Intelligent Tutoring in Virtual Reality for Highly Dynamic Pedestrian Safety Training

Dissertation

submitted towards the degree
Doctor of Engineering (Dr.-Ing.)
of the Faculty of Mathematics and Computer Science
of Saarland University

Yecheng Gu

Saarbrücken, October 2018

Date of Colloquium:	April 01, 2019
Dean of the Faculty:	Prof. Dr. Sebastian Hack
Chair of the Committee:	Prof. Dr. Gert Smolka
Reviewers:	Prof. Dr. Jörg Siekmann
	Prof. Dr. Roland Brünken
	Dr. Sergey Sosnovsky
Academic Assistant:	Dr. Klaus Fischer

Abstract

This thesis presents the design, implementation, and evaluation of an Intelligent Tutoring System (ITS) with a Virtual Reality (VR) interface for child pedestrian safety training. This system enables children to train practical skills in a safe and realistic virtual environment without the time and space dependencies of traditional roadside training. This system also employs Domain and Student Modelling techniques to analyze user data during training automatically and to provide appropriate instructions and feedback. Thus, the traditional requirement of constant monitoring from teaching personnel is greatly reduced. Compared to previous work, especially the second aspect is a principal novelty for this domain. To achieve this, a novel Domain and Student Modeling method was developed in addition to a modular and extensible virtual environment for the target domain. While the Domain and Student Modeling framework is designed to handle the highly dynamic nature of training in traffic and the ill-defined characteristics of pedestrian tasks, the modular virtual environment supports different interaction methods and a simple and efficient way to create and adapt exercises. The thesis is complemented by two user studies with elementary school children. These studies testify great overall user acceptance and the system's potential for improving key pedestrian skills through autonomous learning. Last but not least, the thesis presents experiments with different forms of VR input and provides directions for future work.

Zusammenfassung

Diese Arbeit behandelt den Entwurf, die Implementierung sowie die Evaluierung eines intelligenten Tutorensystems (ITS) mit einer Virtual Reality (VR) basierten Benutzeroberfläche zum Zwecke von Verkehrssicherheitstraining für Kinder. Dieses System ermöglicht es Kindern praktische Fähigkeiten in einer sicheren und realistischen Umgebung zu trainieren, ohne den örtlichen und zeitlichen Abhängigkeiten des traditionellen, straßenseitigen Trainings unterworfen zu sein. Dieses System macht außerdem von Domain und Student Modelling Techniken gebrauch, um Nutzerdaten während des Trainings zu analysieren und daraufhin automatisiert geeignete Instruktionen und Rückmeldung zu generieren. Dadurch kann die bisher erforderliche, ständige Überwachung durch Lehrpersonal drastisch reduziert werden. Verglichen mit bisherigen Lösungen ist insbesondere der zweite Aspekt eine grundlegende Neuheit für diesen Bereich. Um dies zu erreichen wurde ein neuartiges Framework für Domain und Student Modelling entwickelt, sowie eine modulare und erweiterbare virtuelle Umgebung für diese Art von Training. Während das Domain und Student Modelling Framework so entworfen wurde, um mit der hohen Dynamik des Straßenverkehrs sowie den vage definierten Fußgängeraufgaben zurecht zu kommen, unterstützt die modulare Umgebung unterschiedliche Eingabeformen sowie eine unkomplizierte und effiziente Methode, um Übungen zu erstellen und anzupassen. Die Arbeit beinhaltet außerdem zwei Nutzerstudien mit Grundschulkindern. Diese Studien belegen dem System eine hohe Benutzerakzeptanz und stellt das Potenzial des Systems heraus, wichtige Fähigkeiten für Fußgängersicherheit durch autodidaktisches Training zu verbessern. Nicht zuletzt beschreibt die Arbeit Experimente mit verschiedenen Formen von VR Eingaben und zeigt die Richtung für zukünftige Arbeit auf.

Acknowledgments

A number of extraordinary people have supported me in writing this thesis, and I would like to take this opportunity to express my most sincere gratitude towards them.

First of all, I would like to thank my supervisor Prof. Dr. Jörg Siekmann and my closest advisor and collaborator Dr. Sergey Sosnovsky. I thank Prof. Dr. Jörg Siekmann for taking me as his PhD student and for giving me the opportunity to work in his research group. I am very grateful for his mentorship and his support in all aspects of my research, ranging from defining the research topic to finding candidates for the user studies. Dr. Sergey Sosnovsky taught me so much about scientific writing, and it was a great honor to work with him. He was always available for guidance, questions, and discussions. I truly appreciate his words of advice, not only for research but also for problems in real life.

I would also like to especially thank Prof. Dr. Christoph Igel and Dr. Carsten Ullrich for providing me with a daily workplace in the E-Learning group (formerly CeLTech, now EdTech) of the DFKI to conduct my research and for their constant support, especially with regard to planning and execution of the SafeChild project. Thank you for providing me the with the opportunity to work on interesting projects and your ideas and contributions towards this thesis.

My special thanks also go to Prof. Dr. Roland Brünken and Prof. Dr. Babette Park for their advices from the point of view of educational science and to Prof. Dr. Roland Brünken in particular for agreeing to review this thesis.

