Hochschule Karlsruhe Technik und Wirtschaft
Fakultät für Informatik und Wirtschaftsinformatik

von

Betreuer:

September 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, September 26, 2019

Zusammenfassung

Diese Arbeit beschäftigt sich mit der Hypothese, ob virtuelle Realität (VR) zukünftigen Studierenden IT Berufe näher bringen kann. Dazu wird eine prototypische Anwendung entwickelt, welche eine Simulation von verschiedenen IT Berufen beinhaltet. Die Berufe sollen spielerisch erkundet werden und die Inhalte in einer Geschichte verpackt sein. In der Arbeit wurde zunächst ein Entwurf der Anwendung gemacht und ein Storyboard entwickelt. Im nächsten Schritt wurde der Beruf des Software Engineer ausgesucht und dessen Berufsbild in der VR Anwendung implementiert. Um den Nutzen der Anwendung und die These zu überprüfen wurde eine Studie durchgeführt. Wir testeten dabei ob sich die Teilnehmerinnen und Teilnehmer nach der Benutzung der VR Anwendung eher vorstellen können als Software Engineer zu arbeiten als vor der Benutzung. Die Ergebnisse der Studie waren positiv. Die Teilnehmerinnen und Teilnehmer zeigten mehr Interesse am Beruf des Software Engineers und fühlten sich besser informiert über das Berufsbild. Darüber hinaus konnte sich ein Großteil der Teilnehmerinnen und Teilnehmer vorstellen, als Software Engineer zu arbeiten.

Abstract

In this work we examine whether virtual reality (VR) helps prospective students in experiencing and learning about IT professions. Therefore, we develop a prototype VR game which simulates IT job roles. Users can explore the various jobs in the game and solve minigames connected to the IT professions. The information of the game is wrapped in a main story. Before beginning with implementing the game we set the requirements and design a storyboard. Then we pick the job role of the software engineer and implement a prototype VR application which contains

Contents

List of Figures iii

1 Introduction and goals 1
1.1 Motivation . . . 1
1.2 Goals . . . 1
1.3 Approach . . . 2
1.4 Hypotheses about advantages of VR for prospective students . . . 2
1.5 Overview of the thesis . . . 2

2 Background studies 4
2.1 Context . . . 4
2.1.1 Information technology courses at Nanyang Polytechnic . . . 4
2.1.2 What is virtual reality? . . . 6
2.2 Virtual reality basic concepts . . . 7
2.2.1 Different dimensions of reality . . . 8
2.2.2 Immersion and storytelling . . . 9
2.2.3 Interaction: Selection, manipulation and navigation . . . 10
2.3 Hardware for VR . . . 14
2.4 Showcases of VR applications . . . 15
2.5 Limits of VR . . . 16
2.5.1 Motion sickness . . . 16
2.5.2 Hardware . . . 17
2.6 Measurement of effectiveness of VR technologies . . . 18

3 Design 19
3.1 Process model . . . 19
3.2 Requirements . . . 20
3.3 Stakeholders . . . 21
3.4 Target group . . . 22
3.5 Main story concept . . . 23
3.6 Storyboard . . . 24
3.6.1 World scene and software engineer scene . . . 24
3.6.2 Cyber security specialist scene . . . 25
3.7 Dialogue design . . . 26
3.8 User interactions . . . 27

4 Implementation **30**
4.1 Overview about Unity . . . 30
4.1.1 Scripting in Unity . . . 31
4.1.2 Unity and Samsung Gear VR support . . . 32
4.2 Implementation of the smart city scene . . . 33
4.2.1 Scene Design . . . 33
4.2.2 Implementation of the dialogue system . . . 35
4.2.3 Deploying to VR headset . . . 38
4.2.4 Support of different input methods . . . 39
4.3 Implementation of the software engineer scene . . . 42
4.3.1 Coding minigame . . . 43
4.3.2 User guidance . . . 45

5 Testing **47**
5.1 Testing group and environment . . . 47
5.2 Test design . . . 47
5.3 Questionnaire . . . 48
5.4 Results . . . 49
5.4.1 Effectiveness of the VR prototype . . . 49
5.4.2 Discussion . . . 50
5.4.3 Immersion and user guidance . . . 50
5.4.4 Discussion . . . 52

6 Conclusion **53**
6.1 Summary . . . 53
6.2 Outlook . . . 53

References **55**

List of Figures

2.1 Head mounted display by [Pes14] 6
2.2 Graphical representation of the dimensions of reality by [**?**] 8
2.3 Selection of targets through ray-casting in the Oculus sample scenes [Gol15] . . 11
2.4 Mini map as part of the WIM technique. By [Arn17] 12
2.5 Redirected walking in VR, virtual walking path vs. actual walking path. By [LS18] 13
2.6 Oculus Go headset (left) by [Syc18] and Samsung Gear VR headset by [Pes14] 14

3.1 Board for visualising tasks and workflow. 20
3.2 Smart city scene from where the user can explore different IT job roles (own illustration) 24
3.3 Software engineer scene: dialog and minigame (own illustration) 25
3.4 Cyber security specialist scene: searching log entries (own illustration) 25
3.5 Cyber security specialist scene: encrypted message 26
3.6 Screen space dialogue panel by [scr11] vs. world space dialogue panel. 27
3.7 Samsung Gear VR handheld controller [Sam19a] 28

4.1 Screenshot of Unity editor (own image) 31
4.2 Screenshot of smart city environment (own image) 33
4.3 Software Engineering laboratory entry point (own image) 34
4.4 UML class diagram for the dialogue component (own image) 35
4.5 Example of a dialogue, asking to enter a building (own image) 36
4.6 State machine diagram of the dialog flow in the world scene (own image) . . . 37
4.7 Handheld controller with button description. Edited from [Sam19a] 39
4.8 UML class diagram for managing different target platforms (own image) 40
4.9 Screenshot of the coding minigame (own image) 43
4.10 Highlighting of interactive objects with additional task description 45

5.1 Histogram of the answers from statement 1 50
5.2 Histogram of the answers from statement 2 51
5.3 Histogram of the answers from statement 3 51

1 Introduction and goals

The purpose of this thesis is to develop a virtual reality (VR) application for students and prospective students to experience different professions in the IT industry. The decision for a specialization of a higher education can be very challenging to students because it has a great impact on their future. Schools and universities struggle to provide comprehensive information about courses and their related job roles. This thesis investigates how the usage of VR can support prospective students by allowing them to experience different professions in the IT industry and giving them an overview of courses that are required to obtain a higher-education degree.

1.1 Motivation

Missing knowledge about future professions can result in high dropout rates, as seen in [H+17]: In total, about one third (32%) left German universities without a degree in 2016. Among the dropouts, the most common reasons were high requirements and difficult exams (30%). Other reasons were a lack of motivation (17%) and not enough practical experience during the studies (15%). Students experience a lack of perseverance in their studies, because of missing imagination and information about their future profession. Therefore, motivation lacks occur. It is harder for students to master theoretical parts of their studies. Furthermore, the study shows that researching about the programme before applying for a course decreases dropout rates.
In our opinion, many reasons for a university dropout can be avoided. Therefore, the question rises as to how we can provide suitable information about IT professions to prospective students. Moreover, how is it possible to engage students in getting to know their future job roles?

