Hochschule Karlsruhe Technik und Wirtschaft

Fakultät für Informatik und Wirtschaftsinformatik

von

Betreuer:

August 2019

Ich versichere, die Arbeit selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet zu haben. Die wörtlich oder inhaltlich übernommenen Stellen sind als solche kenntlich gemacht.

Alina Jaud Karlsruhe, August 21, 2019

Contents

List of Figures

1 Introduction and goals

show uni dropout rates and reasons from German statistics; point out missing knowledge about job rules; describe how it should be that less people quit uni

The purpose of this thesis is to develop a virtual reality (VR) application for students and prospects to experience different professions in the IT industry. The decision about the specialization of the education can be very challenging for prospective students because it has a great impact for their future. Schools and universities struggle to provide comprehensive information about the courses and especially for the job roles connected to the courses. This thesis investigates, how the usage of VR can support prospective students in their decision making.

1.1 Motivation

A lack of missing information about future professions can result in high drop out rates, as seen in [H$^+$17], a study about dropout rates in Germany and their reasons:
In total, about one third (32%) left German universities without a degree in 2016. Most of the students quit their course because of too high requirements and difficulties in the exams (30%). The second important reasons were a lack of motivation (17%) and the missing of practical work and exercises (15%). Looking at the second and third most important reasons for a university dropout together, it is with over one third a very large group, but could be avoidable. With missing imagination and information about the profession the students are later able to work in, the endurance might not be very high. As a conclusion, motivation lacks occur and it is harder for the students to get through more theoretical parts of their studies. As well, it is also shown in the study that occupying with the subject before applying for a course results in less amounts of university dropouts.
The question is now: How to provide prospective students suitable information for the profession of their desired course? And more important: How is it possible to engage students in getting to know their future job roles?

1.2 Goals

There are many ways to get information about IT courses and IT job roles. For example, it is possible to read articles or brochoures, watch a video or ask experienced persons. All those approaches give information about IT job roles but the way they provide it is passive. Prospective students should actively be involved in their future IT professions and experience the functions and challenges of the job roles themselves. The media described above are not able to provide integrate the user in a first persons view.
VR describes a computer generated 3D environment which can be experienced by users. Users are wearing a special headset through which they are fully shielded off from the real surroundings. It is possible to get an immerse and real experience of the virtual environment, and users feel as they are actually present in the virtual world. [Lin15]
With the usage of VR, it is possible to provide prospective students a visualization of their future professions and they are able to experience the job roles themselves. This thesis aims to connect the medium VR with the topic of student acquisition. It should help students with

getting a better experience for the different IT professions. Universities should profit from this research by having a way to bring the various fields of IT closer to prospective students.

1.3 Approach

The core target group are prospective students in Singapore, who finished their secondary education and start with a specialized education at a polytechnic.
The approach is to develop a VR application, which introduces and simulates several IT job roles. This application will provide information about the responsibilities and the daily life in a IT job role. The application will also contain user interactions. Users will complete tasks which are connected to the various IT job roles. This is to follow a learning-by-doing approach. The success of the VR prototype application relies on the design, the storyline and the interactivity, not only on the use of VR itself. Therefore the application will be designed with a focus on user interactivity and a good story line.
To measure, how much the application helps students in their experience of IT job roles, the application will also be tested. A questionnaire will be developed to see, how the understanding of IT job roles by the participants has changed. The two comparing states will be before and after playing the VR application.

1.4 Hypotheses about advantages of VR for prospective students

Explain why VR and future professions work well together, point out immersion of VR, name examples of similar VR projects

Within the VR application prospective students should not only learn about IT professions but also experience the various job roles through a realistic simulation.
VR can create virtual environments in which users experience a stronger feeling of presence than compared to other media. Therefore it is expected, that VR helps prospective students to dive into the job roles and provide a realistic as possible overview about the IT professions. It is expected that students will show more interests in IT job roles than they did before playing the VR application.
VR is a currently researched topic of information technology. Students are able to use technologies, which they later get into closer touch with. The idea is to arouse interest in the technology behind VR and to make the students want to develop their own application. Prospective students can profit as well from the cheaper and less time consuming approach in getting to know a future job role, compared to an internship.

1.5 Overview of the thesis

In chapter 2 there will be a research in basic VR technologie and current developments. Common problems which occur when designing a VR application as well as their solutions will be described. Following, in chapter 3, a storyline for the application will be designed, suitable hardware and software will be evaluated. The next step will be to implement a prototype VR application in chapter 4. This application should provide information about the different IT job roles related to the IT diploma courses at Nanyang Polytechnic. After this, the application will be tested by students. The main focus will be set on how their expectations about IT job roles changes after playing the VR application. Chapter 5 sums up the test procedure and the results. In conclusion, the whole project will be evaluated under the aspect of fulfilling the purpose of helping prospective students in understanding their future job roles in chapter 6.

2 Background studies

Before beginning with the design and development of the prototype application, a fundamental base of knowledge has to be set. This chapter gives definitions about the most important terms and concepts used in VR. The context in which this project is set in is also explained briefly. Furthermore, this chapter specifies the most important technologies for the prototype and also points out where the limits of VR are. Finally, a measurement for the effectiveness of VR is presented.

2.1 Context

The prototype application will be focused on the information technology diploma courses offered at the Nanyang Polytechnic. In the following, the environment of the school and the courses will be described.

Nanyang Polytechnic (NYP) is a polytechnic located in Ang Mo Kio, Singapore. The school offers post-secondary education for students who successfully passed the GCE O-Level examination. The O-Level examination is an annually examination mainly for students who have visited a secondary school. [Sin18]
This examination allows students to take courses either on a junior college, a technical institute or a polytechnic. In Singapore, a polytechnic offers students a more practical and industry based education than a junior college, which has a more theoretical approach. In general, students study 3 years at a polytechnic until they get their diploma. After getting a diploma, students can continue their education at a university but they can also start to work in the industry. [Min18]

2.1.1 NYP information technology courses

NYP offers a wide range of courses in different fields. One of the fields is information technology (IT). In the following the courses will be introduced briefly: [Nan18]

Information Technology This course offers students a broad range of different fields in information technology. It focuses on an interdisciplinary education. By finishing the course, students are prepared to work in different areas, most commonly in software engineering. During the first year, the fundamental topics of information technology are educated. The second year deepens the knowledge in application programming, database management, software engineering, algorithms and several other topics. The third year will be for specialization. The students can choose between elective courses in the fields of artificial intelligence, enterprise cloud computing, geospatial and mobile innovation and cybersecurity. As in all courses offered at the school of information technology, there is offered one practical industry project and an internship in the third year as well.

Infocomm and Security This course covers topics in IT infrastructure, network engineering and security and the internet of things (IoT). Students learn how to program, but also how to secure applications in a connected environment and they learn about managing connected infrastructures. During the first year there are basic classes in programming and infocomm to get a grounded knowledge in information technology. The second year will deepen the knowledge of network engineering, programming, and IT service management. There will be a practical IoT project for students as well. In the third year there are several elective courses in the areas of system and network security, enterprise infrastructure and infocomm solutions.

Cybersecurity and Digital Forensics The cybersecurity and digital forensic course focuses on IT security. This includes securing systems and data for unauthorized access as well as tracing criminals in case of a security incident. After graduating, students can work as security analysts, network penetration tester, security engineer and similar positions. During the first year students learn fundamental IT skills. The second year offers classes in forensics, network security, operating systems, security standards and more. Students will also complete an applications security and an infosecurity project. The last year offers specialization in the topics cybersecurity track and cyber forensics track. It is possible to choose crossdisciplinary classes from other IT courses.

Business Intelligence and Analytics The business intelligence and analytics course teaches analytics and interpretation of massive amounts of data. Therefore big data technologies as well as artificial intelligence is needed. After graduating, it is possible to work as a data or business analyst, a social media strategist or a digital marketing executive. During the first year students will gain basic IT and business statistic knowledge. The second year offers classes in big data management, digital marketing, predictive modelling and similar topics. Students participate on a big data and a business analytics project. During the third year, students will do a data science project and can choose between several elective classes.

Business and Financial Technology This course focuses on information technology in the financial and business sector. After getting a diploma, students can work as IT or financial consultants, financial application specialists or business and financial analysts. From the beginning the focus lies on connecting business and financial topics with information technology. Therefore, in the first year students learn about economics, accounting and consumer banking as well as about basic programming skills. During the second year this knowledge is deepened through classes like software engineering or financial management. Besides the elective classes, the industry project and internship, students in their third year also collaborate in a fintech innovation project.

2.2 What is virtual reality?

Definition; use; popularity; difference from AR

According to [Fuc11] VR is a simulated virtual world, created with hardware and software. It provides a real time user interaction and users feel the maximum amount of immersion when experiencing the computer generated world. The virtual world is mostly provided by a head mounted display (HMD) as seen in figure 2.1. This device splits the main scene into two different camera perspectives for each eye of the user. The two perspectives will then be merged into three dimensional picture by the users brain.
The computation of the virtual world can either be done on a standalone desktop PC or directly

Figure 2.1: Head mounted display by

in the HMD, for example through a mobile phone which is wrapped inside the HMD.
Different from augmented reality (AR), VR provides a fully self-contained environment and does not interact with elements of the real world. AR on the other hand embeds virtual content into the real environments. Thereby the borders between reality and simulation become blurry. AR is a closely related technology to VR, but this thesis will focus on VR applications only.
The usages of VR are very versatil. [Lin15] describes a variety of different areas in which VR can be used: Especially the gaming industry stands out for one of the first industries to develop VR applications and hardware devices. This is because most of the modern computer games take place in a computer generated, first person, 3d environment. To achieve a higher level of realism, it seems very likely to introduce VR in the gaming branche. Lately, non-gaming VR applications are getting more and more important. Design and architecture is a branch, where VR is used to visualise prototypes in a new way to customers. In education, VR applications are used to train learners in a nearly real environment but without the cost and danger of a real life exercise. Military training or disaster management are good showcases for the usage of VR in education. Another area, in which VR is used, is tourism. Through a VR application it is possible to visit a museum without actually travelling to the place. VR can give more immerse insights of a travelling destination and yet can be an instrument of marketing for the tourist destinations.

2.3 VR basic concepts

covering some history in this chapter too? E.g. explaining the first wave of VR in the 90s

[SC19] describe five core concepts for virtual reality. According to them, this concepts are the key elements when it comes to design a virtual reality. In the following, this concepts will be explained:

Participants A participant is a user of a VR application, who is actually wearing the HMD and experiencing the virtual world. Every participant has a different perception of the application, so every experience can be seen as a unique experience. This is because the several participants differ from their cultural background, their age, their expectations and their knowledge in technology. Therefore, it is a challenge for developers of VR applications to create a virtual world which various users can identify with. The role of the participants is described as the most important role, because all the experience is happening in their imagination. Developers or creators are only capable of creating the framework around the experience.

Creators Creators or developers design and implement a VR application for participants. As mentioned, the developers are not able to create the experience of the users. The core aim of their work is to develop the application, which is a collection of programming scripts, concept designs and data. The experience is created together with the participants. Later in this thesis there will be an insight in developing a VR application from a creator's perspective, whilst chapter 5 will dive into the different experiences of participants using the VR application.

Virtual world In general, the term "virtual world" is defined as the content of a virtual reality application. The virtual world is the skeleton of the virtual experience. Precisely, it is a description of the objects inside the virtual space. Everything which the user can see, hear and interact within the VR application belongs to the virtual world. How the virtual world space is designed influences the experience of a participant.