Definitely not to forget are also all the co-authors, collaborators, students and colleagues (many of whom I call friends) throughout the years. Thank you Thorsten Hey, Patrick Rump, Michael Dietrich, Eric Andres, Patrick Walther, Glenn Schütze, Dr. Jason Orlosky, Dr. Markus Weber, Valeria Palmieri, Andre Günther and Dr. George Goguadze for all the cheerful moments, the fruitful discussions and your contributions to the SafeChild project and this thesis.

Furthermore, I would like to thank all the children, parents and teachers who participated in the user studies voluntarily and the Software Campus organization for providing a funding opportunity for the SafeChild project.

Last but definitely not least, I would like to thank my family for their emotional support, their never ending love and constant encouragement to carry on with my thesis. I deeply thank my wife Qian for shouldering the weight of long distance relationship with me. Thank you for your sacrifices, your understanding and for giving me the time to finish this thesis. I thank my parents and sisters for always providing me with a place of peace, warmth and security, and my father Dr. Wei Gu in particular for proofreading my thesis. With a heavy heart, my last thank you goes to my grandparents who took great interest in my research and encouraged me to pursue a PhD. I am sorry that you could not witness the end of my PhD journey but I hope you would have been proud of me.

Contents

Chapter 1

Introduction

Effective education in the form of teaching, learning, and training has always been a key pillar for success and accomplishments in human life. This includes not only professional, academic or athletic achievements but also everyday aspects such as understanding the rules and dynamics of a society or being a safe traffic participant as driver or pedestrian. Modern computer technology provides a great number of possibilities to enhance education that can improve not only its effectiveness and efficiency but also the general teaching and learning experience. This ranges from educational videos or lecture notes that can be found in the World Wide Web (WWW), available to the learner whenever and wherever it is required, to sophisticated and highly specialized computer programs dedicated to specific domains.

A special class of such programs makes use of Artificial Intelligence (AI) to perform User Modeling (which is called Student Modeling in this case) and adaptation to understand the needs and state of learning of a user to customize the educational content and its presentation accordingly. Such systems are called Intelligent Tutoring Systems (ITS) and will be presented in detail in Chapter 2. Another class of educational computer programs makes use of Virtual Reality (VR) to provide a graphically realistic representation of a virtual world that is interactive and updated near real-time. Consequently, such systems can immerse the user in a virtual environment that looks and feels highly realistic. This allows users to perform and train/learn certain tasks in a natural way and thus facilitates the transfer of skills to real-world applications. These systems are particularly useful in such cases where the real counterpart of the virtual environment is too dangerous or costly to provide or if it does not scale well with large numbers of learners.

ITS and educational programs that make use of VR are not mutually exclusive, but their number is still limited. One of the reasons is the cost of VR technology that has traditionally been too high for average consumers. This is because VR requires not only high computational and graphical power but also (at least for full immersion) specialized hardware devices to track the users' movements as well as to present the graphics in stereoscopic 3D. However, this fact has been changing in the recent years. Driven by the computer games industry, VR is entering the consumer market with great quality and affordable prices. Although we are still years away from democratization of VR for education, this is the right time to conduct the necessary research to prepare for it. In this spirit, this thesis is dedicated to extending the educational domains that benefit from VR and ITS by a common subject of high importance around the world: child pedestrian safety education. As a consequence, the key domains of this thesis are as shown in Figure 1.1. In the remainder of this chapter, the motivation and challenges behind this decision will be

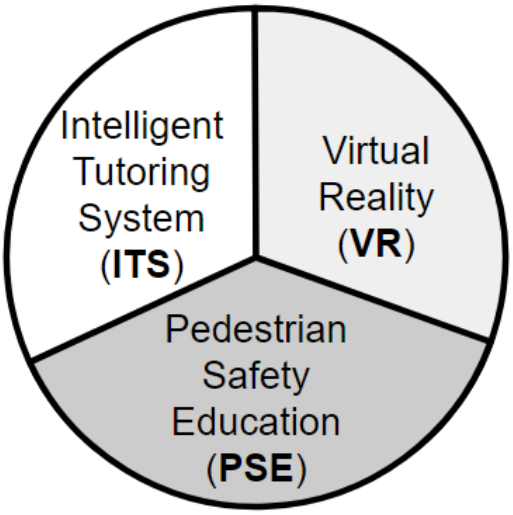


Figure 1.1: Key domains of the thesis

discussed first. This is followed by a more specific formulation of research questions. An overview of the approach to tackle these questions will then be given in the next section. Finally, the last section of this chapter provides an outline of the entire thesis.

1.1 Motivation

As a researcher in the area of e-learning or Technology Enhanced Learning (TEL), the main motivation for this research is to make a contribution by finding new ways of using technology to improve learning or training. In this specific case, the technologies of choice are VR and ITS while the target domain for training is child pedestrian safety. This section explains why this specific combination was chosen.

