame object with a

positive integer (1-60). To get the count of hands visible to the controller in the current frame the frame's tracking data are used. We create a string called handsCountStr, which informs about this in the log. Now if we save our script, and go back to Unity from MonoDevelop we can hit play to compile and run our mini test application. If everything went well, the application will run, there will be an empty blue screen and in the Editor's status bar (bottom left) we will see our message *'Number of hands I see right now: 0'*. If hands are placed over the sensor now, the message will change accordingly. There was no visible object added to the scene yet. To see a virtual representation of the player's hand, a *HandController* object can be added to the scene. *Leap Motion* provides this object and the easiest way to add it to our scene is to download and import it from the *Unity Asset Store*. To open the *Asset Store* go to *Window -> Asset Store* or hit *Cmd+9* on a Mac. The *HandController* is in a package called *Leap Motion V2 Skeletal Assets*. Search for the package with the help of the search field, which, the date of this writing is on the top right side of the page. Click download and import the assets after the download has finished. A new folder called *LeapMotion* will appear in the project's *Assets* folder. Navigate to the folder in the project view and find the *Prefabs* folder. You will find the *HandController* in that folder. Drag and drop the prefab to the scene, place it somewhere in front of the camera and make sure that it 'looks' straight up. If you select the *HandController* object, you will see it has a couple of public properties. You can choose from multiple types of virtual hand representations to assign to the controller. More robot like and human hands are available in the *HandModelsNonHuman* and *HandModelsHuman* folders. The human hands look too realistic and they might be inappropriate for some uses. We suggest the use of robot hands. Now with the added controller to the scene, click play. Placing our hands over the *Leap* a virtual hand should appear on the screen copying the movements of our hand. The virtual hand at this point looks too shady. We have not added any lights to our scene yet. To add some light go to *GameObject -> Create other -> Direction light* and place the light somewhere above the middle of the scene. Set its rotation and intensity as needed. Now if we run the application again and place our hand above the sensor, a virtual hand will appear, which is now nicely lit up.

This tutorial described the steps needed to get the *Leap Motion Controller* to work with the free version of Unity, and showed how to script objects, these steps are essential for every Unity application which uses the *Leap Motion Controller*. The script should be considered as a 'Hello World' kind of basic application. We have shown how to create an empty *GameObject* and how to add a component to it. In our case this component was a script. We have shown how to create a script, how to connect and poll tracking data from the *Leap Motion Controller* and how to log data for debugging purposes. Basically, this is how we started our development process.

6.3 The Game

6.3.1 The Game Menu

When a game is started, usually the first thing what is presented to the user is a splash or load screen followed by the *game menu*. Many times, the game menu means the first interaction users will have with the game. If the user's experience with the menu selection and interaction is frustrating or confusing, he or she might give up on the game even before actually playing it. The game menu is just as important as any other part of the game or application, therefore a good menu design is really necessary, especially when the game is played with some special controller, like in our case the *Leap Motion*. When using controllers to interact with a game or application, it is appropriate to use the controller's capabilities only, avoiding the necessity of using the computer's keyboard or mouse is, in our opinion, recommended. This makes the whole design even more challenging. To avoid the use of a keyboard or mouse, the various parts of the game, like the menu itself, have to be designed with the controller in mind, meaning we have to know what exactly it is capable of. While some best practices from regular computer games can be applied on games created for the *Leap Motion* without any issue, it has to be noted that what

works best in regular computer games might not fit ideally into a *Leap Motion* experience. After reading the Leap Motion *Menu Design Guidelines* we have designed our menu with the so-called 'poke' or touch gesture. This means that the user points with his or her finger to the screen and a filled-in circle appears which indicates the cursor. The user moves the cursor by moving the pointing finger and changing its pointing direction. To select, or click a button the user has to make a poke/touch gesture in the air meaning quickly move the finger towards the screen and then back. While this really works great in theory, in our internal trials it failed to deliver the desired experience. Due to the lack of tactile feedback it is hard to guess by the user when the click happens, if it actually does happen at all. Moreover the gesture itself has an impact on the cursor's position, making the selection even more challenging.

The developers at *Leap Motion* have released the source codes of a couple of menu design prototypes one of which they call *marching menu* seen on Image 36. In this design the menu is a two dimensional menu where the buttons are rectangle shaped and the menu is hierarchical. Hovering the upper level buttons the lower level ones will be displayed. The selectable buttons are indicated by a green rectangle. Swiping this rectangle out of the button performs the selection.