Immersion Immersion in general describes how deeply users connect with a virtual world. They experience something through a medium which they would not be able to experience without it. They see actions and things through a different point of view and can connect with it in some way. The deeper a user can connect to the virtual environment and the objects inside it, the more immerse is the VR experience. In chapter 2.3.2 the term immersion will be explained in more detail.

Interaction Most of the time, interaction means manipulation of a computer generated world. However, interaction can also happen in a non artificial environment. An example for this are interactive novels, or text based computer games. In this kind of media, computer graphics are not required and yet they provide a form of user interaction. When it comes to VR, interaction also means relocating a viewport inside a virtual world. Participants can move physically in the virtual environment. Another form of interaction is the collaborative interaction. In some VR applications it is not only possible to interact with the virtual world but also to share the space with other participants and interact with them. This can happen in a virtual or physical way.

2.3.1 Different dimensions of reality

describe different states: reality, augmented reality, augmented virtuality, vr

Besides VR, there exist other technologies, which serve the purpose to create virtual worlds for the user. They differ in the extend of intervention with the reality. [Tha18] show a spectrum of different types of technologies and their dimension of reality. A visual representation of this spectrum is displayed in 2.2.

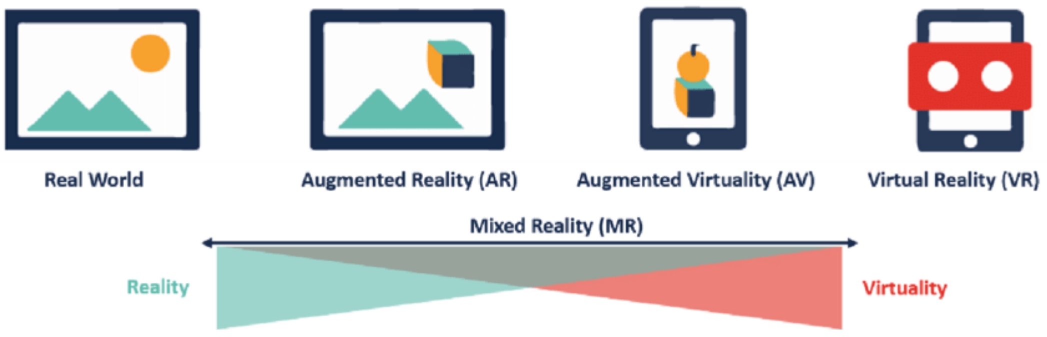


Figure 2.2: Graphical representation of the dimensions of reality by [Lov18]

The reality describes the real world and using non digital devices for interaction, such as a steering wheel for driving a car for example. As mentioned in the indroduction, AR is a related technology to VR. In the spectrum of [Tha18] AR covers the second level of reality dimensions. It describes computer generated elements, which are embedded into the reality. A famous example for an AR application is the game PokemonGo (`https://www.pokemongo.com/en-gb/`), in which players search in the real environment for computer generated items, based on a phone's camera and location services. The next level of reality is the augmented virtuality. This term describes a virtual environment, in which users can interact through analogue methods. An example for this could be a video chat room where the participants are present in a virtual room but communicate through analogue methods. As seen in figure 2.2, augmented reality and augmented virtuality can be summed up with the term m̈ixed realityẅhich describes the interaction between real life elements and the virtual environment. Finally, in the last stage there is VR in which a user is fully surrounded by a computer generated environment. This environment is most likely a reproduction of a real-life environment. The user is not only surrounded by an artificial reality but can also interact with it. It might happen that the user actually feels present in the virtual reality. In the following, this phenomenon will be explained in more detail.

2.3.2 Immersion and storytelling

What is immersion and why so important for VR? What describes a good storytelling? Is there anything special when designing a story for a VR application?

The term immersion often comes up when talking about VR. This term describes an experience which users would not be able to percieve without the use of the specific medium. In general, it describes an experience from a different point of view. This could be for example the story of a different character or the exploration of a unknown location. [Tha18]
Immersion is an important measurement for this work. The experience, students have when

playing the VR application, which will be developed during this thesis, depends on the grade of immersion. When talking about immersion, it is important to distinguish between mental and physical immersion. Mental immersion describes the state, in which the participant feels deeply emotionally connected to the virtual world. Physical immersion on the other hand means that the user's physical interactions are transferred into the virtual environment with the use of technology. The movements should feel as natural as possible to the user. When talking about immersion in books, films or related media, it is often only referred to mental immersion, because these types of media are not able to provide the technology for interacting with their environments. VR on the other hand can create immersion through a physical way as well. That is why VR can provide a higher grade of immersion than other media. [Tha18]
Another term closely related to immersion is presence. Presence in the context of VR describes the subjective feeling of actually existing in the virtual world. The feeling of presence is a mix between mental and physical immersion. The more presence users feel, the more they accept the virtual world as the reality. To gain the sense of presence, the participant's movements in the reality should match with the movements of the virtual world. With a higher grade of presence, participants act more natural in a virutal environment, such as they would in a real environment. Through this, their task performance increases. This is one reason why presence is an important factor when it comes to developing VR applications, especially in the field of education and learning. [SW97]

Since the terms immersion and presence are explained, it is now the question, how to design the story of a VR application in order to gain a maximum grade of immersion. Storytelling is one of the most important elements for mental immersion in a virtual experience. Especially mental immersion can be influenced by the storyline, while physical immersion can be increased through hardware tools.
Storytelling in general is the art of telling a narrative to the user in a way that the user is not only consumer but actually feels immersed with the story [Lou18]. Designing a storyline in VR opens a lot of possibilities in engaging the participant in the story. Therefore there are things to consider when it comes to storytelling in VR as explained by [Kea18]:
One thing to bear in mind is, that space and environment in a VR application are more important than in other media, like books or films. The creator designs a world in a 360 degree view, so there is a lot more space to show. The storyline not only happens in front of the users, but can be all around them. Because the user is able to walk independently in the virtual world, it is very important to softly guide users into a certain direction when necessary for the storyline. The guidance should not be conspicuous, but should attract users' curiosity. This can be achieved through light and sound effects or movements. Another aspect to consider is not to overwhelm users with complex tasks. This could prevent the user in experiencing an immersed virtual world because their full attention would lie on completing the tasks.

2.3.3 Interaction: Selection, manipulation and navigation

different interaction tasks in VR and their common solution: Simple virtual hand, raycasting, go-go, HOMER, scaled world-grab, occlusion; describing ways of movement (walking aroung) solutions in VR

In order to get an immerse experience, the user of a VR application needs to interact with the environment. When it comes to interaction in VR, several challenges occur. First of all, there

is a three dimensional space which has to be covered. Secondly, when users are wearing the HMDs, they cannot see the controllers they are using in the real world. This means, that every hardware controlling device should be mapped into the virtual world. This chapter describes some typical interactions and their common solutions in VR. It is a base overview of interaction methods, which can be considered to be used in the prototype VR application. The final method, which suits the best for the prototype, will be discussed in chapter 3.
Basically, there are two different types of interaction in VR: Modifying the virtual world and moving in the virtual world. When modifying the virtual world, the user should be able to select and manipulate objects. Manipulating includes moving, scaling, changing of look (color, texture) and orientation of an object in the virtual world. When walking around in the virtual world, the user should be able to move freely and make their own decisions in navigation. [DBGJ13]
A challenge when finding solutions to the described interaction problems is, that there do not exist standard solutions, like we have in a screen based 2D context for example. Nevertheless, There are some best practices on how to implement the described requirements in VR. In the following, the most common interaction techniques will be explained according to [DBGJ13].

A very straightforward method in selection could be the selection through gaze. The user can select objects by looking at it. This is a very simple method and works without hardware controllers for user input. An analogy in 2D interfaces would be the selection via mouse over. However, this technique does not work well in VR. The problem is, that users accidentally select objects every time they look around. A simple and natural exploring of the virtual world would be interrupted by object selection.
Another technique is called ray-casting. It works with a virtual virtual ray coming from the users' hand. Therefore, an controller or a hand tracking hardware is needed, which movements are mapped into the virtual world. Every object which collides with the ray is a possible candidate for selection. Most of the time, the object nearest to the user will be selected. ray-casting is an important method in object selection, because it is very straightforward and conforms to the expectations of users (TODO why?). Nevertheless, this method becomes inaccurate when to deal with large distances, because the movements of the hand has to be more accurate the longer the distance gets.
The next method, the Go-Go technique overcomes this problem. A virtual arm, mapped with the user's hand movements is displayed in the virtual world just as done with the ray-casting method. The difference of the Go-Go- technique is, that the virtual arm can be lengthened, if the selectable object is out of range. When the virtual arm is lengthened, the user's hand movements are mapped in a smaller range, than when the virtual arm is in the normal position. This mapping is not linear and overcomes the problem of inaccuracy when selecting objects to far away. The technique can be well used for relocating objects, because the user does not have to walk on the same time. However, with this technique, as well as with the ray-casting technique, only objects which are in viewing range can be selected or manipulated. Some objects could be overlapped by other obstacles and cannot be selected. Also, objects in far distance appear very small, which makes manipulation difficult.
The HOMER technique deals manipulation of objects in distance. The abbreviations stands for

Hand-Centered Object Manipulation Ray-casting. As the names says, it uses techniques of the ray casting for selection. To make the manipulation easier, the object moves to the users hand, once selected. The user can now perform their desired manipulation. After the manipulation is finished, the objects teleports back to its original position. This technique still has the problem of selecting objects which are not in the user's viewing range.

An alternative to this method is the world in miniature (WIM) technique. As seen in figure

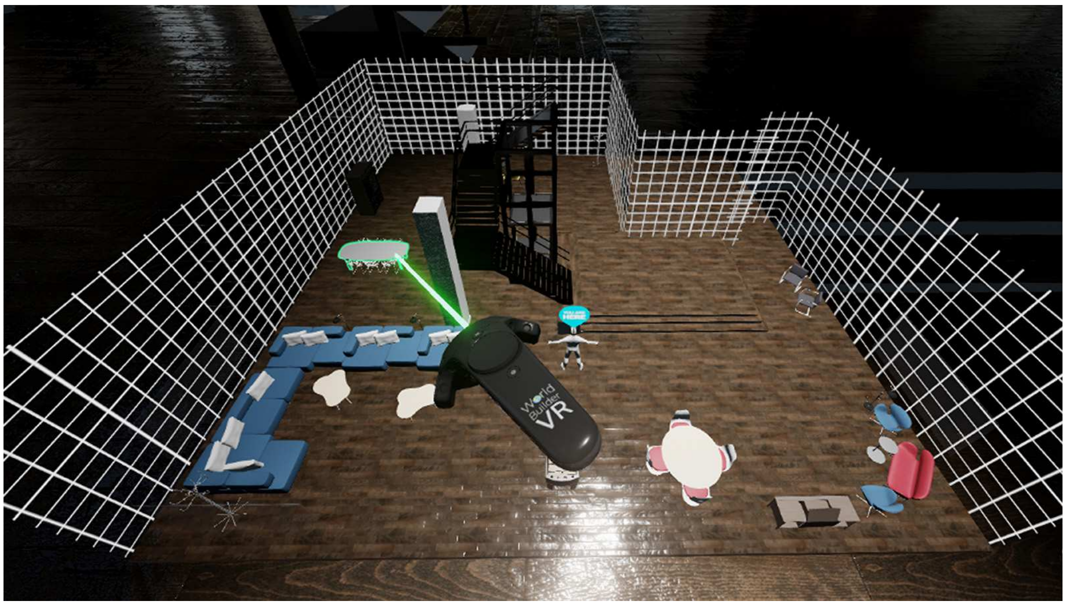


Figure 2.3: Mini map as part of the WIM technique. By [Arn17]

2.3, the user gets a miniature map of the virtual world with all selectable objects. The user can now select the desired object in the map and it gets highlighted in the virtual world. With this method, the user looses their ego centred point of view. This can be seen as a decrease in an immerse experience, although it increases usability.