The motivation to engage in VR for this thesis lies in the fascination of the author for modern computer games and the potential therein for learning. The goal of VR is to create an immersive experience for the user using computer technology. A number of computer games have accomplished this with realistic 3D graphics and interactive open worlds. These games are able to motivate users around the world to spend hundreds of hours to engage in different entertaining activities. In order to succeed in these games, complex skills have often to be acquired. For instance, this includes understanding the games' mechanics, coordinating abilities with other players or non player characters (NPCs) and predicting and reacting to enemy actions. By replacing these game related skills with real-life skills, VR based serious games can serve as a great learning platform that provides both an effective but also enjoyable learning experience. Although this concept has already been shown in previous studies, e.g. by Sitzmann (2011), it is still not part of mainstream education. One main reason until now could be the effort required to develop high-quality VR applications and games. Besides the conceptual and design work, game assets including 2D and 3D graphics as well as sound and music have to be produced, and the logic needs to be implemented. However, this landscape is changing with the expansion of the gaming market. Thanks to increases in computational and graphical power in consumer devices and the popularity of gaming on mobile devices, game development became much more streamlined and standardized over the last couple of years. In fact, most VR applications and games nowadays are based on the same set of tools and ecosystems. These development platforms offer convenient high-level editors and a great number of

ready to use assets on their markets to enable even small teams of lightly trained developers to create VR applications and games of great quality. The growth in popularity for VR and AR in both the entertainment and industry market in the recent years has further supported this development. With this problem out of the way or at least strongly relaxed, the doors are open for widespread use of VR based game-like learning software to mainstream education that could bring significant benefits. Therefore, it is highly motivating for the author to deliver a contribution in this area.

The motivation for adding an ITS to the equation comes from the proven success and effectiveness of this technology for education (Kulik & Fletcher, 2016). Although the combination of VR and ITS was studied before (see Chapter 2), the traditionally low availability of VR, as mentioned above, also prevented a more widespread use of such systems. As a consequence, the techniques and frameworks to perform the ITS related processes for VR such as Student and Domain Modeling are not standardized and dedicated to specific use cases. Therefore, one of the goals of this thesis is to investigate possibilities to facilitate the implementation of ITS features for VR based systems.

Last but not least, the domain of child pedestrian education was a natural choice. VR based education is especially beneficial for this domain since it solves one of the key problems. VR provides a safe and controlled yet highly realistic representation of a traffic environment. Due to children's limited abilities to transfer written instructions to actual behavior, it is mandatory to exercise and train safe behavior in real-world scenarios. Previous research has already shown the great potential of VR for this domain (see Chapter 2 for a detailed discussion). Theoretically, the addition of ITS functionalities could then also drastically reduce the reliance on human teachers. The resulting system would thus enable children to conduct effective training whenever and wherever they want.

1.2 Research Questions

There are a number of challenges and questions to be solved in order to achieve the goal stated in the introduction of this chapter. This section presents a list of research questions (RQs) to be tackled by this dissertation with a detailed explanation of their meaning.

RQ1: What are the skills in the domain of pedestrian safety education which are best to be trained in a VR-ITS? Pedestrian safety education, in particular for children, is a complex topic and requires a number of different skills. Not all skills are fit to be trained in a VR environment. For instance the choice of clothing or doing preliminary plannings before engaging in the pedestrian trip. Therefore it is important to identify and select those skills that would benefit the most from VR training to set the right focus.

RQ2: How should the skills and their correct implementation be encoded in a VR-ITS and how should the system evaluate user actions to detect errors? The second research question is related to the common problem of Student and Domain Modeling in ITS design. The skills and knowledge of the pedestrian safety domain need to be encoded in a way such that the resulting system can make judgments about correct behavior and errors while the user engages in exercises. The challenge here is to deal with the constraints of VR (real-time user interaction and scene updates) and the ill-defined, highly dynamic nature of the pedestrian safety task.

RQ3: What VR interface is suited for widespread pedestrian safety training? In terms of VR interfaces, there are a number of variations providing different levels of immersion on different costs. This ranges from full-scale VR rooms with multiple walls as projection surfaces (fully immersive VR) to standard monitor with keyboard and mouse setups (Desktop VR) with other options in between. In order to provide effective training to a large number of potential users, the right setup has to be found.

RQ4: How is the general user acceptance of the main target group of young children for a VR-ITS to teach them pedestrian safety? In previous studies with VR for the use case of pedestrian safety training, most of the teaching task has been carried out by human personnel. If the educational instructions and feedback are to be carried out by an ITS, the general user acceptance needs to be evaluated. Also, the VR environment itself including visual appearance and interaction needs to fit the target user group.

RQ5: Can the resulting system developed in this thesis actually help children improve in pedestrian safety skills and do different feedback strategies cause a difference? At the end of the day, a VR-ITS for pedestrian safety has the goal of improving pedestrian safety behavior through self-training. Despite the limited scope of a single PhD thesis and the interdisciplinary aspect of educational success, the potential and effects of different feedback timing strategies are to be examined for the system developed in this thesis. The goal is to provide a platform and foundation for further research.

1.2.1 Overview of the Approach

This section gives an overview of the research approach with the individual steps that have been taken to tackle the research questions stated above.

1. Conduct extensive research about the state of the art and related work in the areas of Intelligent Tutoring and child pedestrian safety education with a focus on such work that incorporates VR as interface.

2. Select, adapt and/or develop methods with formalisms to represent the content to be taught in the child pedestrian safety domain for an ITS using VR (Domain Modeling). This includes behavior rules and errors.

3. Select, adapt and/or develop methods to assess learner behavior at training time to determine correct or incorrect execution in a near real-time fashion (Student Modeling).

4. Design and experiment with different combinations of VR setups to determine their compatibility with the target use-case of providing child pedestrian training opportunities to children in the near future.

5. Bringing the results of the previous steps together and implement an ITS using VR for the purpose of providing children with a way to train pedestrian safety tasks mostly autonomously.