1.2 Goals

There are many ways to learn about IT courses and job roles. For example, it is possible to read articles, watch a video or ask experienced persons. These approaches provide information about IT job roles in a passive way. Prospective students should actively be involved in their future professions and experience the responsibilities and challenges of the job roles themselves. The media described above are not able to give a first hand impression of a job role.
In VR, users can experience a computer generated environment from a fist person perspective. They wear a special headset which fully shields them from the real surroundings. It is possible to get a real and immerse experience of the virtual environment. Users feel like they are actually present in the virtual world. [Lin15]
With the usage of VR, it is possible to provide a visualization and simulation of future professions to prospective students. Thus, 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 stu-

dents with getting a better experience for different professions. Universities can bring 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 higher-education degree at a polytechnic.
The approach is to develop a VR prototype, that introduces and simulates several IT job roles. This application should provide information about responsibilities and daily tasks in an IT job. We pursue a learning-by-doing approach by letting users complete tasks or minigames. The success of the VR prototype depends upon the design, the storyline and the interactivity, not only on the usage of VR itself.
To measure whether the application helps students in improving their understanding of IT job roles, we also test the application. We develop a questionnaire to see, how the understanding of IT job roles has changed. The two comparing states are before and after playing the VR application.
The goal of this work is not to implement a complete ready-to-use application, that contains all job roles of the IT industry. Instead, the prototype should contain the simulation of at least one IT job role. The experiences made during the implementation process and the results of the user tests build the result of this work and the base for further implementations.

1.4 Hypotheses about advantages of VR for prospective students

VR can create virtual environments in which users experience a stronger feeling of presence compared to other media. Therefore, it is expected that VR helps prospective students to dive into job roles and provide a realistic overview about IT professions. We expect that students will show more interests in IT job roles after playing the VR application than they do before playing.
VR is currently a trending topic in IT. Students are able to use technologies, which they can later work with. The idea is to grow interest in the technologies behind VR and encourage students in developing their own application. Compared to an internship, prospective students can profit from a cheaper and less time consuming approach in getting to know a job role.

1.5 Overview of the thesis

In chapter 2 we give an overview about basic VR technologies and current developments. Common problems, that occur when designing a VR application and their solutions are discussed. Following, in chapter 3, we design a storyline for the application and evaluate suitable best practices in hardware and software technologies. The next step is the implementation of the prototype VR application in chapter 4. This application should provide information about different IT job roles related to the IT diploma courses at Nanyang Polytechnic. In the following, we design and perform a user study to test the user experience as well as the main hypothesis. The focus is set on how participants' expectations on IT job roles change after playing the VR application. Chapter 5 sums up the test procedure and results. Finally, we evaluate the whole project under the aspect of fulfilling the purpose of helping prospective students in their understanding of IT job roles in chapter 6.

2 Background studies

Before introducing the design and development of the prototype application, a fundamental base of knowledge has to be set. In this chapter, we give 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, we specify the most important technologies for the prototype and also point out where the limits of VR are. Finally, a measurement for the effectiveness of VR is presented.

2.1 Context

The prototype application focuses on the information technology diploma courses offered at the Nanyang Polytechnic. In the following, the environment of the school and the courses are 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. This examination is mainly for students who have visited a secondary school. [Sin18] Students are allowed to take courses either on a junior college, a technical institute or a polytechnic after passing the O-Level examination. In Singapore, a polytechnic offers students a more practical and industry based education than a junior college, which provides more widespreaded education. In general, students study three years at a polytechnic until they get their diploma. After getting a diploma, students can continue their education at a university or start to work in the industry. [Min18]

2.1.1 Information technology courses at Nanyang Polytechnic

NYP offers a wide range of courses in different fields. One of the fields is information technology (IT). In total, there are five courses. 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 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 is for specialization. The students can choose between elective courses in the fields of artificial intelligence, enterprise cloud computing, geospatial and mobile innovation and cybersecurity. This course as well as all other IT courses at NYP offer one practical industry project and an internship in the third year.

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 deepens the knowledge of network engineering, programming, and IT service management. There is 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 bewaring systems and data of 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 also complete an applications security and an infosecurity project. The last year offers specialization in the topics cybersecurity and cyber forensics. It is possible to choose cross-disciplinary 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 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 at a big data and business analytics project. During the third year, students complete a data science project and 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. It is focused on connecting business and financial topics with information technology from the beginning. 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.

2.1.2 What is virtual reality?

According to [Fuc11] VR is a simulated virtual world, created with hardware and software. It provides a real time user interaction. Users feel the maximum amount of immersion when experiencing the computer generated world. The virtual world is mostly displayed through 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 a three dimensional picture by the users brain.

Figure 2.1: Head mounted display by [Pes14]

The computation of the virtual world can either be done with a standalone desktop PC or with the HMD only, for example through a mobile phone which is wrapped inside the HMD.
Different than 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 environment. Thereby, the borders between reality and simulation become blurry. [DBGJ13]
AR is a closely related technology to VR, but this thesis will focus on VR applications only. In section 2.2.1 we will differentiate VR and AR in more detail.
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 promising to introduce VR in the gaming branche. Lately, non-gaming VR applications are getting 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. For this work, the usage of VR in education is very important.

2.2 Virtual reality basic concepts

[SC19] describe five core concepts for VR that are the key elements when it comes to design a virtual reality. In the following, these concepts are 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. Therefore every experience is a unique experience due to the cultural background of the participants, their age, their expectations and their knowledge in technology. Therefore, it is a challenge for developers of VR applications to create a virtual world that 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 we will give 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. The experience of a participant is influenced by the design of the virtual world space.

Immersion Immersion in general describes how deeply users connect with a virtual world. Users experience something through a medium that they would not be able to experience without it. They see actions and things from a different point of view and can connect with it in some way. The deeper users can connect to the virtual environment and the objects inside it, the more immerse is the VR experience. In chapter 2.2.2 the term immersion will be explained in more detail.

Interaction Most commonly, 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 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.

2.2.1 Different dimensions of reality

Besides VR there are other technologies which serve the purpose of creating virtual worlds for the user. They differ in the extend of intervention with 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 [?]

The reality describes the real world and uses non digital devices for interaction, such as a steering wheel for driving a car. 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 [1], in which players search, based on a phone's camera and location services, in the real environment for computer generated items. 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 is 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 "mixed reality" which describes the interaction between real life elements and the virtual environment.

[1] for further information see: `https://www.pokemongo.com/en-gb/`

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.