When it comes to moving in the virtual world, the problem of only seeing the virtual world but not the physical world becomes more real. There is a danger of physical damage when colliding with real world object while wearing a HMD. There is a conflict between the real world limited space and the unlimited virtual world (TODO source). Therefore some techniques are described also by [DBGJ13] how to deal with the problems of navigating through virtual worlds:
One method of moving around in VR can be the direct manipulation of the camera with a suitable input controller. The user moves the camera with the help of a joystick for example. Another method, which is very common in 3D computer games and therefore conforms highly to the expectations of users, is the walking in the direction of the user's point of view. A disadvantage of this method is, that the user cannot look around while moving. Both described techniques have the disadvantage, that they can cause motion sickness. This phenomenon will be described in chapter 2.6.
To overcome the motion sickness problem, a very common technique for navigation is teleporting. Teleporting means to translate the user directly to a specific position without a time delay. The technique can be realised in VR by selecting the desired translation point via ray-casting

and jumping to the point after selection. This method successfully overcomes the problem of motion sickness, but there is a loss of immersion, since this kind of navigation is not very natural. [**?**]

The most natural and immerse way of navigating is physical walking. Walking in a virtual environment deals with all the problems of space and the above mentioned conflict between real and virtual world space. Natural walking in general needs more complex hardware for tracking the users movement than navigation with the help of a hand controller. One method of walking in VR is the walking in place method. As the name describes it, the user moves in the virtual world by lifting their legs but they are not actually changing their position in the real world. There are several hardware installations which support walking in place, some of them make walking possible through a treadmill like system.

It is also possible to let the user walk around in room which was designed for virtual reality usage and provides suitable tracking sensors. To avoid colliding with the real world borders of the room, the redirected walking technique was developed. With this technique it is possible to make the user believe they are walking straight, while small changes in the virtual world make them actually move in circles or in similar paths as seen in figure 2.4

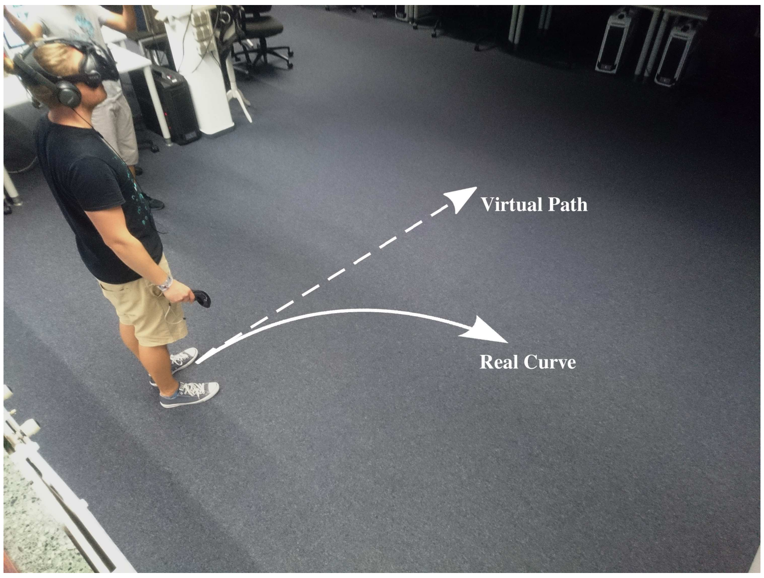


Figure 2.4: Redirected walking in VR, virtual walking path vs. actual walking path. By [LS18]

2.4 Hardware for VR

There has been a lot of research in hardware technologies to support physical immersion in VR. Some important objectives when it comes to new hardware technologies in VR are the increase of immersion, the reducing of costs and the decreasing of size and weight. The two main types of HMDs currently available are the desktop HMDs and the mobile HMDs. The first type runs the application on a standalone desktop and the HMD measures the sensorical

describe difference between mobile and desktop HMDs, show examples HTC Vive and Google Daydream, maybe also Google Cardboard. What possibilities of interaction do they provide?

data from the user. The second type uses the phone to run the application and the phone's sensors to measure head rotation and other relevant data. In general, the desktop HMDs have the better performance than the mobile HMDs because the processing of the game is transferred to a standalone desktop PC instead of a phone inside the HMD. [DBGJ13]
An addition third type of HMDs are the standalone devices, they are similar to mobile VR headsets but there is no need of attaching a mobile phone into the HMD. Instead, the device works out of the box. The performance of those standalone devices is lower than the desktop but higher than the mobile HMDs. [**?**]
For this project, the game will be deployed on two different devices. The devices provided for development are the Samsung Gear VR headset and the Occulus Go headset. This chapter compares the two devices and point out their advantages and disadvantages. Both devices can be seen in figure TODO.

Samsung Gear VR Samsung Gear VR is a mobile based VR headset which was developed by Samsung and Occulus, introduced in 2017. The performance and screen resolution of the HMD depends on the mobile phone, used with the headset. Gear VR is currently supported only by the newer Samsung phones, including Galaxy S6 and higher, Galaxy Note 5 and higher, Galaxy A8 and higher. It comes with a bluetooth hand-held controller which can either be held in the left or right hand. Inside the HMD is a gyro sensor to detect head rotation. There is a proximity sensor as well in the headset to detect if the user is currently wearing the HMD. The device does not support motion controlling. [Sam19]
Gear VR works intuitive with the supported devices and is supported by the game engine unity, which makes the software application development easier. However, the use of a Samsung mobile phone is required, and not all Samsung phones are supported.

Occulus Go This device was announced in May 2018 and is a standalone device. That means there is no need of an additional smartphone to be attached in the device. Inside the HMD is instead a Qualcomm Snapdragon processor which is widely used in smartphones. The screen resolution is 2560 x 1440 pixel. This resolution is comparable to high class smartphones of the year 2018 and therefore comparable to the Gear VR in combination with a suitable smartphone. The sensors and the handheld controller are very similar to the Samsung Gear VR device.
Looking at the technical data, there is not so much of a difference between the two devices. Nevertheless, an advantage of the Occulus Go is, that no phone has to be attached to the device. Also the head strap and the audio output was improved.[**?**]

2.5 Showcases of VR applications

There are a variety of VR applications available for very different use cases and made with diverse concepts. The following chapter will present one VR application which is of the category of 360 degree videos, one application from the field of gaming and the last application will be about education and learning.

2.6 Limits of VR

At current status, what is VR not able to achieve? What are the challenges VR researchers face currently?

The idea of VR is not a new invention of the past few years. There has already been a peek in research and also prominence to the public in the late 90's and beginning of 2000. However, the technology did not prevail in the economy, because there were several limitations especially in performance of computation. Therefore only a limited amount of ideas were able to be actually implemented. Another problem at that time were the very large and heavy HMDs (TODO insert image). The user could not wear them for a very long time and it caused neck pain and other physical problems. The devices were also not portable. [Jer16]
Looking at VR technologies now, a lot has changed. At first, the performace of our processors have increased rapidly. Secondly the HMDs became a lot smaller and lighter compared to the ones of the late 90's. But yet, we are not at the point, where all research is done – on the contrary: A lot of challenges still occur when creating a VR application today. In the following, the problem of motion sickness will be explained. It will also be pointed out, what solutions are upcoming in the near future. There is also a lot of research in progress, when it comes to maximise physical immersion. It will be explained, what interactions cannot be covered by today's hardware, and what technologies are in development or testing at current stage.

2.6.1 Motion sickness

Motion sickness is a very common problem in VR. It can be compared to the kind of sickness, some people experience when being in moving vehicles, such as buses, cars or roller coasters. The symptoms are nausea, dizziness or even vomit. Most of the time this symptoms occur after a usage time of more than 10 minutes. Unfortunately, motion sickness does not occur on a very small percentage of people and therefore cannot be ignored easily. In the study of [SBW17], over a half of the test persons experienced one of the described symptoms during playing a VR roller coaster game.
The most likely reason for motion sickness in VR is the mapping between real movement and virtual movement. For example, there is a high propability of feeling sick, when there is a latency between head movement and the movement in the virtual world. This latency will be registered by the brain and is experienced as unfamiliar and weird. Other situations, such as walking in the virtual world without physically moving can be a reason for motion sickness as well. [DBGJ13]
[SBW17] also listed other reasons such as excessive speeds, for example first person moving speed or the moving speed of nearby obstacles, insufficient graphics or general technical problems(e.g. quality of lenses or comfort of HMD). [Fen13] mentions the removal of player control as a reason for motion sickness. It is more likely to experience the symptoms, if the games takes control over the head rotation instead of the player.
To overcome the problem of motion sickness, the latency between physical and virtual movement has to be as small as possible. This is of course limited by the performance of the hardware devices. Many creators try to manage the problem by creating short time experiences with less than 10 minutes durance and trying to avoid fast moving objects in their application. [**?**]

This approach however, limits the possibilities of VR applications. It would be hard to create an action shooter without fast moving objects, for example.
Besides this constraints, there have been new hardware developments lately, which promise to solve the problem of motion sickness in VR. One of this hardware inventions is a device by [**?**] with vibration motors attached to it. User wear a headband with the motors placed near to the ear. Everytime users move in the virtual world, the vibration motors provide a haptic feedback. According to [**?**] this vibration should affect the vestibular system and deludes a movement to the brain. The gap between physical and virtual movement can be tricked in a way through the device. However, the headband is still a prototype and so far, the effects and reasons for it are a hypothesis. There have to be more clinical studies until a clear statement of the effectiveness can be made.
Looking at the VR prototype application, which will be developed during this work, the risk of motion sickness should be held to a minimum. To fulfill this requirement, a balance between a high level of immersion and the reducing of motion sickness has to be hold. How this is realized, will be described in chapter 3 and **??**. If the requirement is fulfilled, will be reviewed in chapter 5.

2.6.2 Hardware

There are several limitations in hardware devices when it comes to VR at the time of writing. First of all, the sensors of a HMD have to work very accurate. Also computation of the head movement and the virtual world movement has to be as fast as possible (see previous chapter). The display resolution is too low currently. Screen resolutions which are working well when it comes to desktop screens or mobile screens do not work so good in VR, because the lenses zoom the picture and the pixels become visible.
When looking at physical immersion, there are a lot of potential enhancements in providing the user a more natural way of interacting with the virtual world. Hand held controllers have to be used in with the most commercial available HMDs. A body tracking system would provide more immersion, but requires very accurate sensors. Relocating the player in the virtual world is also a challenge at the time of writing. In chapter ?? the use of treadmill system was already mentioned. The following section will present a current available VR treadmill for VR. Within this example the current limits of physical walking VR are explained.