6. Evaluating the implementation in a real world classroom scenario with the target user group in terms of educational potential and acceptance of the system.

1.3 Thesis Outline

To conclude this chapter, the following section provides an outline of this thesis with a short description of all remaining chapters.

- **Chapter 2 Related Work** presents previous and relevant research within the related research areas Intelligent Tutoring Systems, Pedestrian Safety Education and Virtual Reality Training. A special focus is set on such work that covers more than one of these research areas (Intelligent Tutoring in Virtual Reality or Virtual Reality for Pedestrian Safety Education).

- **Chapter 3 Behavior Tree** based Tutoring introduces a novel Domain and Student modeling approach based on the concept of Behavior Trees (BTs) called Behavior Tree Tracing (BTT). The chapter compares this novel method with existing methods and points out strengths and weaknesses. The formal definition of BTT is accompanied by practical examples.

- **Chapter 4 Modeling the Pedestrian Safety Domain** uses the BTT approach presented in Chapter 3 and implements it for the Pedestrian Safety Domain. It contains detailed descriptions of the skills and rules considered in this thesis and shows with several examples how the resulting model can be used to evaluate learners‘ exercise performance and provide helpful pedagogical intervention at the right time.

- **Chapter 5 VR Interfaces** is about the investigation of different VR interfaces for pedestrian interaction in an open real-time city simulation including moving cars, traffic lights and zebra crossings. The chapter describes the simulation and its parameters first before introducing several hardware combinations with their interaction model. This includes a fully immersive setup with a Head Mounted Display (HMD) and motion tracking wearables, a semi-immersive setup with multiple 3D monitors and a motion tracking camera and, last but not least, a standard desktop VR setup with a single monitor and conventional input devices. The strengths and weaknesses of each setup will be discussed, and an explanation will be provided why the desktop setting will be used throughout the rest of the thesis.

- **Chapter 6 Implementation** presents the implementation of the SafeChild system, an ITS with a VR interface for Pedestrian Safety aimed at children. This process includes 3D modeling, 3D scene composition as well as programming for the interaction logic, traffic simulation, and BTT related features. The description includes the tools and frameworks used, the programming language and development environment as well as architectural and use case diagrams.

- **Chapter 7 Evaluation** is dedicated to user studies with the target user group of children that have conducted training with the SafeChild system. The first pilot study delivered preliminary results about acceptance of the systems general look and feel while the second main study focused on actual learning effects under different types of pedagogical feedback and instructions.

- **Chapter 8 Summary and Conclusion** brings the thesis to an end by recapping the major points of this thesis. It then addresses the research questions formulated in Section 1.2 to provide the answers based on the results of this thesis. In addition, the limitations of this thesis will be discussed, and finally, a discussion about future work and research directions will be provided.

Chapter 2

Related Research Areas

This chapter summarizes the research areas of this thesis. Since the topic of this thesis is Intelligent Tutoring in a highly dynamic Virtual Reality environment using the example of the Child Pedestrian Safety Education domain, this chapter introduces Virtual Reality, Intelligent Tutoring, and Pedestrian Safety Education. Further, a more detailed look is taken into the joint research areas Pedestrian Safety Education using Virtual Reality as well as Intelligent Tutoring Systems using Virtual Reality. The scope of this chapter is illustrated in Figure 2.1. It also shows that there is, to the best of my knowledge, no previous work on the combination of Pedestrian Safety Education and Intelligent Tutoring.