2.2.2 Immersion and storytelling

The term immersion is already introduced shortly in section 2.2. It is an important measurement for this work. An immerse experience for this VR application means that students can dive into the IT job roles and get a real insight from a first person's view. It is expected that the higher the grade of immersion is, the better participants can understand the IT job roles.
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 a 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, similar to their behaviour in a real environment. Through this, their task performance increases. This is a reason why presence is an important factor when it comes to developing VR applications, especially in the field of education and learning. [SW97]

Now that the terms immersion and presence are explained, the question is now, how to design the story of a VR application in order to gain a maximum 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 [Tha18]. Since the access to hardware tools is limited by the school in this project, we rely on a good storytelling to increase the immersion of the VR prototype.
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 with the story. When it comes to storytelling in VR, the following things should be considered:
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. [Kea18]

2.2.3 Interaction: Selection, manipulation and navigation

Users of a VR application need to interact with the environment in order to increase immersion. Several challenges occur when dealing with interaction in VR. 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. In this chapter we describe some typical interactions and their common solutions in VR. It is a basic overview of interaction methods, which are considered to be used in the prototype VR application. The final method, which suits the best for the prototype, is 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 (colour, 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. [DBGJ13] describe some of the most common interaction techniques.

Figure 2.3: Selection of targets through ray-casting in the Oculus sample scenes [Gol15]

For this section, we explain the techniques, which we most likely consider to use in the prototype of this work: gaze selection, ray-casting, Go-Go, HOMER and world in miniature.

A very straightforward method in selection could be the **gaze selection**. 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.

Ray-casting works with a virtual ray coming from the users' hand, which can be seen in figure 2.3. Therefore, a 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 meets with the expectations of users. Nevertheless, this method becomes inaccurate when dealing 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 selectable objects are out of range. When the virtual arm is lengthened, the user's hand movements are mapped into a smaller range than if the virtual arm is in a normal position. This mapping is not linear and overcomes the problem of inaccuracy when selecting objects too far away. The technique can be used for relocating objects, because the user does not have to walk on the same time. However, only objects which are in viewing range can be selected or ma-

nipulated with this technique. Some objects could be hidden by other obstacles and cannot be selected. Also, objects in far distance appear very small, which makes manipulation difficult.
The **HOMER technique** deals with manipulation of objects in the distance. The abbreviations stands for Hand-Centered Object Manipulation Ray-casting. As the names says, it uses techniques of ray casting for selection. To make a manipulation easier, the object moves to the users hand after selection. The user can now perform their desired manipulation. After the manipulation is finished, the objects teleport back to their 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 2.4: Mini map as part of the WIM technique. By [Arn17]

2.4, the user gets a miniature map of the virtual world with all selectable objects. The user can now select desired objects and see the selection in miniature map. 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, there is a conflict between limited space in the real world and the unlimited virtual world [SC19]. It is possible to collide with real world objects while wearing a HMD. Therefore, we describe some techniques by [DBGJ13] on 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. For example, the moves the camera with the help of a joystick. Another method which is very common in 3D computer games and therefore matches with the expectations of users, is 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 of possible motion sickness. This phenomenon will be described in chapter 2.5.

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. [B+16]

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 2.5

Figure 2.5: Redirected walking in VR, virtual walking path vs. actual walking path. By [LS18]

2.3 Hardware for 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 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. [Vit]
Current HMDs on the market can be divided into three categories: Low cost consumer HMDs, High End consumer HMDs for frequent VR players and high standard VR headsets, targeting customers in the industry. Looking at the new VR headset which are announced for 2019, the low cost VR headsets are improved in weight and screen resolution. Many of the new devices are mobile or standalone products, like the Oculus Quest. The high end VR headsets for private and industry customers introduce also the technique of eye-tracking, like the HTC Vive Pro Eye, and new display techniques. In general, there is a more variety of VR headset products and manufacturers in the year 2019 in comparison to the previous years. [Per19]
For this project, the game will be deployed on two different devices which are from the low cost category. The devices provided for development are the Samsung Gear VR headset and the Oculus Go headset. In the following, we compare the two devices and point out their advantages and disadvantages. Both devices can be seen in figure 2.6.

Figure 2.6: Oculus Go headset (left) by [Syc18] and Samsung Gear VR headset by [Pes14]

Samsung Gear VR Samsung Gear VR is a mobile based VR headset which was developed by Samsung and Oculus and 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 recent Samsung phones including Galaxy S6 and higher, Galaxy Note 5 and higher, Galaxy A8 and higher. It comes with a bluetooth handheld 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 inside the headset to detect if the user is currently wearing the HMD. The device does not support motion controlling, like walking. [Sam19b]
Gear VR 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.

Oculus Go This device was announced in May 2018 and is a standalone device. Inside the HMD is 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 much of a difference between the two devices. Nevertheless, an advantage of the Oculus Go is that no phone has to be attached to the device. Also the head strap and the audio output were improved.[Rob15]

2.4 Showcases of VR applications

There are a variety of VR applications available for very different use cases and made with different concepts. In this section we present the game *Labster* which is an educational learning application and was ported to VR.
The objective of Labster is to teach university students new topics in biology and chemistry by self exploring and self-teaching instead of frontal teaching. This objective is very similar to the goals of this work where we aim to let students explore IT job roles instead of getting only passive information. Labster has multiple scenarios which contain an interesting storyline. The stories are real life problems which can be solved by using the introduced knowledge about chemistry and biology, such as solving a murder case. The scenes take mostly place in 3D laboratory where users can choose to work with different instruments and learn how to use them. A study from the founder of the product, , found out that students show better results in tests when they learned the topics with Labster instead of front teaching. [Bod15]
We could not be able to test the game, but from watching walkthrough videos and reviews, we tried to reconstruct, how Labster creates an immerse experience and is educational at the same time. The first thing, which Labster focus on is a good storyline: It connects the theoretical

knowledge with a real life scenario and in the same time introduces job fields in biology and chemistry. The students learn that they can do something meaningful with their knowledge which motivates them and let them immerse deeper with the game. Secondly, Labster guides users by having dialogues with an assistant and showing textual hints when hovering over interactive items. This helps that users do not feel disoriented and can focus on solving the tasks instead of figuring out how user interactions work.
The game Labster can be seen as a good example and role model for the prototype application. In chapter 3 we will discuss what concepts seen in Labster can be useful for the VR prototype in this work.

2.5 Limits of VR

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. 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.

2.5.1 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 ten 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. 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 ten minutes durance and trying to avoid fast moving objects in their application. [**?**]
This approach however, limits the possibilities in VR games. 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 [SJM19] which stimulates the galvanic vestibular systems. User wear a necklace with electrodes placed behind the ear. Everytime users move in the virtual world, there is a electric stimulation of the vestibular system. According to [SJM19] this should simulate a movement to the brain. The gap between physical and virtual movement can be tricked in a way through the device. However, the device is still a prototype and so far, it has been tested with only 20 participants. 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 fulfil this requirement, there has to be a balance between a high level of immersion and the reducing of motion sickness. How this is realized we will describe in chapter 3 and 4.

2.5.2 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. The display resolution of VR devices has to be a lot higher compared to TV screens or desktop screens, because there is a magnification of the pixels through the lenses inside HMDs
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. Most of the currently available HMDs for consumers require handheld controllers for user interaction. A body tracking system

would provide more immersion, but requires very accurate sensors. The HTC Vive uses body positioning techniques but still needs handheld controllers. Relocating the player in the virtual world is also a challenge at the time of writing. In chapter 2.2.3 the usage of a 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. [Ska19]
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.