In [CH14] the cyberith virtualizer is introduced. It is a device which supports natural walking in VR through the walking in place method. Users are wearing special shoes and lean into a belt system, which can be seen in figure TODO. The feet are sliding onto the surface and translated as a movement into the virtual world. The movement mapping is realized through various sensors which are placed into the ground floor of the treadmill. It is possible to use the cyberith virtualizer with controller specialized games, because the software of this treadmill is emulating controller input commands.
This device, however is not yet available on the consumer market. The company had problems of finding customers after announcing and presenting it in 2014. In 2016 they announced that they would concentrate on the B2B market and yet they are not in serial production in 2019. [**?**

]
The reasons for this could be the expensive price and also the large size of the device. The consumer market might not be ready yet for devices like that. This example shows how challenging it is for commercial companies to make innovations in VR. Therefore it is important to push public scientific research in VR further forward.

2.7 Measurement of effectiveness of VR applications

name existing questionnaires about immersion, explain why they do not cover all aspects which are required for this application

3 Design

This chapter will focus on the prototype VR application which will be implemented. Before starting with the development, the content of the application has to be set. Also the requirements of the application have to be clear. For which target group the application is developed? What is the storyline of the application? How will the application be used later on? In the following, it will be shortly introduced, how the mode of operation is done during the implementation. After that the requirements will be set. A target user group for the VR application will be set, in order to start with the storyline. The application will have a fictive storyline, similar to a computer game. In order to design the story, a storyboard was created and will be presented in this chapter. Once the content is set, the thesis will take a look at HMDs and choose the suitable hardware device for the implementation.

3.1 Mode of operation

The objective of this thesis is to develop and evaluate a prototype VR application to display the different IT courses offered at NYP. To handle a complex task like this, it is important to apply a suitable mode of operation. For the design and development part of this thesis an agile approach is chosen. This means that the planning is done in an iterative process to remain flexible towards unexpected challenges. An agile approach does not mean that the main objective will change, but the task planning is adapted for one period and can be changed and evaluated after each iteration. Specifically for this thesis, there are weekly updates with the supervisor. During meetings, the work is reviewed and evaluated. Every six week there is a presentation of the work done for the last period and the tasks for the next six weeks are planned.
Each IT course, which is planned to be represented in the VR application will be treated separately. This means, that during an iteration, not more than one job role will be in focus of the design and development part.

3.2 Requirements

The application focuses on prospective students as users. After using the VR application, they should have a clearer understanding, what the IT courses offered at NYP are about. Also they should gain a basic understanding about the jobs they can attend after getting a diploma in IT. In order to achieve this, several functional requirements have to be set:

- Introduction of IT job roles: The application should introduce to the user, what the job roles are about.
- Start scene: The application starts in a futuristic city which can be explored by the user.
- Interaction with items: The user can interact with several objects in the starting scene and enter different scenes through this.

- Separation of each job role: Each job role should be presented in a separate scene. The user should be aware of which job role the scene is referred to at all times.
- Completing tasks: Each introduction of a job role should contain information of the job and a task which should be completed by the user.
- Following a main story line: There should be a main objective and each job role task is a step towards completing the main objective.
- Gamification: The tasks should be structured as minigames and there should be a reward after completing them successfully.

Besides the functional requirements, there are several non-functional requirements which have to be considered when developing the prototype:

- Setting: The application should take place in a futuristic environment. Users should see how they can impact the future with IT.
- Genre: The application should be a mix between a informative educational and a gaming application.
- Educational: The user should gain more knowledge of the IT job roles after playing the VR application.
- Portable: The application should be presented at various public events, such as open houses or fairs.
- Budget: The product will not be distributed commercially. Therefore the costs for external tools, graphic or audio assets should be held to a minimum.

3.3 Stakeholders

This chapter defines the different actors for the VR application. After answering the question of what to do, we will answer the question of who will do it. This will be done by naming the different stakeholders of the application.

- School of Information Technology
- Project leader
- Prospective students
- Developer

These stakeholders have different interests and expectations to the project. The School of Information Technology would like to have more students signing up for an IT course. They expect that the VR application arouses interest in IT by prospective students. Their task is to provide the application at public events, such as open houses and similar events. The School of Information Technology is also responsible for providing the IT courses which are introduced in the VR application.
The project leader expects a structured and well communicated application development. They expect that the objectives of the work is clear and the deadlines are met. The project leader is

responsible for the guidance of the developers and the communication between the customer (School of IT) and the developers. The project leader sets and monitors the milestones in collaboration with the developers.
The prospective students are the end users of the VR application. They expect a diverting experience while using the application. They also want to learn more about the IT courses offered at NYP. Their task is to play the actual application. In case they are interested in one of the IT courses, they are responsible in applying for a diploma course at NYP.
The developers expect clear objectives from the project leader and a detailed description of the requirements. Their responsibility is to provide the actual application based on the frame given by the project leader and the customer. They develop a storyline, program the application and test it as well. In close communication with the project leader, they apply changes to the system.

3.4 Target group

outlining target group: prospective students; creating persona

Before starting with the design of the application, a target group has to be defined. This makes developing of the application purpose easier and also helps not to loose the focus during the design process. A very important aspect of defining a target group is also, to develop the application directly to the needs of the future users. In the end this helps to increase the acceptance of the application.

The main target group of the virtual reality application will be prospective students, who are thinking of taking a diploma at NYP. As mentioned in the introduction, those students mostly come from a secondary school and have finished their O-Levels, which is an entry requirement for polytechnics. The secondary school educated the students on a very widespread basis, and the students can now decide their specialisation.
When the students leave their secondary school, the average age is 16. In this age, people are known to be open to new technologies and trends. Also, the people in the target group are in the stage to make independent decisions for themselves without influence of parents. The application will be designed for prospective student, which show an interest in information technology but yet have no experience or knowledge in that field.
In the following, a characteristic of a fictive person is displayed. This person stands as a representative for the target group of prospective students:

Characteristics

Name: Jessica Lee

Age: 16

Place of residence: Singapore

Education: Finished secondary school

Personal interests: Dancing, computer games, TV series, technology

Level of tech experience: Rich consumer, no programming skills

Expectation: Wants to experience a fun virtual reality application

Since the fictive person Jessica Lee's main expectation is to experience a joyful application, there will be the need of gamification elements in the application. Her interests in computer games and technology lead the application's scene setting into a futuristic environment with new technologies. She has experience in digital media as a consumer, but not as a professional (programmer). Therefore, common gestures for user interaction can be applied, but the difficulty level of tasks about information technology should be on a beginner level.

3.5 Main story concept

To gain the users attention and let them grow interest in the application it is important to have a good main story. The main story will create a bridge between the several IT courses displayed in the application.
The application will start in the so called "Smart City". This is a futuristic urban environment in which the user can move around freely.There are five different buildings along the streets. Each building is interactive and represents a course offered by the NYP. By interacting with the buildings the user can enter laboratories, so called "labs", and complete different tasks. Every successful completion of a task has an impact to the main story.
The main goal of the VR application is to create a delivery drone which will deliver orders from online shops, just like Amazon announced in a video of 2013 (source). This is still a vision of future for now, but it is a good showcase for the user to show them how they can impact the future with IT. During the completion of the labs several challenges will occur. Once the user started programming their first drone, a lot more drones will start flying around in Smart City. To avoid crashing and chaos, the next task is to provide a proper and stable internet connection to the drones, so they can communicate. But the connected drones have their advantages and disadvantages: At first they are flying normally and communicating to each other but then an unknown IoT-Hacker infiltrates a virus into the drone firmware, so they start to misbehave. Now the user has to trace the hacker with basic cyber security concepts. After the hacker was found, more and more inhabitants of Smart City start to use the drones and provide a lot of data. It is now the job of the user to analyse this data and optimise the drones. By combining relevant data, it is possible to add some enhancements to the drone. Once the user completed all tasks and visited all labs, the user can exit the game.
It has to be mentioned that not all courses offered by NYP are covered in this main story so far. This is the case, because the application prototype will concentrate on the job role of the software engineer and the cyber security analyst. Other job professions will be added in an iterative process.

3.6 Storyboard

In this chapter, the structure and the details of the main storyline are explained with the use of a storyboard. Since the requirements for the prototype application are already set, the storyboard

points out the details of the requirements in a visual way. The main objective of a storyboard is to get a scene by scene overview of the story and the sequence of events. Through a sketch draft, an idea of the visuals of the application is displayed. Still the focus of the storyboard lies on the storyline. Logical orders of scenes are displayed through arrows, textual explanations are added, whenever necessary.
For the storyboard of the prototype, a tool called invision freehand is used [**?**]. In the following, some important parts of the storyboard are presented. The complete storyboard can be seen in [**?**].

3.6.1 World scene and software engineer scene

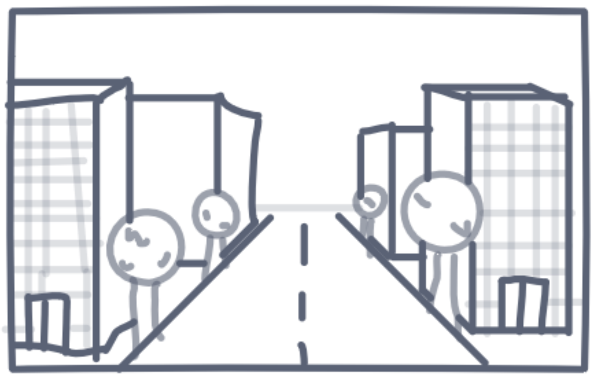


Figure 3.1: Smart city scene from where the user can explore different IT job roles.

The first scene of the storyboard is the smart city scene, as sketched in figure 3.1. Here the user can walk around freely. When clicked on the software engineer building, an assistant asks the user to enter the building. If answered with "yes", there is a transition into the software engineer laboratory scene. The laboratory contains a testing drone and a package as well as working desks with laptops. After the transition, the assistant explains the first task, as displayed in 3.2. This is the first minigame, in which the user should gain understanding in how programming works.

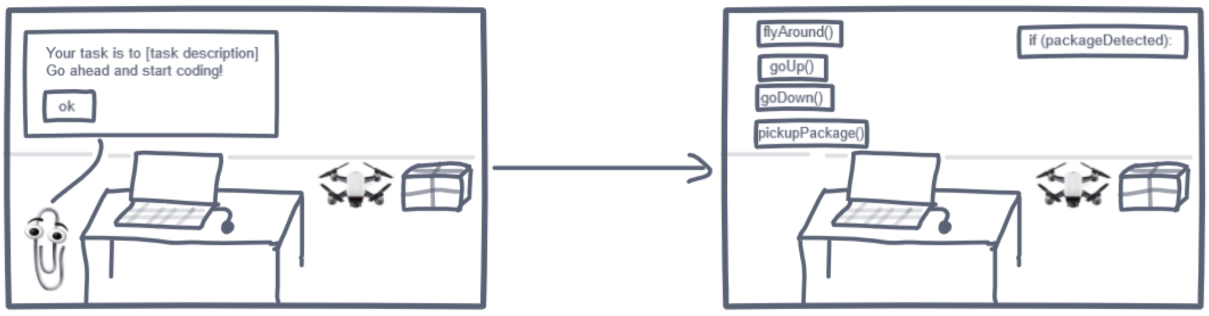
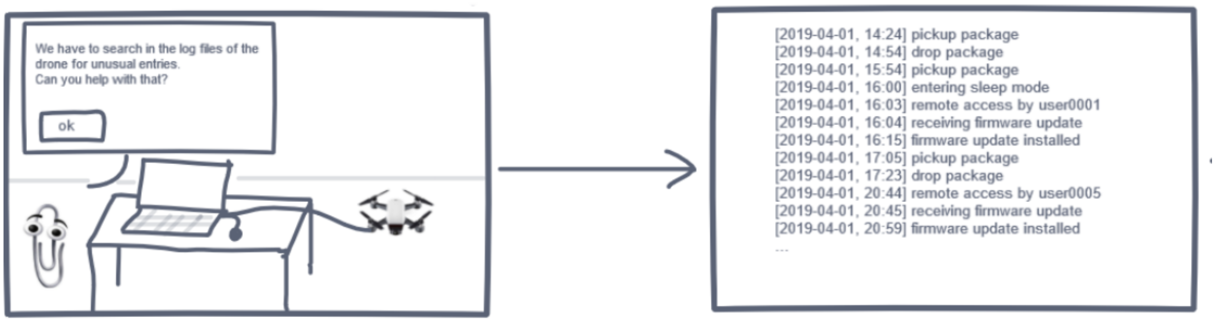


Figure 3.2: Software engineer scene: dialog and minigame

The information is taught in a very simplfied way, because the requirements of the application are, that it is playable without knowledge in IT. The objective of the task is to program the package delivery drone, so that it actually picks up a package. This is done by arranging lines of code in a logical order. The working environment is a desk with a laptop. After the user has arranged and ran the code successfully, they can watch the testing drone in the lab picking up the package. With the completion of this minigame, the scene ends and the user is transferred back to the world scene.