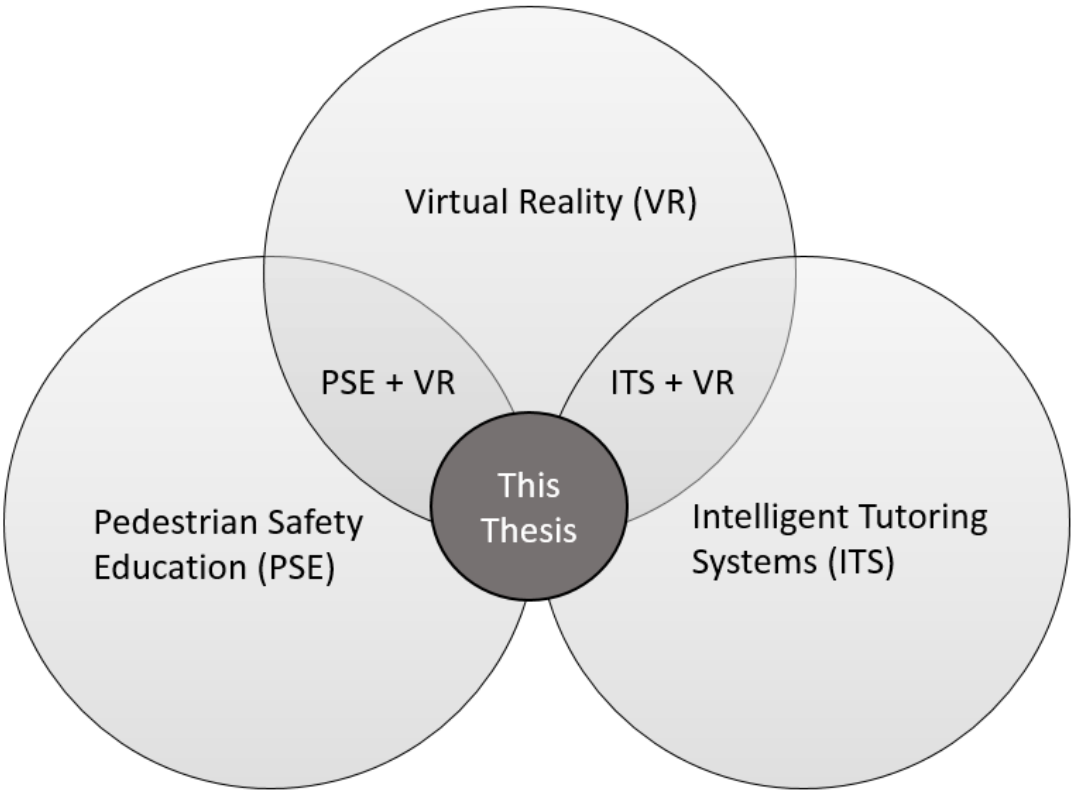


Figure 2.1: Related research areas

2.1 Virtual Reality

2.1.1 General Overview

The general idea of Virtual Reality (VR) is to use computer technology to either simulate reality or to create artificial worlds that appear highly realistic to the user. The initial concepts of VR are at least half a century old, and an often cited origin is Sutherland's (1965) vision of the Ultimate Display which he describes as "... a room within which the computer can control the existence of matter. A chair displayed in such a room would be good enough to sit in. Handcuffs displayed in such a room would be confining, and a bullet displayed in such a room would be fatal. With appropriate programming, such a display could literally be the Wonderland into which Alice walked". It was also Sutherland (1968), the "father of computer graphics", who created the first so-called Head Mounted Display (HMD) that provided stereoscopic displays and head tracking. The device called the "Sword of Damocles" was so heavy that it needed to be mounted to the ceiling (Figure 2.2).

Since the "Sword of Damocles", VR has developed into different shapes and forms and it has even started to enter the consumer market recently. There is no unique definition to the term, and depending on author and context, different interpretations can be found. For instance, Coates defined VR as "electronic simulations of environments experienced via head-mounted eye goggles and wired clothing enabling the end user to interact in realistic three-dimensional situations" (Coates, 1992) and thus restricts VR to specific hardware. Others, such as Streuer define VR more hardware independent as "real or simulated environment in which a perceiver experiences telepresence" (Steuer, 1992), where Telepresence

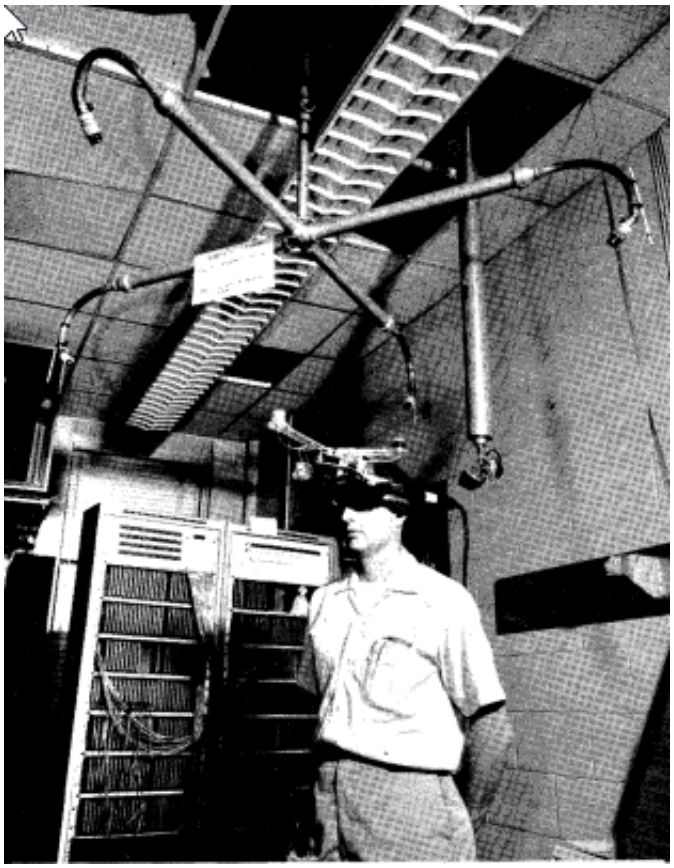


Figure 2.2: Ivan Sutherland's Sutherland (1968) "Sword of Damocles"

is referred to as the experience of presence in an environment by means of a communication medium. The interpretation of VR for this thesis follows the second, more general definition. Further, there are also authors that differentiate between different types of VR. For instance, Robertson, Card, and Mackinlay (1993) define immersive VR similarly to Coates while introducing non-immersive VR that "places the user in a 3D environment that can be directly manipulated, but it does so with a conventional graphics workstation using a monitor, a keyboard, and a mouse". To bridge the gap between immersive and non-immersive, the term semi-immersive has also appeared in the literature (for instance by Mujber, Szecsi, and Hashmi (2004)). This usually describes VR setups that go beyond the typical monitor, keyboard and mouse trio but does not reach the full immersion. In order to clarify the differences between those different types of VR, the next section will provide an overview of VR related Hardware.

2.1.2 VR Hardware

There is a great number of hardware components that can be used for a VR system. In general, three types of devices are required: computation, interaction, and display. While the interaction devices capture user input, the display device depicts the virtual environment to the user. The computation device is responsible for reacting on user input and updating the representation of the virtual environment in real time.