2.6 Measurement of effectiveness of VR technologies

Since we talked about new technologies and its limits in the previous section, the question is now how to measure the impact of VR technologies. How can we be sure that a new technology, hardware or software, actually fulfils the promises? Measurements like performance, screen resolution or weight are criteria which are easy to measure and compare. It is more difficult, to see how effective is a VR technology when it comes to user experience and immersion. In the VR prototype of this work, we focus on this measurements.
We focused our research in finding usability tests which deal with the topics of immersion, presence and user experience. The questionnaire from [WS98] focus on immersion and presence and includes the factors of control, sensory, distraction and realism. This questionnaire can be very useful for usability tests on the VR prototype in this work. However, we could not find a questionnaire which measures the understanding of job roles or similar objectives. Therefore, a new questionnaire has to be developed for the user testing.

3 Design

In this chapter we focus on the prototype VR application which is 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, we shortly introduce, what kind of process model we choose for the implementation. After that, we set the requirements and target group for the prototype. The application will have a fictive storyline, similar to a computer game. In order to design the story, we create a storyboard and present it in this chapter. Once the content is set, we take a look at designing a dialogue with the user and choose suitable user input methods.

3.1 Process model

The objective of this thesis is to develop and evaluate a prototype VR application to simulate the different IT courses offered at NYP. To handle a complex task like this, it is important to apply a suitable process model. The project is developed in a group of two people, therefore a model like SCRUM would not be suitable in this case. Nevertheless, we divided our project in iterations and did a review after the end of each iteration. For this project an iteration of six weeks is chosen. The review contains a presentation in front of the supervisor and two independent lecturers. They give feedback to the prototype and ideas for enhancement. Every week, there is a meeting with the supervisor, in which current problems are discussed and tasks are set. For organizing our tasks we use elements from the process model Kanban, because this model helps keeping an overview of the status and the progress of different tasks. We visualise as cards and display them on a board, which can be seen in figure 3.1. Each card has a description, an assignee and an importance level. The cards are put in different containers: To-do, in progress, ready for testing and done.
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.

Figure 3.1: Board for visualising tasks and workflow.

3.2 Requirements

In order to achieve the main objective of this work, several functional requirements have to be set:

- Introduction: The application should introduce the user to the various IT job roles.
- Start scene: The application starts in a futuristic city which can be explored by the user.
- Interaction: The user can interact with several objects in the starting scene and enter different scenes through this.
- Separation: 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.
- Main storyline: 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:

- Interesting: The user should gain more interest in the job roles introduced in the VR 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

In this chapter we define actors for the VR application. This is done by naming the different stakeholders of the application. 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 joyful 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

Before starting with the design of the application, we have to define the target group. This makes developing of the application purpose easier and also helps keep 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 VR application will be prospective students, who are considering to sing up for a diploma course 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. At this age, people are known to be open to new technologies and trends. 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. 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 story we present in this chapter is the main story line of the final product. We will only be able to realize parts of the main story in this work.
The VR application, hereinafter called game, 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 NYP. When 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 game is to create a delivery drone which will deliver orders from online shops. 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.
During the development process of this work, it will not be possible to implement the whole main story and add all job roles. Instead, we will concentrate on one or two job roles. Other job professions will be added in an iterative process.

3.6 Storyboard

In this chapter, we explain the structure and the details of the main storyline 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. The focus of the storyboard lies on the storyline. Logical orders of scenes are displayed with arrows, textual explanations are added whenever necessary.
For the storyboard of the prototype, a tool called *Invision freehand*[1] is used. In the following, some important parts of the storyboard are presented [2]. We introduced the IT courses from NYP in chapter **??**. The final product should contain all job roles of the courses, but for the prototype in this work we focus our design on the job role of the software engineer scene and the cyber security specialist scene.

3.6.1 World scene and software engineer scene

Figure 3.2: Smart city scene from where the user can explore different IT job roles (own illustration)

The first scene of the storyboard is the smart city scene, as sketched in figure 3.2. Here the user can walk around freely. When the user selects the software engineer building, an assistant asks the user to enter the building. If the answer is “yes", there is a transition into the software engineer lab scene. The lab 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 figure 3.3. This is the first minigame, in which the user should gain understanding in how programming works.

[1] `https://www.invisionapp.com/feature/freehand`
[2] The complete storyboard can be seen in: `https://www.invisionapp.com/feature/freehandhttps://projects.invisionapp.com/freehand/document/Vbu2DFCdd`

Figure 3.3: Software engineer scene: dialog and minigame (own illustration)

The information is taught in a very simplfied way, because the requirements of the application are that it should be 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 the code blocks 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 enters the lab, they are informed 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.

Figure 3.4: Cyber security specialist scene: searching log entries (own illustration)

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 3.4. It turned out that two lab employees have accessed the drone during the last 24 hours. in the second part of the minigame, users search the computers of the two lab employees for further information. One employee has an encoded

Figure 3.5: Cyber security specialist scene: encrypted message

message on their desktop (figure 3.5).
Finally, in the last section, users 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.

3.7 Dialogue design

As seen in the storyboard, the application will contain a lot of conversation between the assistant and the user. The dialogues are very important for the game, because they provide all the relevant information about the IT job roles. The dialogues should be presented in a natural and interesting way, because the application should arouse interest in IT job roles.
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, like it is displayed in the left picture of figure 3.6. This cannot be realised in the same way in VR. Users are not able to focus a screen space text when it is not directly in the middle of their view. Therefore, the dialogue texts will be placed into the virtual world space, like seen in the right picture of figure 3.6. The challenge of this method is to guide users to the position, where the dialogue is displayed. This will be realised by placing the

Figure 3.6: Screen space dialogue panel by [scr11] vs. world space dialogue panel.

dialogue panel close to interactive items in the game. Once the user interacts with the items, the dialogue starts. The user is already focusing the objects and will 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

In chapter 2 we already introduced the different user interaction methods in VR which are commonly used. In this chapter we take a look at the different methods and evaluate, which method is the most suitable for the prototype VR application. There are several constraints and limitations in user interaction because of the hardware and the budget for the project. User interactions should also fit to commonly known user interactions like touch gestures or first person walking in computer games.

First of all we have to evaluate, what kind of user input the Samsung Gear VR as well as the Oculus Go provide. The following input methods are available with the Samsung Gear VR: A wireless handheld controller or a touchpad attached to the HMD. Oculus Go provides a handheld controller which works the same way as the controller of Gear VR, but no touchpad at the HMD.