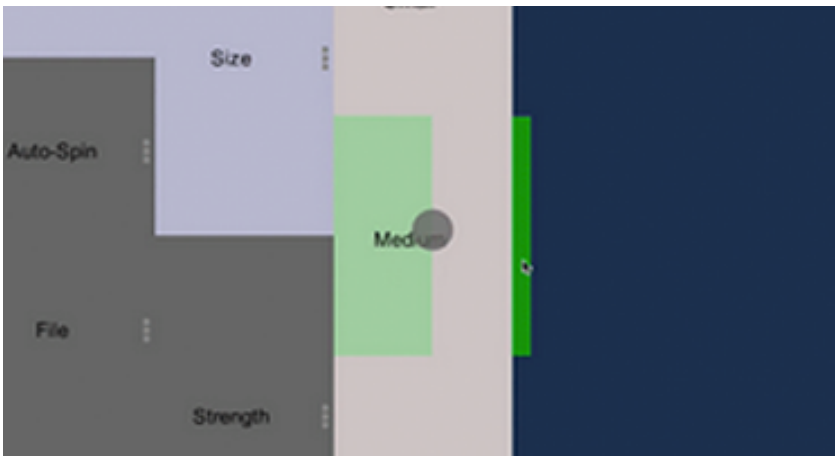

Image 36 - Marching menu

While this design carries some great innovative ideas, its implementation was not that trivial, and we were searching for some more classical menu design that is still

intuitive and similar to those found in regular computer games. After some exploration on this topic on the internet searching for ideas how the menu, which does fit the *Leap Motion* nicely, may be done right, we have found a blog post by *Pierre Semaan* who described the same issues as we have had with the initial game menu design, and he had a really great suggestion for the menu item selection which he named 'hover clicks'. The player still moves the cursor by pointing on the screen above the sensor, but the item selection, hovering over a button for a given time does the clicks. This could be indicated in various ways, from slowly filling out the button with another colour to showing a counter or a hourglass next to the cursor. In our internal trials this technique had a great success and we found that it provided the best experience, the interaction is a bit slowed down due to the waiting times on the clicks though. The game menu of our prototype is full three-dimensional. The buttons are rectangle shaped boxes 'floating' in the clouds. The pointer is a green circle. When hovering over a box, over a button, it slides slightly closer to the camera, indicating that we are hovering over it and the program knows. The click is then triggered if we hover over a given button for one second. Being a three dimensional scene the trickiest part of programming was to detect whether we are hovering over a button.

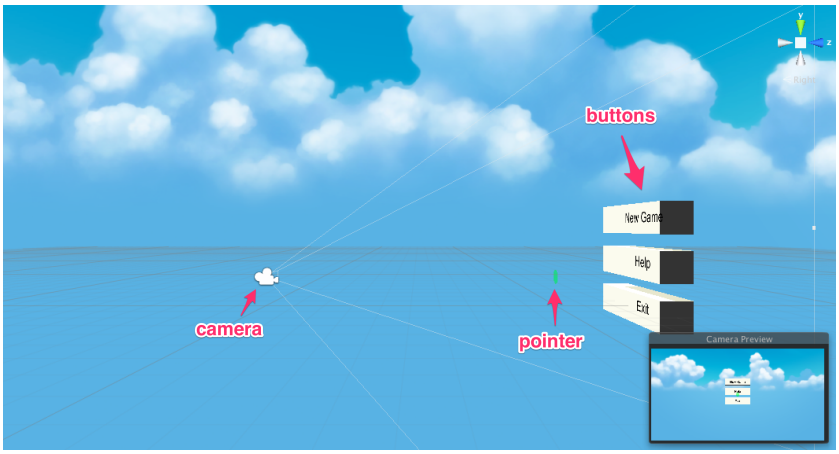

Image 37 - Game menu scene

The scene is set up as seen on Image 37. The camera in this scene is static. We had to detect whether the cursor is hovering over a button as seen from the camera. Thus, if

the cursor looks to ‘cover’ a button from the camera’s perspective. The detection of this was achieved by the help of *Raycasts*.