3.6.2 Cyber security specialist scene

The second scene represents the job role of the cyber security specialist. The scene is a laboratory with several office working places and the drone from the previous scene, which is displayed on a table in the middle of the room. When the user enteres the lab, it is explained, that the drone, they had programmed before, was corrupted by an unknown hacker. During the next task, the user should gain knowledge in the methods, cyber security specialists use to investigate digital crimes. The user learns about simple data decoding and encoding as well. The minigame is divided into three parts.



We have to search in the log files of the drone for unusual entries.
Can you help with that?

```
[2019-04-01, 14:24] pickup package
[2019-04-01, 14:54] drop package
[2019-04-01, 15:54] pickup package
[2019-04-01, 16:00] entering sleep mode
[2019-04-01, 16:03] remote access by user0001
[2019-04-01, 16:04] receiving firmware update
[2019-04-01, 16:15] firmware update installed
[2019-04-01, 17:05] pickup package
[2019-04-01, 17:23] drop package
[2019-04-01, 20:44] remote access by user0005
[2019-04-01, 20:45] receiving firmware update
[2019-04-01, 20:59] firmware update installed
...
```

Figure 3.3: Cyber security specialist scene: searching log entries

In the first part, the user searches the log data of the drone for unauthorized access and marks suspicious entries, which can be seen in figure 3.3. It turned out that two lab employees have accessed the drone during the last 24 hours. The second part of the minigame is to search the computers of the two lab employees for further information. One employee has an encoded message on their desktop (figure 3.4).

The last part is to decode the message by trying out different commonly used codes. The message reveals the hacker and the person can be prosecuted. This marks the end of the cyber security specialist scene.

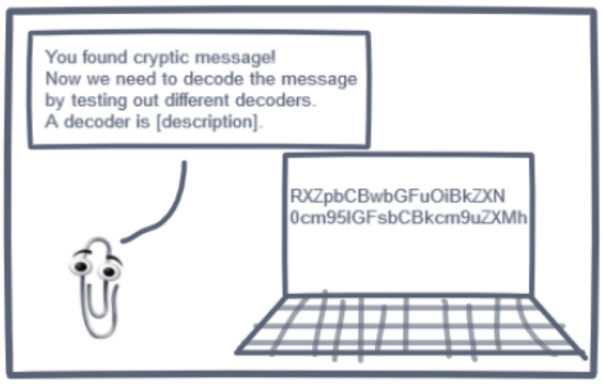

You found cryptic message!
Now we need to decode the message
by testing out different decoders.
A decoder is [description].
RXZpbCBwbGFuOiBkZXN
0cm95IGFsbCBkcm9uZXMh

Figure 3.4: Cyber security specialist scene: encrypted message

3.7 Dialogue design

As seen in the storyboard, the application will contain a lot of conversation between the assistant and the user. Therefore, a decision has to be made on how to realise conversations in VR. The dialogues are indeed very important for the game, because most of the relevant information about the IT professions are given through the conversation with the user. The dialogues should be presented in a natural and interesting way, because the application should arouse interest in IT job roles and the user should not perceive the conversations as lengthy.
In the previous chapter different ways of designing a dialogue system were already mentioned. This section will now determine and evaluate the most useful dialogue system based on the above mentioned objective. Also the decision has to be made under the aspect of simplicity, time and budget.
Spoken dialogue systems may be the most natural way of communication, but on the other hand are not easy to realise and need a lot of computation. This application will therefore stick with a text based dialogue system. To still provide a natural experience to the user, the dialogues should be mixed with interactions. Nevertheless, a voice over for the assistant will be considered.
When it comes to displaying the dialogue text in VR, there are some differences compared to a 2D space text displaying method. In 2D interfaces, for example in computer games, the conversation text of non playable characters (NPCs) is often displayed on the bottom of the screen and fixed to the screen space. This cannot be realised in the same way in VR, because users are not able to focus on the text when the text is fixed to the screen. Therefore, the dialogue texts will be placed into the virtual world space. The challenge of this method is to guide users to the position, where the dialogue is displayed. This will be realised by placing the dialogue panel close to interactable items in the game. Once the user interacts with the items, the dialogue starts. The user is already focusing the objects and wil notice the dialogue appearing in their view. Another method is used when there is a dialogue at the beginning of a new scene: The user's position and view is set close to the dialogue panel initially.

3.8 User interactions

Which interaction methods are chosen for this application? What makes sense? What are the constraints by the hardware? Regarding to the target group and their world knowledge, what concepts can be reused?

Chapter 2 already introduced the reader to the different user interaction methods in VR, which are commonly used. This chapter looks at the different methods and evaluates, which one fits the most for the use case of the prototype VR application. There are several constraints and limitations in user interactions made by the chosen hardware and the budget for the project. The user interaction should also fit to concepts which are already known by the target user group.
The following input methods are available with the Samsung Gear VR: A wireless handheld controller or a touchpad attached to the HMD.
The wireless controller can either be used with the left or right hand. In order to use the controller, it has to be connected with the phone via bluetooth. To register user input there are several buttons/touchpads. The trigger button is a primary button, which can be pressed with the index finger. There is a touchpad which registers the thumb position and has a primary button as well. The last primary button is the back button. The controller registers the hand/arm rotation through sensors. There are the home buttons and the volume buttons, which are reserved by the system.
The touchpad is located on the right side of the HMD and registers touch position and touch gestures. Unlike the handheld controller, the touchpad does not need to be connected manually and always registers user input, when the HMD is used. It is therefore often used as a backup solution, in case no bluetooth controller is connected. However, the application won't be available on the consumer market. It will be primary used during public events and the usage is guided by supervisors. They can ensure to connect a controller before playing the application. Therefore, the user input development will be focused on the usage of the handheld controller.
Looking at the story board there can be several user interactions extracted:

- Relocating in the virtual world
- Selection of objects
- Relocating objects
- Dialogue between assistant and user

First of all, we take a look at relocating in the virtual world, which is walking in VR in other words. Natural walking methods cannot be used because of missing hardware sensors. With the use of the handheld controller, the teleporting method or the camera manipulation method can be used. Since the target group are primary students which are experienced in the usage of digital media, the method of the camera manipulation is chosen. It is expected, that the target group is already introduced to this kind of relocating, since it is widely used in computer games. To reduce the risk of motion sickness, the walking speed is held slow. The user can start walking when pressing the index trigger button. The direction of walking can be changed by moving the head.
The selection of objects will be done with the help of the handheld controller. When using a controller, the ray-casting method is a very straightforward method for selecting. Ray-casting is also used in the default home application from Occulus. It is better to reuse known user interface concepts than to introduce new input methods. Therefore, the ray-casting method is

chosen for the selection of objects in the VR prototype application. The implementation will be similar to the Occulus home application.
Relocating of objects is necessary when completing tasks. Especially during the minigame in the software engineer scene, where the user has to arrange lines of code in order to code the software for the drone. For relocating the lines of code, the drag and drop method is chosen. By pointing and clicking on an object, it starts to stick with the users hand. Clicking again, the current position is applied to the object. This is similar to a drag and drop action on 2D interfaces. Every object which has an interaction, will be highlited when the user points with the controller at it.
The dialogue between user and the assistant is solved via a text based dialogue system, which is used in retro games for example. The user can control the dialogue flow and answer questions through simple button clicks.

4 Implementation

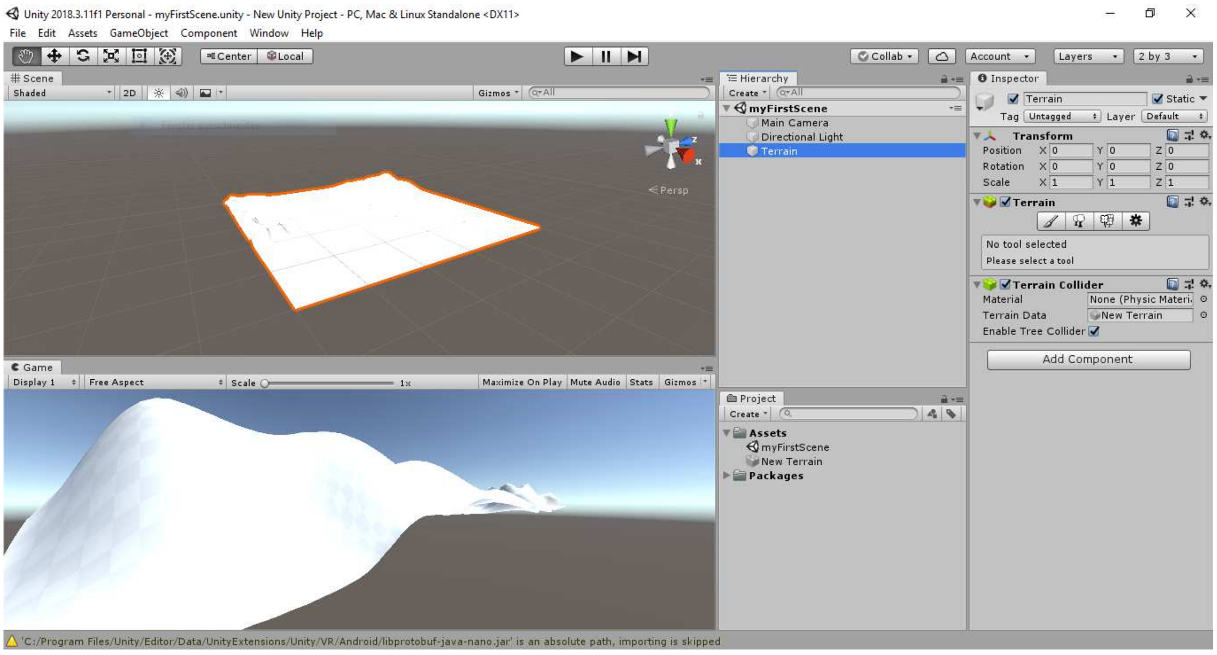
Chapter 3 gave an overview about the contents of the prototype application. The requirements as well as the storyline were set. Also, there was a look at how the provided hardware can be used for user interactions in the application. The chapter mentioned the different stakeholders and their tasks and requirements. This chapter will take a look at the application from the developer stakeholder role. It is a documentation of the development process of the VR application. There will be an introduction to the hardware and software tools from a technical point of view. The software architecture will be described as well as problems which occurred during the development and their solutions.