The handheld controller has gyro-sensors to detect its rotation but there is no position tracking. This means that the current controller position is not mapped into the virutal world. As a workaround it is possible to reset the positioning of the controller with a button click. The 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/-trackpads, which are described in figure 3.7. The trigger button is a primary button, which can be pressed with the index finger. There is a trackpad which registers the thumb position and has a primary button as well. The last primary button is the back button. The controller registers

Figure 3.7: Samsung Gear VR handheld controller [Sam19a]

the hand/arm rotation through sensors. There are home buttons and the volume buttons which are reserved by the system. The handheld controller of Oculus Go has the same buttons and functionalities, except the volume buttons which are located at the HMD directly.
The touchpad, located on the side of the HMD from Samsung Gear VR, 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 touchpad is only available for the Samsung Gear VR. We want the application to work with Oculus Go as well as with Samsung Gear VR. Therefore, the user input development will be focused on the usage of the handheld controller.
Looking at the story board, several user interactions can be 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 Oculus. It is better to reuse well established 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 Oculus 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. To relocate the lines of code, the drag and drop method is chosen. By pointing and clicking on an object, it starts to follow the controller movements. Clicking again, the current position is applied to the object and its position is fixed in the world space. This is similar to a drag and drop action on 2D interfaces. Every object which has an interaction, will be highlighted 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

In this chapter we take a look at the application from the developer's point of view. It is a documentation of the development process of the VR application. At the beginning, we give an introduction into the hardware and software tools on a technical base. Later we describe the development process of the smart city scene and the software engineer scene from the environment design to the deployment to the device. Problems and challenges which occurred during the development will be discussed and their solutions presented.

4.1 Overview about Unity

Unity is a multi platform game engine. It is available for Windows, macOS and since 2019 for some Linux distributions as well. A game engine holds a bundle of tools which are necessary for game development such as tools for graphical design, audio and scripting. Game engines support developing games or applications with a lot of 2D and 3D graphics and also support developing VR applications. [DBEK17]
For the development of our game, the engine Unity and the *Oculus Unity SDK* are used. The Oculus Unity SDK is a support library for Samsung Gear VR and Oculus Go. There are other game engines besides Unity, but we chose to work with this engine because it is beginner friendly, according to [DBEK17], 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.
The Unity editor, as seen in figure 4.1, is the main tool which is used to create our game. 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 [Uni19b]:

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 in Unity define a specific property and always belong to a game object. Every game object can have a variety of components attached. Components change the appearance and behaviour of a game object in the game. There are several build-in components available in Unity, for example the Rigidbody. This component adds physical behaviour to a game object. With the scripting API it is possible to create highly customisable components. Unity gives an

Figure 4.1: Screenshot of Unity editor (own image)

overview of all components of the selected game object. New components can be attached in the editor.

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. The properties of the component are saved within the prefab.

Materials and shaders 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. Shaders define what properties of a material can be edited. Shaders are scripts which define the rendering of the material through algorithms. A 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 or an audio source for playing music. However, it is possible to create components and attach a script to it for customizing the behaviour of a game object. A script can either be written in C# or a modified JavaScript by Unity. When using C#, the .NET environment is the base for scripting in 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 inherits 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. [Uni19a]
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 by assigning values in the code or assigning values in the editor with the component view. The second approach is only possible with object variables which are public. This is why many variables are public in this project, although they are not used outside their own class.

4.1.2 Unity and Samsung Gear VR support

In Unity, VR support can be simply enabled by ticking a box in the player settings. This invokes rendering the graphics in a split screen mode with one screen for each eye. However, to make a game which is compatible with the Samsung Gear VR or the Oculus Go, more work has to be done. First of all, we have to import the Oculus support library. It can be downloaded from the Unity asset store or the Oculus developer website: [Occ18]. This library contains a variety of components and prefabs. For this prototype the `OVRCharacterController` prefab from the SDK is used. It contains all necessary scripts and objects for the communication between Unity and the VR hardware. The `OVRCharacterController` 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 has a script called `OVRInput` attached. This script can register connected controllers and input from Samsung Gear VR as well as from Oculus Go.
To make the game playable on the HMD, the above described prefabs have to be dragged into the scene. Additional to the `OVRCharacterController`, 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. For the complete task in this section, there was no need to write code. All we did was adding examples and prefabs from the Oculus support library to our scene.

4.2 Implementation of the smart city scene

According to the storyboard from chapter 3, 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 environment 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

Figure 4.2: Screenshot of smart city environment (own image)

image with an urban environment is chosen. The image is a 360 photography of a place in a city surrounded by skyscrapers. 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.

Figure 4.3: Software Engineering laboratory entry point (own image)

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, the *Urban Environment Pack*[1] from the Unity asset store is used. It contains a high variety of pre-designed objects like buildings, streets and traffic lights, plants and urban obstacles. Figure 4.3 shows an extract of the city environment. Initially we used a different asset for the smart city scene. We changed it to the asset seen in 4.3 because we got feedback that the previous environment lacks interesting details and variety. After changing to the new asset the feedback was positive. 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. An example can be seen in figure 4.3. This buildings will become interactive in the further development process and should be recognized easily.

[1]Link to the asset store:`https://assetstore.unity.com/packages/3d/environments/urban/urban-environment-pack-90000`

4.2.2 Implementation of the dialogue system

Figure 4.4: UML class diagram for the dialogue component (own image)

The application contains several conversations with the virtual assistant, as seen in in the storyboard in chapter 3. The dialogues in the game between the virtual assistant and the user take place in various places 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 the design of the Unity framework, 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 4.4. 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. 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, called `sentences` in the dialogue script. Every entry in `sentences` is the text which should be displayed on the panel. When the user clicks the continue button, the next entry from `sentences` 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 displays 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 in the dialogue panel in the game. This is necessary in the world scene when asking the user to enter a building. Users can either agree and invoke a scene change or refuse and continue exploring. An example for this type of dialogue is seen in figure 4.5.

Figure 4.5: Example of a dialogue, asking to enter a building (own image)

Internally, 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 corresponding building. Managing the correct order is done by implementing a state machine pattern. The `DialogManager` class holds a reference of a `stateMachine` variable which defines different states and sets their logical order. 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. For the world scene, a state machine diagram is displayed in figure 4.6: After displaying the beginning dialogue in `StartDialog`, `NoDialog` is entered. When the user interacts with a building the software engineer dialogue (or another job role dialogue, implied by `OtherJobroleDialog`) is displayed, following by the question to enter the lab in `SEQuestion`. If the user refuses to enter the lab and answers with "no" we go back to `NoDialog`, otherwise there is a scene transition and the current scene ends.

Figure 4.6: State machine diagram of the dialog flow in the world scene (own image)

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.
We had two different devices available for testing: Samsung Gear VR and the Oculus Go. They are described in more detail in section 2.3. At 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, we used Samsung Note5. The first step is to switch 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 `Plugins` folder of the project. After this step, we set the project in Unity 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, 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 can be done by 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. We deployed the game on the device directly and no phone has to be attached to the HMD. The process of building with Unity works the same way as 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 of attaching and detaching 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 development, a continuous testing is necessary. We tested the game every time when completing a task. During testing, we played the part of the game we changed and tried to cover as much scenarios as possible. Sometimes, VR specific functionalities had to be tested, but most of the time the testing affected only the gameplay changes. 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 it is necessary. In other cases the game is played on the desktop.
The problem of this procedure is that the input methods of both target platform differ: For the VR headset, the handheld controller is used, while on the desktop PC mouse and keyboard are the primary input methods. Therefore a solution has to be found on how to make the game playable on both platforms without changing the code when switching the target platforms.
To remember, Samsung Gear VR as well as Oculus Go use a handheld controller as user input method. Figure 4.7 displays the input buttons of the Samsung Gear VR controller in more detail. On the other hand, the keyboard and mouse is used as input device when the game is played on desktop.