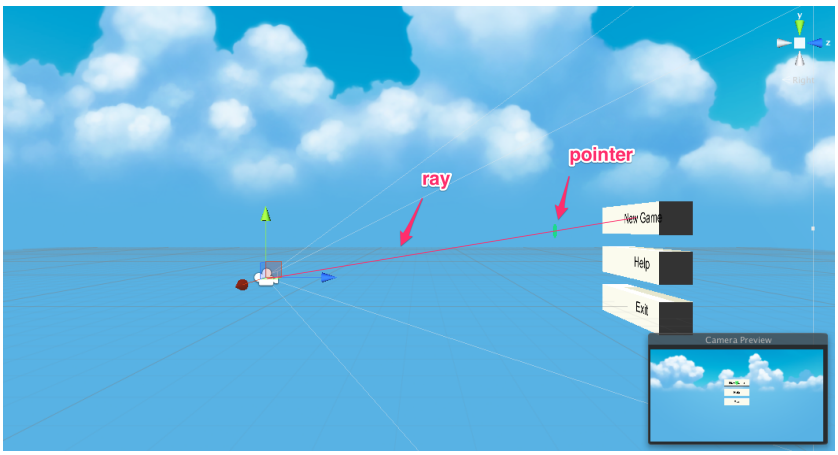

Image 38 - Raycast

We basically send out an invisible ray from the camera in the direction, which goes through the centre of the cursor, and we check if the ray hit something (see Image 38). If the ray hits a button, meaning that the cursor seems to hover over that button as seen from the camera’s perspective, we trigger a counter. If that counter runs out, it triggers the click with the given button. If the cursor moves out of the position being over the button, the counter is reseted. This technique can be seen in more detail in the attached code below in Listing 3:

```
if (pointable.IsValid) {
   finger.SetActive(true);

   finger.transform.localPosition = new Vector3(newVector.x, newVector.y-4,
finger.transform.localPosition.z);

   Vector3 v3 = new Vector3();
   v3 = Camera.main.WorldToScreenPoint(finger.transform.localPosition);

   Ray ray = Camera.main.ScreenPointToRay(v3);
   RaycastHit hit;

   if (Physics.Raycast(ray, out hit, 100)) {
      myGmObj = hit.transform.gameObject;
      menuCounterWithGameObject(myGmObj.name);
      Vector3 v = new Vector3(myGmObj.transform.position.x, myGmObj.transform.position.y, 2f);
      myGmObj.transform.position = v;
   } else {
      menuCounterWithGameObject("");
      menuTimer = 0f;
      Vector3 v = new Vector3(myGmObj.transform.position.x, myGmObj.transform.position.y, 3f);
      myGmObj.transform.position = v;
   }

} else {
   finger.SetActive(false);
}
```


Listing 3 - Unity, raycasting

We have put this code into the *Update* function. It begins with a check if our 'pointable' is valid. The 'pointable' is the frontmost finger of the hand, thus if the sensor tracks a hand and it detects a frontmost finger, we can poll the data from the sensor to move our cursor. If the 'pointable' is valid, we activate the finger object which is our cursor, meaning the cursor becomes visible and we change its position on the screen according to where the vector of the frontmost finger points to. In the next step we create a vector, which stores the converted vector of the cursor from world space into screen space. The *ray* object returns a ray going out from the camera through the cursor, we cast our ray with a length of 100 units and as there are no other objects in our scene except the buttons, we call the function which starts the counter with the button the ray hit. We also slide the button the ray has hit, slightly towards the camera. If the ray does not hit any object we make sure the buttons return to their original position and we reset the counter. If our 'pointable' is invalid, thus the sensor does not track any hands or fingers at the moment, we hide the cursor.

This approach worked really well in our internal trials and we were satisfied with the experience it provided.

6.3.2 The Virtual World

After selecting the *New Game* button, the game itself gets started and another scene is presented with the first level. The game menu has its own scene, its own file in Unity. There is then, only one scene for all the levels, which are dynamically generated inside that scene. The first level is more like a tutorial with only one balloon in it as the result to a single exercise. As already described before, the player has to navigate through the virtual world searching for the balloon, which carries the result to the presented exercise. The game world was created with a terrain object, which is a Unity object, basically a plane, but it allows forming a terrain by adding hills and valleys. The terrain is coloured to look like if it was grass. For our prototype we have used a free Unity asset called *Fantasy Skybox Free* found on the Asset Store. This asset contains a terrain, some cartoon-like trees, bushes, rocks and a skybox. The

skybox is basically the world's sky or the background. In this asset pack it has a cartoon-like cloudy sky look. The world is not infinite. The terrain is however, large enough to serve our prototype well. Our game world looks like seen on Image 39.

Image 39 - Game world

The terrain object, due to its persistence on the screen during the whole game, is the main game object, which has a script component, the main game controller script. This controls almost the whole gameplay.