4.1 Overview about Unity

Unity is a multi platform game engine. A game engine holds a bundle of tools which are necessary for game developments such as tools for graphical design, audio and scripting. Game engines support developing games or applications with a lot of 3D and 3D graphics and also supporting developing VR applications. For the development of this prototype VR application, the game engine Unity and the Occulus Unity SDK is used, which is a support library for Samsung Gear VR and Occulus Go. Unity is the preferred game engine, because it is beginner friendly and free for personal and academic use. Another positive aspect of Unity is the large community from hobby game developers to professional teams. The unity asset store provides many 3D objects, support libraries, textures, audios and other useful tools for game development. This so called assets can be imported into the engine very easily. There is a variety of free assets as well as paid assets. `https://sundaysundae.co/unity-vs-unreal/`
The Unity editor, as seen in figure 4.1 is the main tool, which is used while creating the prototype. It has a variety of views for designing the environment, modifying and testing the game.
In the following, some concepts of Unity will be explained, which are relevant for the implementation process of the prototype [**?**]:

Game objects Game objects in Unity are a very important concept. They can be 3D world objects, lights, cameras and special effects. A game object can also be seen as a container which can contain other game objects. Game objects can be relocated and modified in the unity editor. They can be visible items in the scene or be invisible and work as a trigger area for example.

Components A component in unity defines a specific property and always belongs to a game object. Every game object can have a variety of components attached. Components can change the appearance and behaviour of a game object in the game. There are several build in components available in unity, but through the scripting API it is possible to create highly customisable components. Unity gives an overview of all components of the selected game object. New components can be attached in the editor.



Unity 2018.3.11f1 Personal - myFirstScene.unity - New Unity Project - PC, Mac & Linux Standalone <DX11>

myFirstScene
- Main Camera
- Directional Light
- Terrain

Terrain — Static — Tag Untagged — Layer Default

Transform

Position	X	0	Y	0	Z	0
Rotation	X	0	Y	0	Z	0
Scale	X	1	Y	1	Z	1

Terrain Collider

Material	None (Physic Materi
Terrain Data	New Terrain
Enable Tree Collider	✓

Display 1 — Scale 1x

Assets
- myFirstScene
- New Terrain

Packages

'C:/Program Files/Unity/Editor/Data/UnityExtensions/Unity/VR/Android/libprotobuf-java-nano.jar' is an absolute path, importing is skipped

Figure 4.1: Screenshot of Unity editor

Prefabs Prefabs are reusable game objects with components. Once a game object is copied into the prefabs folder in the project, it can be reused between different scenes and even between different projects.

Materials and shaders Materials can be seen as the skin of game objects. Materials define how the surface of objects look like. The material is a component which can be modified in the Unity editor. Which properties can be edited depends on which shader is selected for the material. Shaders are scripts which define the rendering of the material through algorithms. The shader defines which input parameters can be modified in the material component.

4.1.1 Scripting in Unity

Unity provides a variety of predefined components, which can be attached to game objects, such as a Rigidbody for physical behaviour or an Audio source for playing music. However, for customizing the behaviour of a game object, it is possible to create components and attach a script to it. A script can either be written in C# or a modified JavaScript by Unity. For this prototype application, C# is chosen, because it is the wider used programming language within the Unity community. Every component is a class and inheritates from the MonoBehaviour class. Through the scripting API all existing game objects within the game and their components can be accessed and modified, it is possible to create new game objects, react to events, load new scenes and control the game flow. [**?**]
`MonoBehaviour` is a class from the Unity scripting API, which every game object inheritates from. It contains several methods which can be overwritten to execute code after various lifecycle events. The update method for example is called every frame update of the game object.

It can be used to get continues updated values from the game object for example. The start method is called once before the gameplay starts. This method is mostly used for initialization of other game objects or components. [?]
In general, all rules, best practices and design patterns which are common in the .NET environment can be used within Unity as well. Initialization of variables can be done through assigning values in the code or assigning values in the editor through the component view. The second approach is only possible with object variables which are public. This is why in this project, many variables are public although they are not used outside their own class.

4.1.2 Unity and Samsung Gear VR support

describe registering device

In Unity, VR support can simply be enabled by ticking a box in the player settings. This invokes rendering the graphics in a split screen mode with one sceen for each eye. However, to make a game which is compatible with the Samsung Gear VR or the Occulus Go, more work than that has to be done. First of all, the Occulus support library has to be imported. It can be downloaded from the Unity asset store or the occulus developer website: [?]. This library contains a variety of components and prefabs. For this prototype the texttttOVRCharacterController prefab from the SDK is used. It contains all necessary scripts and objects for the communication between Unity and the VR hardware. The texttttOVRCharacterController is a first person controller which can be used in the VR environment. It controls the camera from a first person perspective. The support library also contains handheld controller support. There is a prefab which contains a script called texttttOVRInput. This script can register connected controllers and input from Samsung Gear VR as well as from Occulus Go.
To make the game playable on the HMD, the above described prefabs have to be dragged into the scene. Additional to the texttttOVRCharacterController, a handheld controller representation has to be added to the scene. This can be done by attaching a controller prefab to the player game object. This prefab displays a controller model in the virtual environment and translates controller rotation into the game.
After finishing this tasks, the prototype can be played in VR. However, user interactions like walking, selection and manipulation are not possible yet. How these interactions are implemented in the game are explained in section .

4.2 Implementation of the smart city scene

According to the storyboard from chapter **??**, the smart city scene is the first scene of the game and the entry point for the different labs, which represent IT job roles. This section describes the implementation of this scene, including the scene design, the gameplay and the deployment to the HMD. It also describes solving the challenges which occurred during the implementation, such as the dialogue system and supporting of different input methods.

4.2.1 Scene Design

Before starting programming of the game logic, the outlook of the city scene has to be designed. This is an important step in the implementation because the look of the environment defines a first impression and has impact when it comes to immersion. The more realistic the environment is designed, the more present users feel in the virtual world.
Skyboxes are a concept in unity to design the background of the game. They contain of a picture wrapped around the scene to give an impression of a wider environment. [**?**] For the prototype, an image with an urban environment is chosen. The image is a 360 photography of a place in a city surrounded by skyscrapers (TODO image). This image gets attached to the skybox component of the main camera. The image is used in all scenes, the world scene as well as the different laboratory scenes, to state out that the user has not left the city.
The smart city environment is an urban environment which should look like a South-East-Asian city, because of the location of NYP. The environment should also encourage free exploration. This means that there should be a high level of details and the game objects of the scenes should be of a high quality. To focus the work on the gameplay and on the same time provide a high quality environmental design, a city scene from the Unity asset store is used TODO ref. This pre designed environment is adapted to fit the requirements of the gameplay. The sample scene of the asset was reduced in size and the buildings were changed to have five outstanding skyscrapers and several less high buildings. Each skyscrapers is a laboratory where the different job roles can be explained. This buildings will become interactive in the further development process and should be recognized easily.

4.2.2 Implementation of the dialogue system

The application contains several conversations with the virtual assistant, as seen in chapter 3. The dialogues between the virtual assistant and the user take place in various places in the game and after different events. The implementation of the dialogue component contains setting the input texts, animating the text appearance during the dialogue, controlling user input and displaying the right dialogue texts at the right time. Especially the last task is a challenge, looking at the fact that by Unity design, the game runs in a stateless game loop. The dialogue component is implemented as a reusable component. It should be possible to add new sentences or change the appearance of the sentences without significant expense. Also the same dialogue class should be used within different scenes. Therefore a reusable dialogue component was designed. The architecture of this component and the dependent classes can be seen in figure 4.2. In the following, it is described how the component works.
The dialogue component is a combination of the dialogue script and the dialogue panel, a game object which contains the text and the user input buttons (see figure TODO). The dialogue script is attached to an empty game object named `DialogManager`. From there, references to the dialogue panel are set. The sentences, which should be displayed by the dialogue system are represented as a string field in the dialogue script. For the world scene, there is one string field for the beginning dialogue, which contains a welcome from the assistant and an explanation of

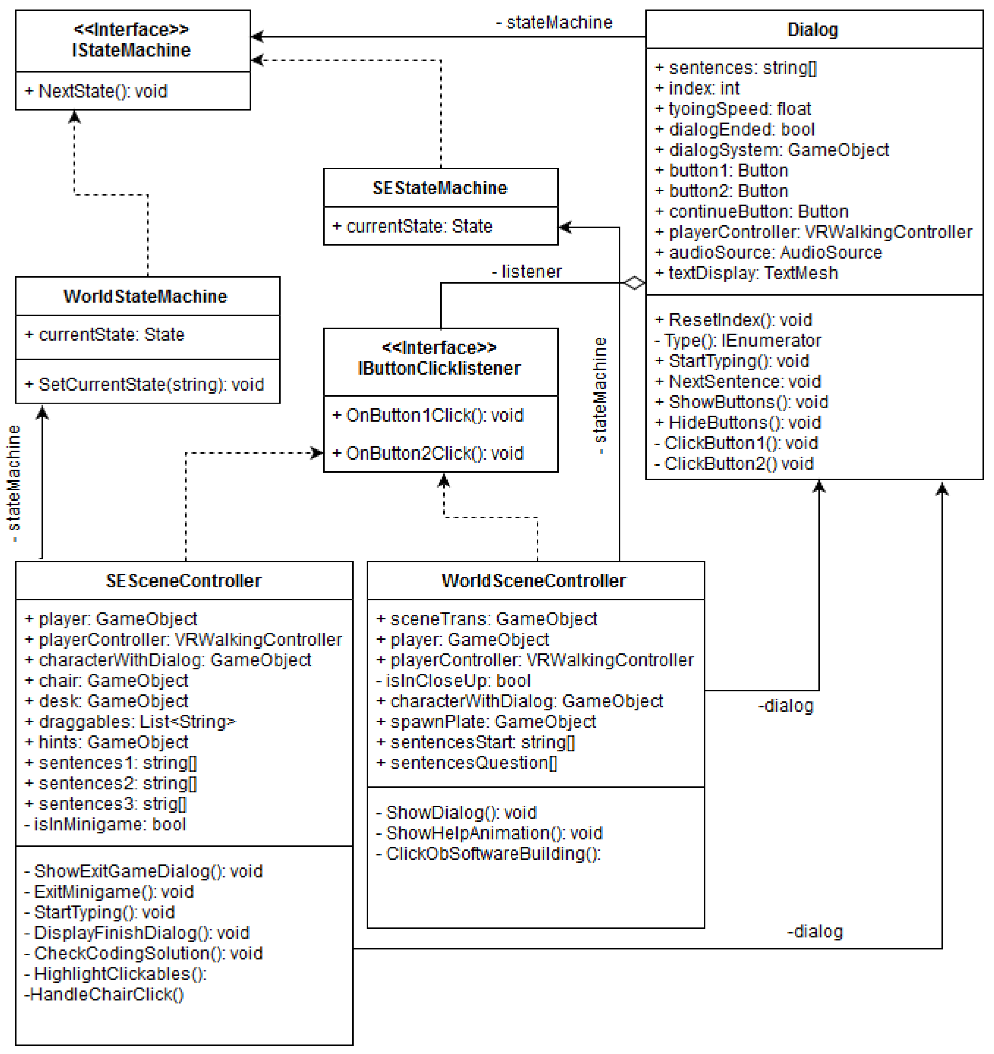


Figure 4.2: UML class diagram for the dialogue component

the controller usage. The remaining string fields contain sentences to explain the different job roles and asking the user to enter a laboratory. Every entry of a field is the text which should be displayed on the panel. When the user clicks the continue button, the next entry from the string field is displayed. This variable has a public accessory. Therefore the content of the dialogue can be changed by any class at any time.