Interaction Devices

If the user can interact with a VR system in a natural way, it increases the level of immersion. Therefore, natural user interfaces (NUIs) that track user movements are often used for VR. This includes a variety of devices that track different body parts with different accuracy. To name a few examples (illustrated in

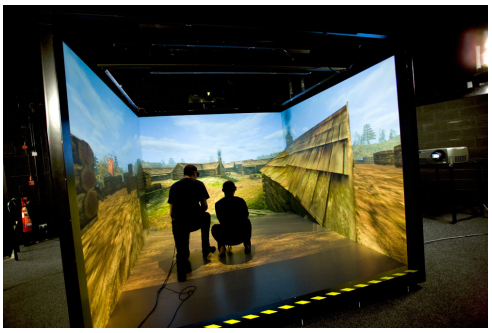
(a) "Leap Motion [Online image]" (n.d.)

(b) "Thalmic Labs Myo [Online image]" (n.d.)

(c) "Microsoft Kinect [Online image]" (n.d.)

(d) "HTC Vive Controller [Online image]" (n.d.)

Figure 2.3: VR interaction devices



(a) “CAVE Automatic Virtual Environment [Online image]” (n.d.)

(b) “Oculus Rift [Online image]” (n.d.)

Figure 2.4: VR immersive display devices

Figure 2.3), there is the Leap Motion[1] for Hand and finger tracking, The Myo wristband from Thalmic Labs[2] for hand- and arm gesture tracking, the Kinect camera from Microsoft[3] for full body tracking and, last but not least, flysticks that are controllers with buttons tracked in 3D space.

Besides tracking devices, other traditional interaction devices such as joysticks, gamepads, keyboards, and mice can be used as well. Finally, there are VR systems that utilize highly specialized interaction hardware. Flight simulators for pilot training for instance (Figures in Section 2.1.3) make use of an accurate replication of an entire airplane cockpit.

Display Devices

Since vision is our primary sensory channel, the depiction of the virtual environment has a great influence on the degree of immersion achieved by a VR system. The best effect is created by stereoscopic 3D surround displays, that adjust the picture according to head movements of the user (head-tracking). The most prominent examples of this type of displays are Cave Automatic Virtual Environment (CAVE) (Cruz-Neira, Sandin, & DeFanti, 1993) and, as mentioned before, Head-Mounted-Display (HMD). While a CAVE is a room that the user stands in with back-projected walls as displays, an HMD is a device that the user wears on his or her head with displays placed directly in front of the eyes. Unlike the “Sword of Damocles”, modern HMDs are lightweight and do not require additional fixation on walls. In both cases, trackers are used for determining the user’s current gazing behavior. For CAVE, usually extra goggles have to be worn while for HMD, the trackers are directly built in. The following Figure 2.4 illustrates these display devices.

Besides these stereoscopic 3D surround displays, multiple screens, spanning a smaller visual angle can be used to achieve semi-surround displays (e.g. Nvidia 3D Vision Surround[4] Figure 2.5. Although these type of displays do not cover the entire field of vision and thus lack behind in terms of provided immersion, there are also benefits. For instance, by retaining a part of the real world in the field of vision, it reduces the risk for motion sickness (Schwebel, Gaines, & Severson, 2008) and accidental collision with physical objects.The following picture shows an example of a semi-surround display. Finally, for

[1] https://www.leapmotion.com, accessed 19-04-2018
[2] https://www.myo.com, accessed 19-04-2018
[3] https://developer.microsoft.com/de-de/windows/kinect, accessed 19-04-2018
[4] http://www.nvidia.com/object/3d-vision-surround-technology.html, accessed 19-04-2018

Figure 2.5: “Nvidia 3D Vision Surround [Online image]” (n.d.)

non-immersive VR, also a standard monitor can be used as a display device. Although this is the least immersive option, it is also the cheapest and most common.

Computing Devices

The computing device for VR needs to provide sufficient calculation and rendering power to generate an authentic representation of the virtual environment that reacts to user input and is updated at a high frequency to create the illusion of real time. Depending on the interaction and display device as well as the complexity of the environment, the computing device can consist of a single computer or, for instance for CAVEs, a cluster of computers. The trend in recent years made VR-Hardware more affordable, more capable and at the same time more user-friendly. Nowadays high-end smartphones have already sufficient computing and graphics power to drive and display VR applications.

2.1.3 VR Applications

The software of VR is equally important as the hardware in creating realistic, immersive virtual environments. In fact, designers of computer games have been developing complex and lively worlds that bring players to spend a great amount of time in them with non-immersive VR. On the one hand side, VR is not able to fully replicate reality. On the other hand side however, VR is also not bound by the physical laws and restrictions of the real world. This opens up a great number of possibilities for VR-Software, and several use cases are described later. In general, VR-Applications have to provide some fundamental features in order to create an immersive experience. Since most VR-Applications focus on graphics, and thus the visual sensory channel, it is necessary to generate images at a high frame-rate to create the illusion of real time. According to Gigante (1993), the frame rate should not be less than 15 frames per second (fps), and 30 fps or more would be desirable. In addition to generating the visual representation (or rendering) of the virtual environment, it does also need to react to user interaction in a close to real-time rate as well. This poses hard constraints on software development. In order to achieve the high framerate, the computation time between two frames is strongly limited. Fortunately, computers are nowadays powerful enough to squeeze a great amount of calculation within this short frame and thus allows the creation of rich VR-Applications.