6.3.3 Navigation

We have tried multiple techniques for navigation in the virtual world. The idea was to use our hands to navigate, to move around in the world. The moving should be as intuitive, as easy as possible. The first technique we have tried was to move around by dragging the terrain with the virtual hand and sliding, moving it around, resulting in the move of the camera above the terrain. This required at least three fingers touching the ground, the terrain to be able to drag it around. This method, while appeared to work well at first, turned out to be confusing and did not work that good in our internal trials. The next technique we have implemented is based on a kind of a virtual joystick. The player closes his/her palm slightly like he/she would grab something, and then moves his/her hand around above the sensor. The camera on the

```
void Translate(Frame frame) {
    Vector3 avgVelocity = Camera.main.transform.position;
    avgVelocity = 50 * handSpeed.ToUnityScaled ();
    avgVelocity = new Vector3 (avgVelocity.x, 0, avgVelocity.z);
    Camera.main.rigidbody.velocity = avgVelocity;

    if (Camera.main.transform.position.x <= -30.0f)
        Camera.main.transform.position = new Vector3(-30.0f, Camera.main.transform.position.y,
Camera.main.transform.position.z);

    if (Camera.main.transform.position.x >= 30.0f)
        Camera.main.transform.position = new Vector3(30.0f, Camera.main.transform.position.y,
Camera.main.transform.position.z);

    if (Camera.main.transform.position.z <= -30.0f)
        Camera.main.transform.position = new Vector3(Camera.main.transform.position.x,
Camera.main.transform.position.y, -30.0f);

    if (Camera.main.transform.position.z >= 30.0f)
        Camera.main.transform.position = new Vector3(Camera.main.transform.position.x,
Camera.main.transform.position.y, 30.0f);
}
```

Listing 4 - Unity, camera translation

screen starts to slide into that direction in which the hand moves. The speed depends on that how far the hand is off the centre of the sensor in either direction. There is a speed limit though ☺. So if the palm is flat, the camera will not move. If the grab strength is bigger than *0.45* units, the 'virtual joystick' gets activated. The code snippet in Listing 4 does the translation of the camera's position in 3D space if the grab strength is bigger than *0.45* units. While our world is, the terrain is not infinite. We have created a virtual border to not let the camera slide too far, so that it stays over the ground. We were satisfied by the experience provided by this type of navigation.

6.3.4 The Balloons

Balloons are an essential element of our prototype. The objective was to be able to pop balloons, which do have numbers on them. Balloons in real life are often sphere or kind of elliptic sphere shaped, and they are usually filled with helium, which

is lighter than air, so the balloons float in the air. They are usually fitted with a line so they do not float away. We have replicated this behaviour in *Unity* with *spheres* as balloons, *springs* and a *line renderer* as an analogy to a line, which holds the balloon, and instead of helium we have applied a *force* to the balloon, which works against the virtual gravity in our game world.

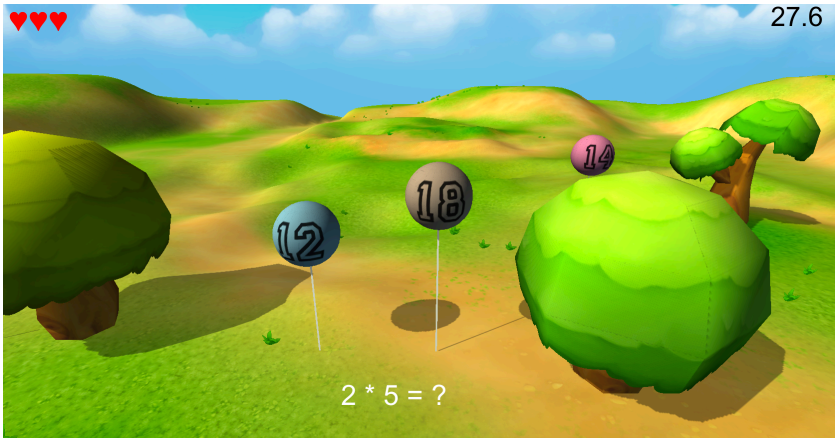

Image 40 - Balloons

For the balloons (seen on Image 40) we have created a *prefab*, a predefined object that consists of a set of different objects. We have tagged this prefab as *balloon* to be able to recognise it. As mentioned above, the balloons in our prototype are basically spheres. The sphere is locked to a given point by an invisible spring, which is a standard Unity component. This spring provides a rubber band like behaviour. The strength of it is customisable, as well as it can be set up to break after it was stretched by a given value. While this spring component is invisible, we have used a *LineRenderer* component, which is drawn on the invisible string's position, to indicate that the balloon is strapped to the ground. The floating effect is achieved by applying a simple physics force with a direction vector pointing upwards. This is done easily by a single line of code as seen in Listing 5, called this time in the *FixedUpdate* function instead of *Update*:

```
void FixedUpdate() {
    rigidbody.AddForce(Vector3.up * 20);
}
```

Listing 5 - Unity, FixedUpdate function

When dealing with a *Rigidbody* which is a component, which makes the object to respond to Unity's physics forces the *FixedUpdate* should be used which is called every fixed frame-rate frame instead of every frame as *Update*. For example, if we have to apply a force to a rigidbody, we have to do this in the *FixedUpdate* function.