The dialogue script has a public method called `StartTyping` which invokes displaying of the sentences in the string field with a typewriting animation. It also handles the continue button click by the user. Once every sentence is displayed, the dialogue panel disappears automatically. In the scene controller class, it is only necessary to set the content of the sentences and to invoke the `StartTyping` method.

To display questions and react to answers of the users, it is possible to customize the buttons appearing on the dialogue panel. In the world scene this is necessary when asking the user to enter a building. The user can either agree and invoke a scene change or refuse and contiue exploring TODO picture. Every button click invokes a callback, which is an interface method. This interface can be implemented individually by the controller classes. In the world scene,

the "ok" button callback is implemented and loads a new scene.
The appearance of the dialogue panel with the right sentences at the right time can also be managed in the controller classes. In the world scene, the welcome dialogue should be displayed at the beginning of the game, while the explaining dialogues of the different job role are displayed when clicked on the correspoding building. This is done by implementing a state machine pattern. The `DialogManager` class holds a reference of a `stateMachine` variable which is also an interface. The reference is set inside the controller classes. A state machine is an object which implements the `IStateMachine` interface. The method `NextState` performs a step through the state machine. The variable `current` returns the current state, the state machine is at the moment. Every state describes an event or a position in the game. Because the DialogManager is a reusable component, it should not know in which state the game is currently. The task of the DialogManager class is only to display all referred sentences when the method is invoked. After all sentences are displayed, the NextState() method is called, to let the controller classes know, that the displaying of the dialogue is finished.

4.2.3 Deploying to VR headset

An important process in the development flow of the VR prototype is testing. Besides from testing the game in the Unity editor, it should be tested on the device itself. This is necessary, because the look and feel, the input methods and the distances differ from playing the game on desktop and playing it in VR.
There were two different HMDs given for testing the game: Samsung Gear VR and the Oculus Go. They are described in more detail in section 2.4. In the beginning of the development process, only Samsung Gear VR was available. To deploy the game to this device, a compatible mobile phone has to be used. In this project, the Samsung Note5 was used. The first step is to swich on the developer settings on the phone and allow USB debugging. When using the device for testing the first time, the game has to be registered at Oculus developer dashboard. This is a required step when using the Oculus Integration libraries and is done by downloading a file containing a unique key. This file has to be stored in the project. [**?**] After this step, the project in Unity has to be set to Android as a target platform. Unity is a multi platform game engine, and therefore it is not necessary to change the code when switching a platform to deploy on, only a change in the settings is necessary.
The next step is to build the game and transfer it to the phone. For this task, the phone has to be connected via USB to the computer. The building process is invoked through Unity. The game engine builds an executable `apk` file and transfers it to the phone with the help of the Android Debug Bridge (adb). To finally test the application, the apk file is opened on the phone and the phone has to be attached to the HMD.
Later during the development process, Oculus Go headset was provided by the school to test the application. Since this device runs on Android OS, the deploying process is very similar to Samsung Gear VR. Both devices use the Oculus Integration library. Before deploying, developer settings have to be enabled on Oculus Go. This done through a configuration app on any phone connected to the HMD. The device also has to be registered to the Oculus Developer

Dashboard. For building the game, Oculus Go has to be connected with the PC via USB. The game is deployed on the device directly and no phone has to be attached to the HMD. The process of building with Unity works the same way than with Samsung Gear VR.
Comparing both deploying processes, the deployment to Oculus Go was preferred. When testing with Samsung Gear VR the phone had to be detached from the HMD and connected to the computer every time. After the game was built, the phone had to be attached again. This process was time consuming, compared to Oculus Go, where simply a wired connection had to be established for building.

4.2.4 Support of different input methods

Deploying the application to the VR headset takes a few minutes every time. During the development, a continuous testing is necessary. If the testing would always be done on the VR device, it would be too time consuming. Therefore the game is only deployed on the VR headset when some VR specific functions need to be tested. In other cases the game is tested on the desktop.
The problem of this procedure is, that the input methods of both target platform differ: For the VR headset, the handeld controller is used, while on the desktop PC mouse and keyboard are the primary input methods. Therefore a solution has to be found, how to make the game playable on both platforms without changing the code when switching the target platforms.

The Samsung Gear VR provides support for user input in different ways. On the right side of the HMD is a touchpad attached. Common gestures like taping and swiping can be performed on the touchpad. The HMD transfers the input commands directly into the VR application. In the implementation of this prototype, the touchpad is not used as an input method. Instead, the Samsung Gear VR handheld controller is used. Nevertheless, the HMD touchpad should be seen as a fallback solution, in case there is no handheld controller available. Adding a support for the HMD touchpad input method is therefore a feature which should be added in the future.
The Samsung Gear VR handheld controller can either be used with the left or right hand. This controller is connected to the phone via bluetooth. The controller offers more possibilities for user input than the touchpad at the HMD. Figure 4.3 displays the input buttons of the controller in more detail. A very important feature of the handheld controller is also that the controller has sensors to detect its rotation. With this functionality it is possible to point at objects in the game to select them.
When switching from the desktop platform to the mobile VR target platform with the handheld controller, following input methods need to be implemented in a different way:

- Looking around
- Relocating
- Selection of objects

Looking around in the game is done by a component named MouseLook in the desktop target platform version of the game. This component is a script from the unity standard assets, which is a free utilities library available in the asset store. It manages looking around in a first person

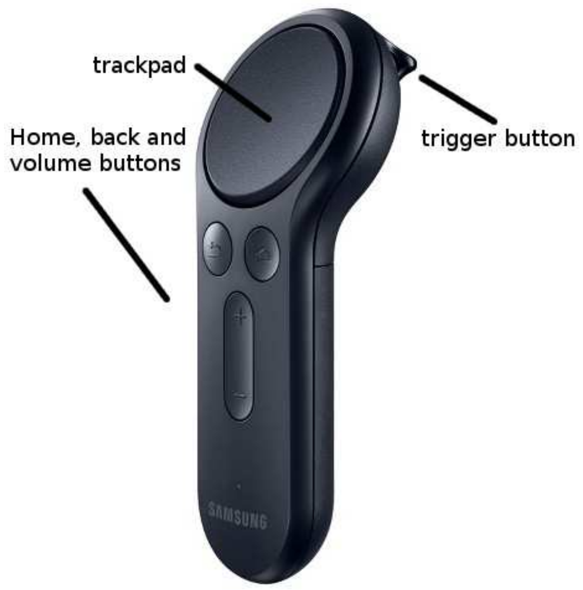


Figure 4.3: Handheld controller with button description. Edited from [?]

view in the direction of the mouse position. When switching the target platform to mobile VR, no mouse position is available, therefore the script does not work. Instead of the MouseLook script, the OVRPlayerController, which was described in chapter 4.1.2 is attached to the first person controller. It manages looking around in the direction of the head movement instead of the mouse movement.
Relocating on the desktop target platform is done with the WASD keys of the keyboard, an input method used in many computer games. The movement is also implemented by a script from the unity standard assets. However, in a virtual environment it is not possible to use the keyboard. Instead, the relocation is done by pressing the trigger button to move forward with a constant speed. The direction of moving is set by the direction in which the user is currently looking. The implementation of this relocating is very straightforward: TODO insert code.

Selection of objects is done with the ray-casting method in both target platforms. How ray-casting in unity works, is seen in example 4.2.4. The ray is created from the mouse position, which is always the center of the screen. The if condition checks whether the ray hit a game object and returns this object inside the `hit` variable.

```
RaycastHit hit;
      Ray ray = camera.ScreenPointToRay(Input.mousePosition);

      if (Physics.Raycast(ray, out hit)) {
         Transform objectHit = hit.transform;
      }
```

Listing 4.1: simple ray-casting

First of all a ray is created from a fixed starting point. If the ray hit an object, this object is stored

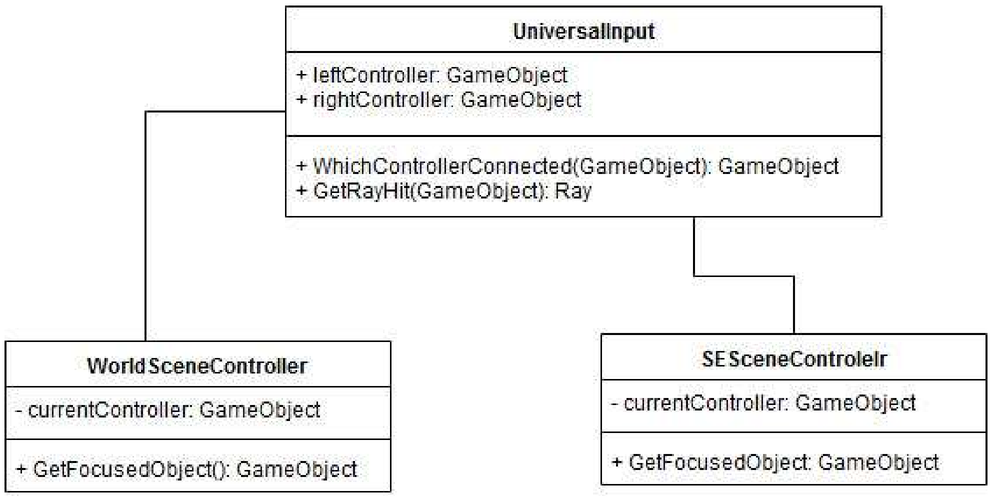

UniversalInput
+ leftController: GameObject
+ rightController: GameObject
+ WhichControllerConnected(GameObject): GameObject
+ GetRayHit(GameObject): Ray
WorldSceneController
- currentController: GameObject
+ GetFocusedObject(): GameObject
SESceneControlelr
- currentController: GameObject
+ GetFocusedObject: GameObject

Figure 4.4: UML class diagram for managing different target platforms

as a Hit in the variable hit. The difference of ray-casting on the desktop and the VR version is that the ray comes from the center of the main camera on the desktop version, whereas in the VR version the ray is initialized from the controller model in the game. In order to make the game work on the Samsung Gear VR, the initialization of the ray has to be changed to the example in listing 4.1

```
RaycastHit hit;
      if (Physics.Raycast(controller.transform.position,
       controller.transform.forward, out hit))
      {
         return hit.collider.gameObject;
      }
```

Listing 4.2: Ray initialized from the controller representation in the game

The method `Physics.Raycast` creates the ray from the controller's position instead of the center screen this time. The method checks for objects hit by the ray at the same time. When changing every ray creation which is done in the game's code to the above showed solution, the game is working on the mobile VR platform but it is not working on the desktop target platform or in the unity editor gameplay mode anymore. In order to use ray-casting on every target platform without changing the code every time, an additional class has to be added. This class is able to detect, which is the current target platform and also if the game is running in the editor gameplay mode. The class creates the ray from either the main camera or the current connected controller depending on the current target platform. It is also able to detect, if a controller is connected and if the controller is set as left or right hand used. In the UML-diagram in figure 4.4, the UniversalInput class fulfills the above explained functionalities. The UML-diagram shown below only shows the method and fields relevant for the problem case of this chapter. Subclasses which are used by the UniversalInput class are not displayed in the diagram. The controller classes of the different game scenes use the methods of the UniversalInput class to get the selected object from the ray. It is now possible to play the application with the Samsung Gear VR and use the controller to point at selectable objects. At the same time, the game can be played on a desktop with selecting objects through the main camera focus.