Figure 4.7: Handheld controller with button description. Edited from [Sam19a]

When switching from the desktop platform to Android 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 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 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 of the controller to move forward with a constant speed. The direction of moving is set by the direction in which the user is currently looking.

Figure 4.8: UML class diagram for managing different target platforms (own image)

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 in line 2, which is always the center of the screen. In line 3 it is checked whether the ray hit a game object. If the condition is true, the game object which was pointed at is returned in the `hit` variable.

```
1 RaycastHit hit;
2 Ray ray = camera.ScreenPointToRay(Input.mousePosition);
3 if (Physics.Raycast(ray, out hit))
4 {
5    Transform objectHit = hit.transform;
6 }
```

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 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 in the desktop version, whereas in the VR version the ray is initialized from the controller in the game. In order to make the game work on the VR device, the initialization of the ray has to be changed.

```
1 RaycastHit hit;
2 if (Physics.Raycast(controller.transform.position,
3    controller.transform.forward, out hit))
4 {
5    return hit.collider.gameObject;
6 }
```

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

The method `Physics.Raycast` in line 2 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 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 called `UniversalInput` and 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. How this is done can be seen in listing 4.2.

```
1  if (Application.isEditor)
2  {
3     var ray = new Ray(Camera.main.transform.position,
4              Camera.main.transform.forward);
5     RaycastHit hit;
6     Physics.Raycast(ray, out hit, 500);
7     return hit;
8  }
9
10 Physics.Raycast(gameController.transform.position,
11          gameController.transform.forward, out hit);
12 return hit;
```

Listing 4.3: platform specific ray initializing

In line 1, we check whether the game is played in the Unity editor on desktop. If this is the case, the ray is created from the main camera. Otherwise the ray is created from the `gameController`. This object stores the current connected handheld controller. The `UniversalInput` class 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 4.8 it can be seen which classes use `UniversalInput`. The UML-diagram shown below only shows the method and fields relevant for the problem case of this chapter. Subclasses are not displayed in the diagram. The controller classes of the different game scenes use the methods of `UniversalInput` 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

In chapter 3 we designed the storyboard for the software engineer and the cyber security analyst. For this prototype we choose to implement only the software engineer scene. Adding another scene to the game would go beyond the scope of this work and would not provide new findings in the effectiveness of this prototype. However we create interfaces in the code as well as in the environmental design to add all remaining job roles later.
The software engineer scene is a laboratory which can be entered through the world scene. The scene contains a minigame through which users can experience programming themselves. The laboratory design is held in a futuristic look with 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 minigame. 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.

4.3.1 Coding minigame

As defined in the storyboard, the minigame 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 minigame, the user has to sit on the work desk in the scene.

Figure 4.9: Screenshot of the coding minigame (own image)

The screenshot in figure 4.9 describes the view of the minigame after sitting down on the chair. The code plates seen in the screenshot 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.
In the following we explain how selecting and relocating of the code plates is realised. When the user clicks on a code plate, it always follows the position of the controller or the mouse. When the user clicks again, the code plate is released and has a fixed position again. This mechanism is similar to the drag and drop method. To realise this feature, we create a script called `Draggable` and add it as component to each code plate. The script has a variable called `current` which indicates if the plate is currently been dragged (state `Dragging`) or is in a fixed position (state `Ready`). When the user clicks on the plate, the code in listing 4.3.1 is executed.

```
1  if (current == State.Ready)
2  {
3     float distance = Vector3.Distance(gameObject.transform.position,
4        Camera.main.transform.position);
5     if (distance > maxDistance) return;
6     current = State.Dragging;
7  }
8  else if (current == State.Dragging)
9  {
10    current = State.Ready;
11 }
```

Listing 4.4: State change when clicked a code plate

Inside the component we define an enumeration called `State`, which represents the states of a code plate. When a code plate is in state `Ready`, which means it is not dragged, it is checked whether the object is not too far away in line 5. If it is in range, the state is changed. If the code plate is already in state `dragging`, it is changed back to `Ready`. In the `Updade()` method, the position of the code plate is updated according to the mouse or controller position only if the current state is `Dragging`

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 by the `Programming` class which is responsible for all visible changes in the minigame. 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 minigame 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

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 we realized this by highlighting interactive objects when the user points at them. For additional support, a textual hint appears, similar to a tooltip. The textual hint was inspired by the game Labster, introduced in chapter 2.4. In this game, there were textual hints as well which give information about the current task and how to interact with the object.
Every game object which should appear as interactive has 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 4.10.

Figure 4.10: Highlighting of interactive objects with additional task description

In example 4.5, 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. 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.

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

Listing 4.5: Shader change when focusing object

For the hint label, a game object containing a plate and a 3D text, seen in figure 4.10, as well as the component `HelpLabel` is needed. We added the component to the game object which should be interactive, for example the chair in the software engineer scene, and set a reference to the 3D text object. The content of the hint text is set in the editor. Every time the user points at the chair, the hint label appears either in the middle of the screen or attached to the controller. The hint label disappears when the user unfocus the object. Internally the name of every interactive game object which has the `HelpLabel` script attached is added in a global list. In the `Update()` method of the `HelpLabel` it is checked whether the current object is in the list. If this is the case the hint label game object gets enabled with the accordint hint text. It also follows the position of the main camera or controller with the same method as described in section 4.3.1.

5 Testing

In this chapter we examine whether the application fulfils the requirement of helping prospective students in choosing their profession by visualizing IT job roles. Therefore we create and execute user tests with the prototype VR application. In general, it is difficult to find an objective method to prove or disapprove the hypothesis of this project. This is because VR creates an unique user experience which is subjective and varies with every user. The focus in this chapter lies on testing the effectiveness of the prototype. Secondly, we want to improve the user experience with the feedback from the test persons. Several test persons play the VR application and answer questions about their subjective experience. In this chapter we explain the testing procedure and present the results of the tests.

5.1 Testing group and environment

We tested the prototype with Samsung Gear VR and Galaxy Note5. For recruting participants for the test, we asked students from the School of Information Technology to voluntarily take part in our tests. One participant, in the following called user, was tested at a time. Every user played the application for the first time. In following, we give an overview about the users and their backgrounds.

Total number of participants: 27

Number of NYP students: 24 (15 year-1-students, 9 year-3 students)

Number of users with no VR pre-experience: 15

Number of users which have used VR 1-3 times before: 11

Number of users which sometimes use VR: 1

5.2 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 perception 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, users answer some general questions about their age and profession. After this we ask them 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. We fit the headstraps of the VR device straps for them, so that they feel comfortable.