Image 41 - Balloon texture

Every balloon, which is created from our prefab, gets its random colour assigned from a range of plastic light colours. It gets its name, which equals to the number on the balloon itself, thus the name of the balloon is the result to a given exercise. This is one of the best ways to identify the balloons. In our first prototype we use the numbers in the range *1-100* for exercises from the small multiplication chart. The numbers on the balloons are pre-rendered images which we have created in Photoshop with its batch image processing feature, so that we did not have to create all the hundred images manually. These images look like the Image 41 and they are mapped to the balloon's sphere to create the desired look.

To indicate a failed attempt to pop the right balloon, the right result for an exercise, the balloon gets coloured in bright red. When the player pops the right balloon it disappears with a visual effect, which looks like a shock wave as seen on the Image 42.

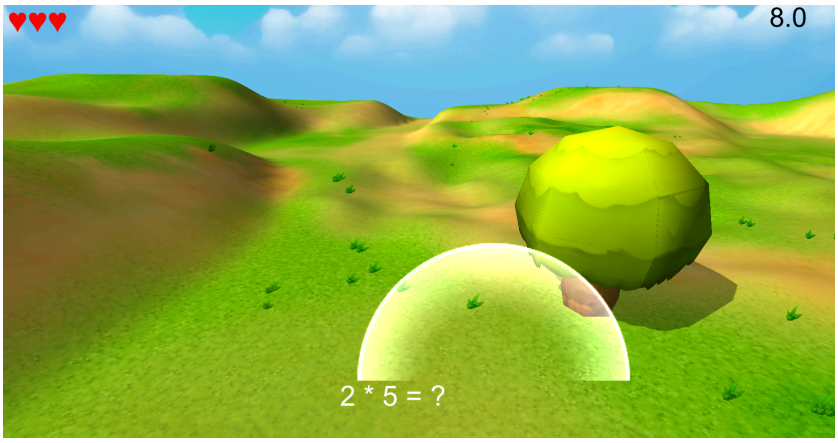

Image 42 - Shockwave

6.3.5 Exercises

The exercises presented to the player had to be selected randomly so that they are interesting. The exercise generation in the same time should not had to be too expensive. In the game logic, each number from *2-10* corresponds to a level. While it does not make too much sense to practice multiplication by one, we have created a tutorial kind of a first level instead. In this level, there is a single exercise so that the player can make him/her-self familiar with the interaction. In each other level, there are five exercises generated randomly. We have chosen such a technique for generating random exercises, that is far less expensive than picking a random number and then checking all the already selected numbers if its there or not. First, we have created an array of all the numbers, which can be used (1-10). Then we used the C# *Random.Range()* function to generate a random number from the range of 1 to the count of the array which contains all usable numbers. We pick the number from the array at the position the randomiser generated and put it to a second array, which will contain the list of numbers, we will use. The number, which was picked, is than removed from the initial array making its count lower. This ensures that the given number will only appear once. This is repeated until the number of exercises is reached. As a result we get an array of randomly picked numbers from a given range, and we can be sure that it contains each number only once. The exercises are then built with these numbers and the second number in the multiplication is the level number. The results are pre-computed and put into a results array. The code snippet in Listing 6 does this computation:

```
ArrayList allNum = new ArrayList();

    shuffledNums.Clear ();
    results.Clear ();

    a = level;

    for (int i = 0; i < 10; i++) {
        int thisID = i+1;
        allNum.Add(thisID);
    }

    for (int i = 0; i < numberOfExercises; i++) {
        int myRand = Random.Range (1, allNum.Count);
        shuffledNums.Add (allNum[myRand]);
        results.Add(a * (int)shuffledNums[shuffledNums.Count-1]);
        allNum.RemoveAt(myRand);
    }
```

Listing 6 - Unity, exercise generation

As we wanted to make the levels more interesting and challenging, we have put more balloons into each level, as the number of exercises. These additional balloons, their numbers had to be generated in a similar way as described above. The range of initial numbers was *1-100* and we have removed the numbers already contained in our results array before generating the others. There are five exercises in each level, and 10 balloons in the lower levels then 15 and 20 in upper levels.