(a) “Full Motion Flight Simulator [Online image]” (n.d.) (b) “Virtual Aviation [Online image]” (n.d.)

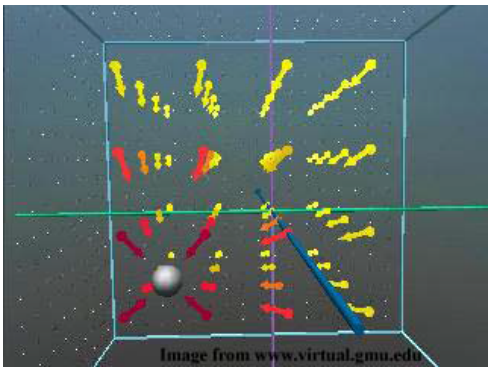
Figure 2.6: Modern VR flight simulator

VR in Education

Using VR for educational purposes brings numerous possibilities to enhance learning and training in many different domains. For instance, it is possible to use the power of VR to create training environments that are too dangerous, too costly or too uncontrollable in the real world. In fact, such systems have been developed since the 80s in the form of military flight simulators for pilot training. The development of such systems have contributed significantly to the development of VR in general (Furness III, 1986) and are nowadays an important part of pilot training. Figure 2.6 shows a modern VR flight simulator as an example.

Following this early example, VR training has been expanded to other domains as well. This includes for instance driving cars, training correct behavior in emergency situations (e.g. (Chittaro, Buttussi, & Zangrando, 2014)), medical simulators for surgeons performing laparoscopy (e.g. (Grantcharov et al., 2004)) as well as many other domains. Pedestrian safety, the example domain for this thesis, has also been trained in VR and is discussed in more detail in Section 2.3. Although these use cases already clearly indicate the value of VR for education, it does not stop with “only” simulating the real world. Instead, VR provides a rich set of possibilities to create artificial learning worlds for more abstract domains. Winn (1993) promotes VR as “the best and probably only strategy that allows students to learn from non-symbolic first-person experience”. He describes three concepts of VR that real-world classroom education does not offer: Size, Transduction, and Reification. Size describes the possibility to make a student experience the learning content at a different scale. On one extreme the student could walk through and interact with cells for biology, on the other extreme virtual learning environments that show the entire solar system at once are also possible. Transduction, on the other hand, describes the possibility to make things perceivable that are not perceivable for humans in the real world. This includes for instance infrared light, the orientation of a magnetic field or the echolocation of bats. Further, Reification stands for creating something that has no physical form, such as population dynamics. In addition to these concepts, VR does also allow the manipulation of time. This could enable students to experience life in another epoch or to see the growth of plants and animals in time lapse. The following Figure 2.7 illustrates some other examples of VR in education.

(a) “LapSim System [Online image]” (n.d.)



(b) MaxwellWorld magnetic-field simulator (Youngblut, 1998)

(c) Aviation safety training
(Chittaro, Buttussi, & Zangrando, 2014)

Figure 2.7: Other VR education use cases

Other Uses

For the sake of completeness, a brief overview of other use cases beyond education is provided here. The biggest sector nowadays and driving force of the recent rise of VR is certainly entertainment and especially the gaming industry. The popular gaming platform Steam[5] has introduced SteamVR and OpenVR as runtimes for game developers and popular gaming engines such as Unity3D[6] or the Unreal Engine[7] provide functions and features that facilitate the development of VR Games. This enabled the development of the SafeChild application of this thesis with low human and monetary budget. By the time of writing this thesis, there are already over 300 game titles on the Steam platform. Besides entertainment, other uses of VR also include architectural and urban design, therapy, theme parks, arts, etc.

In summary, VR is an exciting technology that offers the possibility to create realistic virtual environments that has matured through half a century of research and development. With all the possibilities of VR and the fact that it is entering the consumer market, the future of this technology is promising and opens up opportunities for all kinds of domains.

[5] https://store.steampowered.com, accessed 19-04-2018
[6] https://unity3d.com, accessed 19-04-2018
[7] https://www.unrealengine.com, accessed 19-04-2018

2.2 Intelligent Tutoring Systems

2.2.1 General Overview

Generally speaking, Intelligent Tutoring Systems (ITS) are a special class of software for Computer-Aided Instruction (CAI) that is capable of adaptation. Instead of providing the same content to everyone, the goal of an ITS is to optimize the teaching and learning process for each individual learner according to their learning characteristics and needs. This includes their state of knowledge, learning goals, misconceptions or even their affective state. Nwana describes ITSs as "computer programs that are designed to incorporate techniques from the AI community in order to provide tutors which know what they teach, who they teach and how to teach it" (Nwana, 1990). Corbett formulates the goal for ITSs as "to engage the students in sustained reasoning activity and to interact with the student based on a deep understanding of the students' behavior" (Corbett, Koedinger, & Anderson, 1997). The origins of ITSs can be traced back now for over 40 years. The first system to pioneer the idea of using Artificial Intelligence (AI) for tutoring in CAI was SCHOLAR by Carbonell (1970) for teaching South American geography. The term ITS was coined later by Burton and Brown (1976) who introduced the term "intelligent computer-based tutors" and also made the first attempt to combine ITS and game-based learning for teaching elementary level mathematics.