4.3 Implementation of the software engineer scene

The software engineer scene is a laboratory which can be entered through the world scene. The scene contains a mini game through which users can experience programming themselves. The laboratory design is held in a futuristic look with a lot of glass materials and a high amount of white and grey colors. Robots and drones as decoration element and futuristic furniture assets underline the futuristic look of the laboratory. One of the main challenges of this scene was to develop the coding mini game. The objective of the game is to write the code for a drone, which should be able to pickup a package. Another challenge, which occurred in all scenes of the game, but particularly in the software engineer scene was the user guidance. This section describes which task were done in this scene.

4.3.1 Coding mini game

As defined in the storyboard, the mini game in the software engineer scene is about arranging five different code blocks in a logical order. The code blocks contain method calls and a "if" control sequence. In order to start the mini game, the user has to sit on the work desk in the scene.
The screenshot in figure **??** describes the view of the mini game after sitting down on the chair. The code plates seen in the picture can be selected and dragged into the laptop screen. There they appear as code text on the screen. The buttons above the laptop are to exit the game, return the last step, enable or disable user input hints and to compile the game.
Every code plate in the minigame has a component called `Draggable` attached to the game object. Through this component, objects follow the main camera direction or the controller direction when clicked on it. For releasing objects, users have to click again. Every time, a code plate collides with the screen object of the laptop, the plate disappears and the text on the plates appears on the screen. This is managed through the `Programming` class which is responsible for all visible changes in the mini game. Every time users drag a code plate into the screen, the name of the game object is added to the static string list variable `currentOrder`. This variable is stored inside the static class `ProgrammingLogic`. The `ProgrammingLogic` class also contains a list called `correctOrder`, which represents the right order of the code plate arrangement and a method `CheckSolution()`, which checks the both lists for equality. If the user solves the mini game correctly, the controller class of the scene invokes an animation of a drone flying around in the room and the user is asked to exit the laboratory.

4.3.2 User guidance

TODO: Maybe describe hint label also As defined in the requirement in chapter 3, the game should encourage free exploration. On the same time, the user should always know where to go next and be able to detect interactive objects. In the software engineer scene, the desk, which the user has to click on to start the mini game, changes its texture when pointed a it. Every game object which should appear as interactive, needs to have a tag "Clickable", which can be assigned to the editor. Inside the update method of the controller class, it is checked, if the user points at an object with this tag. Then the shader of this object is changed to a shader called `outlined`, which makes the object appear like in figure **??**. In example 4.3.2, the current highlighted object is called `hit`. A reference of this object is stored to the object variable `focusedObject` in order to remove the shader once the focus changes.

```
if (hit.tag == "Clickable" || hit.tag == "Draggable")
        {
           var renderer = hit.GetComponent<Renderer>();
           if (renderer != null)
              hit.GetComponent<Renderer>().material.shader = outlineShader;
               focusedObject = hit;
           }
        }
```

When the focus changes to another object, the standard shader is applied to the previous highlighted object.
The above code example can be added to other controller classes as well, to highlight interactive objects in other lab scenes. In the world scene a similar method is used for indicating the buildings which can be entered. Instead changing the shaders, the color was changed and a blinking animation was added, to attract the user's attention even more.

5 Testing

The previous chapter described the implementation process of the prototype VR application. In this chapter, it is examined, whether the application fulfils the requirement of helping prospective students in choosing their profession by visualizing IT job roles. As outlined in the introduction, this is the main objective of the project. In general, it is difficult to find an objective method to prove or disapprove this hypothesis. This is because VR creates an unique user experience which is subjective and varies with every user. In this project, several test persons played the VR application and answered questions about their subjective experience. This chapter will explain the testing procedure and present the results of the tests.

5.1 Test design

At the beginning of the testing process, the software engineer scene was completed. The world scene contained the introduction to the game and the entrance point to the software engineer scene. The focus of the testing outcome was therefore, how the idea of the software engineer job role has changed for the test persons through playing the VR application. In order to create measurable data, the test is designed as followed:

Before playing prototype At the beginning, participants answer some basic questions about their age, gender, which school they are and their pre experience in VR. After this they are asked four questions about their subjective understanding of the software engineer job role. Than they get a short introduction on how to use the controller in the game and the VR headset is fitted for them, so that they feel comfortable.

Playing prototype Participants get a short introduction on how to use the controller and than they are told to explore the virtual environment and solve tasks in their game on their own. However they are allowed to ask questions if they stuck in the game or feel disorientated. The participants play one full walkthrough of the prototype which includes exploration of the world scene, clicking on the interactive building and entering the software engineering laboratory. In the software engineer scene they find the desk and solve the mini game. After they solved the mini game correctly they can watch the drone animation and then they are told to exit the game and put off the VR headset by the virtual assistant.

After playing prototype

5.2 Testing group

5.3 Questionnaire

5.4 Testing environment

5.5 Results

6 Conclusion

6.1 Evaluation of test results

6.2 Summary

6.3 Outlook

Bibliography

[Arn17] ARNOWITZ, Ethan: *World Builder VR*. http://ethanarnowitz.com/html/projects/project_WB.html. Version: 2017

[CH14] CAKMAK, Tuncay ; HAGER, Holger: *Cyberith Virtualizer - A locomotion device for Virtual Reality*. 2014

[DBGJ13] DÖRNER, Ralf ; BROLL, Wolfgang ; GRIMM, Paul F. ; JUNG, Bernhard: *Virtual und Augmented Reality (VR/AR): Grundlagen und Methoden der Virtuellen und Augmentierten Realität*. Berlin and Heidelberg : Springer Vieweg, 2013 (eXamen.press). http://dx.doi.org/10.1007/978-3-642-28903-3. http://dx.doi.org/10.1007/978-3-642-28903-3. – ISBN 978–3–642–28902–6

[Fen13] FENLON, Wesley: *The Promise and Challenges of Head-Mounted Virtual Reality Displays*. https://www.tested.com/tech/gaming/454559-valves-michael-abrash-promise-and-challenges-vr/. Version: 2013

[Fuc11] FUCHS, Philippe: *Virtual reality: concepts and technologies*. Boca Raton, Fl : CRC Press, 2011 (A Balkema book). – ISBN 9780203802953

[H$^+$17] HEUBLEIN, Ulrich u. a. ; DEUTSCHES ZENTRUM FÜR HOCHSCHUL- UND WISSENSCHAFTSFORSCHUNG (Hrsg.): *Zwischen Studienerwartungen und Studienwirklichkeit: Ursachen des Studienabbruchs, beruflicher Verbleib der Studienabbrecherinnen und Studienabbrecher und Entwicklung der Studienabbruchquote an deutschen Hochschulen*. Hannover, 2017. – Available online: https://www.dzhw.eu/pdf/pub_fh/fh-201701.pdf; last accessed 5/5/2019

[Jer16] JERALD, Jason: *ACM Books*. Bd. 8: *The VR Book: Human-centered design for virtual reality*. First edition. New York, NY and San Rafael : Association for Computing Machinery and Morgan & Claypool Publishers, 2016. http://dx.doi.org/10.1145/2792790. http://dx.doi.org/10.1145/2792790. – ISBN 978–1–97000–113–6

[Kea18] KEANE, Megan: *Virtual Reality Storytelling – Is it Possible?* https://theblog.adobe.com/virtual-reality-storytelling-possible/. Version: 2018

[Lin15] LINOWES, Jonathan: *Unity virtual reality projects: explore the world of virtual reality // Explore the world of virtual reality by building immersive and fun VR projects using Unity 3D*. Online-Ausg. Birmingham, UK : Packt Publishing, 2015 (Community experience distilled). – ISBN 9781785286803

[Lou18] LOUDEN, Gregory ; VENTUREBEAT (Hrsg.): *'Play, don't show': How VR storytelling compares to movies*. https://venturebeat.com/2018/05/21/play-dont-show-how-vr-storytelling-compares-to-movies/. Version: 2018

[Lov18] LOVREGLIO, Ruggiero: A Review of Augmented Reality Applications for Building Evacuation, 2018

[LS18] LANGBEHN, Eike ; STEINICKE, Frank: *Redirected Walking in Virtual Reality*. Springer Encyclopedia of Computer Graphics and Games, 2018. – 1 – 11 S. http://basilic.informatik.uni-hamburg.de/Publications/2018/LS18

[Min18] MINISTRY OF EDUCATION SINGAPORE: *Education System*. Website, 2018. – Available online: https://www.moe.gov.sg/education/education-systeml; last accessed 18/4/2019

[Nan18] NANYANG POLYTECHNIC: *Full Time Diploma Courses*. Website, 2018. – Available online: https://www.nyp.edu.sg/schools/sit.html; last accessed 22/4/2019

[Occ] OCCULUS: *Native Development Overview*. https://developer.oculus.com/documentation/mobilesdk/latest/concepts/book-native/

[Ogd19] OGDON, Dorothy Carol: Hololens and Vive Pro: Virtual reality headsets. In: *Journal of the Medical Library Association* Bd. 107 January 2019. 2019, S. 118–121

[oth14] OTHREE: *Google Cardboard*. https://commons.wikimedia.org/w/index.php?curid=40703922. Version: 2014. – License: CC BY 2.0

[Pes14] PESCE, Maurizio: *Samsung Gear VR*. https://commons.wikimedia.org/w/index.php?curid=37143658. Version: 2014. – License: CC BY 2.0

[Pes15] PESCE, Maurizio: *HTC Vive*. https://commons.wikimedia.org/w/index.php?curid=39894434. Version: 2015. – License: CC BY 2.0

[Sam19] SAMSUNG: *Gear VR Specifications*. https://www.samsung.com/global/galaxy/gear-vr/specs/. Version: 2019

[SBW17] SIESS, Andreas ; BEUCK, Sandra ; WÖLFEL, Matthias: Virtual Reality – Quo Vadis? How to Address the Complete Audience of an Emerging Technology (Full Paper), 2017, S. 315–323

[SC19] SHERMAN, William ; CRAIG, Alan: *Understanding Virtual Reality: Interface, Application and Design*. 2. Morgan Kuafmann, 2019. – ISBN 1558603530

[Sin18] SINGAPORE EXAMINATIONS AND ASSESSMENT BOARD: *About GCE O-Level*. Website, 2018. – Available online: https://www.seab.gov.sg/home/examinations/gce-o-level/about-gce-o-level; last accessed 18/4/2019

[SW97] SLATER, Mel ; WILBUR, Sylvia: A Framework for Immersive Virtual Environments (FIVE): Speculations on the Role of Presence in Virtual Environments. In: *Presence: Teleoperators and Virtual Environments* Bd. 6(6). 1997, S. 603–616

[Tha18] THAM, Jason et. a.: Understanding Virtual Reality: Presence, Embodiment, and Professional Practice. In: *IEEE Transactions On Professional Communications* Bd. Vol. 61, No. 2, June 2018. 2018, S. 178–195