The elements described in this chapter are the key parts of our prototype. We have pointed out the most interesting things and described some parts of the development process so that the reader can understand, and with the provided tutorial also to try out what is it like to develop a learning app for the *Leap Motion*.

6.4 Field Study

Image 43 - Children playing with the Leap Motion

To test out our prototype for game based learning with the Leap Motion in a real world scenario, it was given to 12 third grade children in an elementary school. The kids played with the application one-by-one. Image 43 illustrates two children playing the game. We were curious how the Leap Motion and the application will

perform in a school class environment, as well as what the kids' feedback will be. The prototype was built for them after all.

We have had a few performance issues during the test. The application sometimes behaved glitchy, unplayable and the sensor did not work well. This might have been due to the strong lights in the classroom, or the fact that the application was each time run in unity, and was not exported as a standalone app, or because the SDK is still in beta, thus some performance issues may occur. In the majority of cases however, it did work really well. The sensor had no problem with the tracking of small hands of the kids.

Before the kids have started to play, we gave instructions and showed them how to play. This is due to that our prototype does not include any graphics yet, which would show how to play. All the kids reported, that they have never seen anything like this before. A short introduction was therefore necessary. After they have seen the game in action they were eagerly waiting for their turn.

The kids really liked the interaction with the game menu. Some said they felt like magicians while starting the game. After the game has started we have realised that the choice of the robotic hand was good, as it did not scare the kids. The majority of children quickly got it how to move in the world with the 'virtual joystick' and they were searching for the balloon with the right number on it. The most interesting part of the trial however, was to observe how different kids played the game. Although our test group was small, after the trial we have categorised the kids based on their performance in the game. There were basically four groups. The first group of well performing kids at the class, which were also good in solving exercises in the game as well as having good hand-eye coordination skills and they had no problems with moving around in the virtual world, finding balloons and popping them. They really wanted to play the game again. In another group, there were children, who were still good in solving exercises, but they had issues with coordination and finding the right balloons. Some of them had so much trouble playing the game that their experience was so frustrating they not really wanted to play the game again. For those kids from this group, who while found the game a bit hard to play, but had fun playing it, practicing similar games as we have seen, might have a positive influence on the hand-eye coordination skills. They performed a bit better as they played the game the

second time. Those students, who were somewhat slower in solving exercises, made the third group but they had great coordination skills and had no problems with game interaction. The play for them was fun, and they wanted to play the game over and over again to be able to gain a better score. This game element was the biggest motivation. After each play, the kids were talking to each other and showing off their score. Those gaining higher were satisfied, while those who have earned a lower score wanted to try it again to make it better. This form of virtual reward seems to work really well. It is motivational enough to make kids want to play a math game. In the last group there were the kids who performed not that well. They had issues with solving exercises. For them, solving math exercises and searching for balloons was not that fun. They quickly realised that the sensor can track two or more hands too, and that it is fun to play with the balloons with the virtual hands, or how cool it is when the virtual hands do the same as their own hands, when they clap and so on. They got bored really quickly.

The trial showed that the majority of kids loved the game. We asked them whether they want to play it again, and the response was positive. Almost all of them wanted to play the game again. They asked what is the name of the game, how and where can they play it. The sensor, how small it is and what it is capable of, amazed them. The kids enjoyed playing the game and they quickly forgot about math and all they wanted to do was to find the right balloon, what obviously was possible only after solving the given exercise, and they wanted to perform better to earn a higher score. This means they were practicing math while playing a game, and not thinking about school or exercises. We asked them how the game compares to the games they play day by day on their iPads. The answer was common: the iPad is nothing special anymore, and '*this is much-much better*' or '*the best game I have ever played in my life*'. Yes, kids tend to overrate things and change their minds quickly over time. But they grew up in a World where there were smartphones and tablets everywhere, so they got used to it. They use iPads at the school where they have a couple of different apps for learning. They said that it is not a big deal to use a touch-screen '*there you have to tap and touch it all the time, here you only swipe with your hands in the air and you play the game*'.

After the students played the game, they were given 5 statements, which they had to evaluate with the help of smiles as marks as seen on Image 44. These statements were the following:

- I would like to play the game again.
- It was easy to pop the balloons.
- It was easy to find the balloons.
- I think the exercises were easy.
- I think the game was easy to play.

Image 44 - Smileys as marks

The majority of students gave the best mark for all the statements. While this is good for the first three statements, it may indicate that the exercises were too easy and they might have been more challenging. Those kids however, which had problems with the navigation and the gameplay, gave the lowest mark for all the statements. This clearly shows, that for them it was not really fun to play the game as it was frustrating to not be able to find and pop the balloons.