Playing prototype The users 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 users play one full walkthrough of the prototype which includes exploration of the world scene, clicking on the interactive buildings 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 the virtual assistant tells them to exit the game and put off the VR headset.

After playing prototype When finished with playing the prototype, the users answer the second part of the questionnaire. This includes the same questions about the software engineer jobe role, they were asked before as well as questions about the game in general.

5.3 Questionnaire

The questionnaire is divided into five groups. The first question group includes questions about the users' age, the school and year they study if they are at NYP, the current profession if they are not at NYP and the pre-experience in VR.
The second group contains three statements about the job role of the software engineer. These statements are self-evaluations on how familiar the users are with the job role of a software engineer. The users can agree or disagree to the statement with a likert scale from 1 to 7, where 1 means "strongly disagree" and 7 "strongly agree". All statements are answered before playing the VR game. The statement in the questionnaire are:

- I know what the job role of a software engineer is about
- I am interested in signing up for an information technology related course at NYP
- I can imagine to work as a software engineer

The third question group contains the same statement as in the previous group, but the users answer them after playing the VR game. There is the additional statement: "I feel that I have a better understanding about the job role of a software engineer after playing the VR game". The users should evaluate their subjective feeling of the impact of the VR application with this statement.
The fourth question group contains statements with the same likert scale about the immersion. This statements were selected and slightly adapted from [WS98]. The last questions are about general feedback for the application. Questions in that section contain whether the mini game was difficult enough, how the user guidance was and whether the test persons liked the balance of the game elements.

5.4 Results

The hypothesis of this project is, whether the application helps students in improving their understanding of IT job roles. In order to approve or disapprove this thesis, the three statements of the second and third question groups are evaluated.
The fourth and fifth question group affect the effectiveness of the VR application indirectly. This questions ask about the immersion of the user and evaluate the user guidance of the game. Based on the results of this part of the test, improvements in the user experience can be made which should improve the overall experience and increase presence.

5.4.1 Effectiveness of the VR prototype

The testing method which is used is the paired t-test, because we have two samples which are dependent. This test was applied with the collected user answers from before playing the prototype and the answers after playing the prototype. The difference of both data sets is expressed through the variable μ_d, which is the subtraction between after playing values and before playing values .
To evaluate the statements, we create a null hypothesis and an alternative hypothesis. The null hypothesis in this test is: “There is no improvement in the statement evaluation after playing the prototype.", so it is $H_0 : \mu_d \leq 0$.
The alternative hypothesis is: “The statement evaluation is higher after playing the prototype.", therefore it is $H_1 : \mu_d > 0$. As a significance level $\alpha = 0.05$ is chosen.

statement	t-value	p-value
“I know what the job role of a software engineer is about"	5.720	$5.088 * 10^{-6}$
“I am interested in signing up for an information technology related course at NYP"	4.264	0.0002
“I can imagine to work as a software engineer"	5.112	$2.499 * 10^{-5}$

Table 5.1: Paired t-test results for the three statements

Applying the paired t-test to the three statement, we get the results seen in table 5.1. Looking at the p variables, all three values are below $\alpha = 0.05$, which means that the probability that H_0 is true is below 5%. This means that the null hypothesis can be discarded.
With a significance level of 0.05, is is therefore true that the evaluation of the three statements is higher after playing the prototype.

5.4.2 Discussion

The second statement has has low value because 23 out of 27 test persons already signed up for an IT course at NYP. Even if the test results show a positive change, the results cannot be used for evaluation. Under this aspect, we cannot state whether the VR application encourages prospective students in signing up for an IT course at NYP. However, the first and second statement clearly show an increase in the evaluation of the statements. After playing the VR application, participants feel that they know more about the job role of the software engineer than before playing. They also show a higher interest in working as a software engineer. Based on these results, we can say that the VR application helps students in understanding software engineer job role better. Beyond that, the application increases the interest of working as a software engineer.
The application was tested with the laboratory scene of the software engineer only. In order to ensure that students improve their understanding of all job roles in the IT industry, the test has to be repeated once all remaining laboratory scenes are completed. Since the effectiveness of the software engineer scene is already proven, we can expect similar results, if the implementation of the remaining scenes will use the software engineer scene as a model.

5.4.3 Immersion and user guidance

In order to evaluate the grade of immersion in the game, the quantity of options chosen by the participants are calculated. Two histograms of statements with meaningfull results are shown in figure 5.1 and 5.2.

Figure 5.1: Histogram of the answers from statement 1

Figure 5.2: Histogram of the answers from statement 2

Figure 5.3: Histogram of the answers from statement 3

24 out of 27 users either agreed or strongly agreed to the statement from figure 5.1. None of the users chose a disagreeing option. Only one user had a neutral position to the statement. A similar result could be seen in the evaluation of the statement “I lost track of time". 26 out of 17 users either agreed or strongly agreed to the statement, only one uer slightly agreed.
A little different results can be seen int the evaluation of the statement "The interactions with the virtual environment felt natural for me" from figure 5.2. The majority of users, 18 out of 27, agreed to the statement, whereas only 2 strongly agreed. 2 users disagreed or slightly disagreed to the statement.
Another relevant result from the user test is, how clear the instructions were and how difficult the mini game was for the users. As seen in figure 5.3, 5 persons slightly disagreed to the statement “I knew what to do all the time during the game". The most frequently chosen answer was slightly agree. In the statement “I found it easy to complete the task in the software engineering

lab", 10 users strongly agreed, whilst 1 person disagreed and 8 persons only slightly agreed to the statement. This results match with the subjective observations of the supervisors during the tests. Some users had no problems at all with solving the mini game, others struggled and could not solve the game without help.

5.4.4 Discussion

In general, the users felt very immersed with the virtual world. Although the interactions in the game did not feel natural for some, the overall result of the immersion question group was positive. The reasons, why some users found the interactions not very natural can be because of hardware limitations. The device, which was tested on only had one handheld controller and there were no body positioning trackers available. Therefore, to create a game with a higher level of physical immersion, a HMD with more sensors and more immerse controller mechanisms should be used. Another reason could be that we used a 3D environment with not enough details and variety. After the testing was finished, we changed the environment to be more detailed and complex.
The mini game and the free exploration in the beginning of the game helped that the user felt involved in the game. The majority of users is pleased with the user involvement.
When it comes to user guidance, the majority of users did not know what to do during the game all the time. Reasons for this results could either be misunderstanding instructions by the assistant or a poor guidance to the next task, e.g. finding the interactive desk in the software engineering scene. To improve the user guidance, the assistant dialogue in the beginning was changed to have a clear instruction to find interactive buildings. More description hint labels were added to interactive objects. In the software engineer scene for example, we made the desk as well as the chair interactive and added a hint label for both. Looking at the mini game, some person rated it to be slightly too difficult. This was also an observation we made when executing and supervising the test. It was observed that some people needed help from someone who already knows how to solve the mini game. Based on this results, we decided to make the mini game in the software engineer scene easier. To help the test persons in finding the correct order, the shape of the code plates was changed to look like a puzzle. This should indicate the correct order of the code fragments. The idea was inspired by the educational programming language scratch [1].