These findings clearly show, that such games might not be ideal for all types of students, particularly students with a need of special education or those, whose hand-eye coordination skill is not that well may have problems playing such games. The majority of children however tend to get used to things quickly. They can successfully use the widespread technology in their learning process, and our prototype has shown that bringing in fresh air, something new to them can boost their motivation.

7 Conclusion

The goal of this thesis was to make use of a new controller device and to develop a prototype of a learning application for it, which would mean a possible way to improve the game based learning process at elementary schools. We made a research on the topics ‘Game Based Learning’ and ‘Gamification’, we showed what is current state of the literature, what researchers say on this topic, what they criticise or what they think the positives are. We presented the most important information about the *Leap Motion* in detail, described how it works, how precise its tracking is and what can be done with it with some tinkering. In the Implementation chapter, we presented a tutorial how to begin with the development of an application for the Leap Motion in Unity under OS X. There is a subchapter which showcases the state of the art of the applications made with Unity. Unity itself, as well as its features is also described. Finally, we presented how the prototype was built and what were the problems we had during the development process and how we solved them.

In the development process we got familiar with the Leap Motion SDK and Unity, as well as with the C# programming language. We have only had little experience with C# before, this project was a good opportunity to learn C# and to learn making games in Unity. Unity turned out to be a really capable system to build 3D games fast and easily and its learning process was also straightforward.

In our practical tests we went to an elementary school and let some third grade students to try out our game prototype. After the trial we found, that game based learning might not be for everyone. Especially students needing special education had trouble succeeding in the game. For other students however, such games utilising innovative devices, which create a Natural User Interface, apart from the fact that students practice the school subject, might help to improve their hand-eye coordination skills. For well performing students with great coordination skills such games might mean a fun way to practice the school subject.

To declare, that game based learning with innovative input devices such as the *Leap Motion* can significantly improve the learning process of some school subjects - if the applications are done right - would need a longer and scientific trial. However, as we have seen in the classroom, such games can be motivational, engaging and fun to play enough to make students forget about the subject while still practicing it.

8 Future Work

While the prototype fulfilled its purpose, we would like to present some ideas how it could be made even more interesting and more fun to play. As we have not added any sound effects to the game, this would be an essential part of a possible future work. To add sounds in the game menu as well as into the game. For example environment sounds or music, or sound effects when popping the balloons would make a great addition.

In addition to sounds it would be nice to add more trees, bushes and rocks to the scene so that its more interesting, or to even generate these dynamically in each level. Every level would then look different. There is then room to add more levels, or more game modes, not only multiplication, but also addition, subtraction and division.

To make the gameplay itself more interesting, we could add different physics forces like wind. The balloons might float in the wind and the player would need to catch and pop them.

As the Leap Motion is able to track two, or even more hands at the same time, a multiplayer mode would also be a good addition, which would make the gameplay even more fun. There is room to make the prototype become a good game.

Regarding the trials, to find out how the game improves the learning process, a larger scale scientific study could be done.

9 References

Altman P. (2013) *Using MS Kinect Device for Natural User Interface*, Pilsen, Czech Republic

Barata G., Gama S., Fonseca M. J., Goncalves D. (2013) *Improving Student Creativity with Gamification and Virtual Worlds*, Stratford, Ontario, Canada

Barata G., Gama S., Jorge J., Goncalves D. (2013) *Improving Participation and Learning with Gamification*, Stratford, Ontario, Canada

Borges S., Reis H. M., Durelli V., Isotani S. (2014) *A Systematic Mapping on Gamification Applied to Education*, Gyeongju, South Korea

Chatzopoulos N. (2013) *How To Use Leap Motion (Crazy Future Technology) In The Classroom* http://www.edudemic.com/leap-motion-in-the-classroom/ (last visited 2014.7.13)

Coelho J. C., Verbeek F. J. (2014) *Pointing Task Evaluation of Leap Motion Controller in 3D Virtual Environment*, The Hague, The Netherlands

Colgan A. (2014) *Giving Deaf People a Voice: MotionSavvy's Real-Time Sign Language Translation* http://blog.leapmotion.com/giving-deaf-people-a-voice-motionsavvys-real-time-sign-language-translation/ (last visited 2014.9.6)

Colgan A. (2014) *How Does the Leap Motion Controller Work?* http://blog.leapmotion.com/hardware-to-software-how-does-the-leap-motion-controller-work/ (last visited 2014.8.12)

Dede, C. (2002) *Vignettes about the future of learning technologies. In Visions 2020: Transforming education and training through advanced technologies*. Washington, DC

Deterding S., Dixon D., Khaled R., Nacke L. (2011) *From Game Design Elements to Gamefulness: Defining "Gamification"*, Tampere, Finland

Deterding S., Sicart M., Nacke L., O'Hara K., Dixon D. (2011) *Gamification - Using Game-Design Elements in Non-Gaming Contexts*

Griffiths, M. (2014) *Playing video games is good for your brain – here's how* http://theconversation.com/playing-video-games-is-good-for-your-brain-heres-how-34034 (last visited 2014.11.15)

Gupta A. (2014) *Digital Assistance Beyond the Touchscreen* http://blog.leapmotion.com/digital-assistance-beyond-the-touchscreen/ (last visited 2014.9.6)

Gupta A. (2014) *Wearing Your Robot: A New Level of Robotic Arm Control* http://blog.leapmotion.com/wearing-your-robot-a-new-level-of-robotic-arm-control/ (last visited 2014.9.6)

Han J., Gold N. (2014) *Lessons Learned in Exploring the Leap Motion Sensor for Gesture-based Instrument Design*, Goldsmiths, University of London, UK

Holz D. (2014) *Leap Motion Sets a Course for VR* http://blog.leapmotion.com/leap-motion-sets-a-course-for-vr/ (last visited 2014.9.1)

Huotari K., Hamari J. (2012) *Defining Gamification - A Service Marketing Perspective*, Tampere, Finland

Kuntz Ch. (2013) *Controlling Osirix with the Leap Motion While Scrubbed into Surgery* https://www.facebook.com/photo.php?v=10152121411384392 (last visited 2014.8.15)

Liebeskind D. (2014) *The Beginning of a Drone Revolution* http://blog.leapmotion.com/the-beginning-of-a-drone-revolution/ (last visited 2014.9.1)

Magerko B., Heeter C., Medler B., Fitzgerald J. (2008) *Intelligent Adaptation of Digital Game-Based Learning*, Toronto, Ontario, Canada

Means, B. (2000) *Accountability in preparing teachers to use technology*. In Council of Chief State School Officers, 2000 State Educational Technology Conference Papers. Washington, DC

Metz R. (2013) *Leap Motion's Struggles Reveal Problems with 3-D Interfaces* http://www.technologyreview.com/news/518721/leap-motions-struggles-reveal-problems-with-3-d-interfaces/ (last visited 2014.8.21)

Mitchell K. (2014) *Your Beer Deserves a Better Paint Job* http://blog.leapmotion.com/your-heineken-beer-deserves-a-better-paint-job/ (last visited 2014.10.3)

Plemmons D., Holz D. (2014) *Creating Next-Gen 3D Interactive Apps with Motion Control and Unity3D*, Vancouver, British Columbia, Canada

Plemmons D., Mandel P. *Introduction to Motion Control* https://developer.leapmotion.com/articles/intro-to-motion-control (last visited 2014.8.21)

Potter L. E., Araullo J., Carter L. (2013) *The Leap Motion controller: A view on sign language*, Adelaide, Australia

Qiu S., Rego K., Zhang L., Zhong F., Zhong M. (2014) *MotionInput: Gestural Text Entry in the Air*

Rao V. (2014) *DexType: Virtual Keyboard For Leap Motion Controller* http://www.assistivetechnologyblog.com/2013/07/dextype-virtual-keyboard-for-leap.html (last visited 2014.8.15)

Richard J. Noeth, Boris B. Volkov (2004) *Evaluating the Effectiveness of Technology in Our Schools*

Rojas D., Kapralos B., Dubrowski A. (2013) *The Missing Piece in the Gamification Puzzle*, Stratford, Ontario, Canada

Vikram S., Li L., Russel S. (2013) *Handwriting and Gestures in the Air, Recognizing on the Fly*, Paris, France

Watson D., Hancock M., Mandryk R. L. (2013) *Gamifying Behaviour that Leads to Learning*, Stratford, Ontario, Canada

Weichert F., Bachmann D., Rudak B., Fisseler D. (2013) *Analysis of the Accuracy and Robustness of the Leap Motion Controller, Department of Computer Science VII*, Technical University Dortmund, Dortmund, Germany

Whitton, N. (2007) *Motivation and computer game based learning*. In ICT: Providing choices for learners and learning. Proceedings ascilite Singapore 2007. http://www.ascilite.org.au/conferences/singapore07/procs/whitton.pdf

Zuckerman O., Gal-Oz A. (2014) *Deconstructing gamification: evaluating the effectiveness of continuous measurement, virtual rewards, and social comparison for promoting physical activity*, Herzliya, Israel

Table of Figures

Table of Listings

Table of Tables