[1] `http://scratch.mit.edu/`

6 Conclusion

6.1 Summary

We implemented a protoype VR application which contains a futuristic city and a laboratory, representing the job role of a software engineer. An interface for the remaining job role laboratories was added in the game. We designed the scene for the cyber security specialist job role and scripted a storyboard but we could not implement the scene because it would have been too extensive in the scope of this work. A main storyline for the whole VR game was scripted as well.

There were user tests to prove whether the application helps students and prospectives in understanding their future job roles in the IT industry. The outcome of the tests showed positive results. Participants rated their knowledge of the software engineer job role higher after playing the VR prototype. They could imagine more likely to work as a software engineer compared to their answers before playing the application. However the study is based on the subjective perception of the participants and the results would have been more accurate if there was a higher number of test persons. Unfortunately it was not possible to set up a test with a wider and more objective extent. This would have been gone beyond the scope of this work. The study does not meet international academic standards but should give an impression of the effectiveness of the VR application. It is necessary to repeat the study with a different setting once the implementation is finished. During the user tests, we also tested the usability and user guidance of the game. Based on the results we could improve the guidance by adding more textual hints and highlighting interactive items.

We must not forget that the objective was to develop a prototype, not a full application. This objective was fulfilled. It is shown through the user test results that the prototype creates higher interest in the software engineer job role. The methods and concepts used for the implementation of this work's prototype can be reused to implement the remaining job role scenes.

6.2 Outlook

This work shows that it is possible to create a virtual experience which lets users explore real life jobs on first hand. There is no need to rely on passive reports from professionals, or doing a time consuming internship. This project will be continued by other students of NYP. They will design and implement the remaining IT job role scenes. Furthermore the dialogues in the game have no voice over yet. This feature can be added in future as well. The application can be adapted to support standalone HMDs. This could increase immersion, because the those HMDs have stronger hardware. We expect that the school can present the VR application on open houses to prospective students, once the implementation is finished.

The idea of using VR to simulate job roles can be extended to other industries. Through applications similar to this work, the interest in job roles with a less positive reputation, like nursing

or crafting, could be increased.
Based on the outcome of this work, we can see that VR creates immerse experiences which cannot be created by traditional media. Because VR is currently a highly researched topic, there will be cheaper and more powerful hardware available on the market to create even more immerse virtual experiences.

Bibliography

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

[B+16] BOZGEYIKLI, Evren u. a.: Point and Teleport Locomotion Technique for Virtual Reality, 2016, S. 205–216. – online available at https://dl.acm.org/citation.cfm?id=2968105

[Bod15] BODEKAER, Michael: *This virtual lab will revolutionize science class*. https://www.ted.com/talks/michael_bodekaer_this_virtual_lab_will_revolutionize_science_class#t-674479. Version: 2015 (TEDx)

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

[DBEK17] DICKSON, Paul E. ; BLOCK, Jeremy E. ; ECHEVARRIA, Gina N. ; KEENAN, Kristina C.: An Experience-based Comparison of Unity and Unreal for a Standalone 3D Game Development Course. In: *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*. New York, NY, USA : ACM, 2017 (ITiCSE '17). – ISBN 978–1–4503–4704–4, 70–75

[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

[Gol15] GOLDSTONE, Will: *Get started with VR! Sample pack and Learning Articles*. https://blogs.unity3d.com/2015/12/09/get-started-with-vr-sample-pack-learning-articles/. Version: 2015

[H+17] HEUBLEIN, Ulrich u. a. ; DEUTSCHES ZENTRUM FÜR HOCHSCHUL- UND WISSENSCHAFTSFORSCHUNG (Hrsg.): *Zwischen Studienerwartungen und Studien-*

wirklichkeit: 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

[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

[Occ18] OCCULUS: *Oculus Utilities for Unity*. https://developer.oculus.com/downloads/package/oculus-utilities-for-unity-5/. Version: 2018. – last accessed: 8/8/2019

[Per19] PERTZBORN, David: *Virtual Reality: Diese VR-Headsets erscheinen im Jahr 2019*. https://www.computerbase.de/2019-04/virtual-reality-diese-hardware-2019/. Version: 2019. – last accessed: 8/9/2019

[Pes14] PESCE, Maurizio: *Samsung Gear VR*. https://commons.wikimedia.org/w/index.php?curid=39894108. Version: 2014. – License: CC BY 2.0

[Rob15] ROBERTSON, Adi: *Oculus Go review: mobile VR, minus the phone*. https://www.theverge.com/2018/5/1/17306458/oculus-go-standalone-vr-headset-review. Version: 2015. – last accessed: 8/9/2019

[Sam19a] SAMSUNG: *Gear VR Controller*. https://www.samsung.com/uk/mobile-accessories/gear-vr-controller/. Version: 2019. – last accessed: 21/07/2019

[Sam19b] 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

[scr11] *A common layout for a visual novel game*. https://commons.wikimedia.org/w/index.php?curid=1197564. Version: 2011. – License: CC BY-SA 3.0

[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

[SJM19] SRA, Misha ; JAIN, Abhinandan ; MAES, Pattie: Adding Proprioceptive Feedback to Virtual Reality Experiences Using Galvanic Vestibular Stimulation. In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, ACM, 2019 (CHI '19). – ISBN 978–1–4503–5970–2, 675:1–675:14

[Ska19] SKARREDGHOST: *Cyberith announces 2nd gen Virtualizer, still thinks about its backers*. https://skarredghost.com/2019/05/18/cyberith-

virtualizer-elite-2-backers/. Version: 2019. – last accessed: 7/7/2019

[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

[Syc18] SYCED: *Oculus Go - 7*. https://commons.wikimedia.org/wiki/Category:Oculus_Go?uselang=de#/media/File:Oculus_Go_-_7.jpg. Version: 2018

[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

[Uni19a] UNITY TECHNOLOGIES (Hrsg.): *Unity User Manual (2019.2) - Creating and Using Scripts*. Unity Technologies, 2019. https://docs.unity3d.com/Manual/CreatingAndUsingScripts.html

[Uni19b] UNITY TECHNOLOGIES (Hrsg.): *Unity User Manual (2019.2) - Creating Gameplay*. Unity Technologies, 2019. https://docs.unity3d.com/Manual/CreatingGameplay.html

[Vit] VITILLO, Antonio: *What is Standalone Virtual Reality, and Why Are Enterprises Betting On It?* http://www.isaca.org/Knowledge-Center/Blog/Lists/Posts/Post.aspx?ID=975&utm_referrer=direct%2Fnot%20provided/oculus-go-standalone-vr-headset-review. – last accessed: 1/9/2019

[WS98] WITMER, B. G. ; SINGER, M. J.: Measuring Presence in Virtual Environments: A Presence Questionnaire. In: *Presence* 7 (1998), June, Nr. 3, S. 225–240. http://dx.doi.org/10.1162/105474698565686. – DOI 10.1162/105474698565686. – ISSN 1054–7460