The motivation for the research and development of ITSs lies in the fact that individual one-to-one lecturing by human tutors can lead to significantly higher learning achievements as compared to teaching in classroom settings. The prominent study by Bloom (1984) showed that the difference in examination scores could be as much as two standard deviations. In a one-to-one tutoring setting, the teacher can adapt to the pace of the learner and make sure that the content is well understood before moving on to the next chapter. In addition, motivation, rewards, feedback, and instructions can be provided at the right time and the right form to benefit learning. However, human teachers are rare, costly and thus not available to the broad mass of learners. Therefore, ITSs attempt to enable some of these benefits from one-to-one tutoring in an automated way through computers.

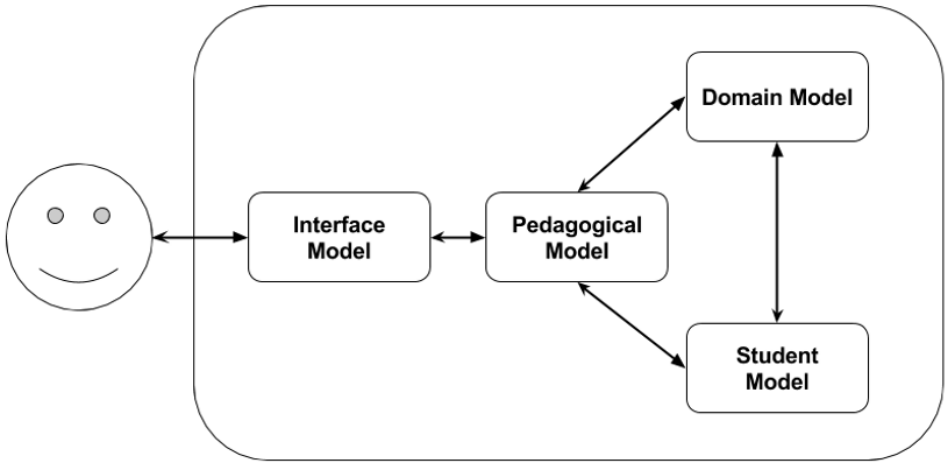


Figure 2.8: General ITS architecture (modified from (Nwana, 1990))

2.2.2 Architecture

Typically, an ITS consists of four major components: Domain Model, Student Model, Pedagogical Model and Interface Model (Corbett et al., 1997; Nwana, 1990). These components are sometimes named slightly differently in literature but describe the same concepts: The Domain Model is responsible for defining "what to teach", and thus, the content and structure of domain skills and knowledge. The Student Model represents the current state of learning of an individual student. It could for instance be a subset of the Domain Model with the learning content that the student has already mastered, but also include additional information such as, history of exercises, demographic data, learning goals and affective state (such as frustration). Further, the Pedagogical Model defines "how to teach". For example, a Pedagogical Model could consist of rules that describe when and in which form to provide instructions and feedback or how to select the next exercise based on the outcome of the current exercise. The last but equally important Interface Model defines how the ITS interacts with the student. This includes the problem-solving environment and how to process and interpret student input and provide feedback.

From a bird's eye view, the interaction between these four major components can be described as follows: first, the user interacts with the ITSs through the Interface Model. Then, his or her actions are interpreted by the Pedagogical Model according to the rules and concepts defined in the Domain Model. Based on the interpretation results, for instance whether the user made a mistake or successfully completed a task, the Student Model is updated accordingly. Finally, the Pedagogical Model decides based on the information provided by the other components whether and which pedagogical actions should be taken. This process is illustrated in the following Figure 2.8.

Depending on factors such as domain, target user group, and interface, the actual implementation of these components vary greatly. While some domains, such as mathematics and physics are clearly structured and defined, other domains, such as legal argumentation, are more ill-defined. Thus, different models are required to describe the domain content and correctness of answers. Furthermore, an ITS that is designed for the training of university students might need to provide very different forms of motivation compared to an ITS for elementary school children. The hard- and software interface do also influence the design of the ITS. While a standard desktop application relies on a keyboard and a mouse, mobile devices make use of touch and rotation. VR, as described above, comes with a large number of special interaction and display devices. Over the years, a great variety of ITSs have been developed for many kinds of use cases, and it would exceed the frame of this thesis to present a complete overview. Instead, the following sections introduce two of the most influential types of ITS as well as several other ITSs that makes use of VR.

2.2.3 Cognitive Tutors

Cognitive Tutors are a prominent group of ITSs that are based on the Adaptive Control of Thought - Rational (ACT-R) theory (Anderson, 1996). According to ACT-R, cognitive knowledge can be divided into two kinds of representations: declarative knowledge and procedural knowledge. On the one hand, declarative knowledge (also called chunks in ACT-R) is knowledge about facts, such as definitions or historical events. On the other hand, procedural knowledge encodes how to apply declarative knowledge to tasks or situations in order to achieve the desired outcome. For instance, knowing and remembering the distributive property in its formulation would be declarative knowledge and applying it correctly to transform a mathematical expression would be procedural knowledge. In ACT-R and in Cognitive

Tutors, procedural knowledge is represented as a set of production rules that have the form of IF - THEN statements. A production rule for addition of fractions could, for example, be:

IF goal is addition of two fractions with unequal denominators

THEN set subgoal to finding a common denominator

Using this kind of production rules, Cognitive Tutors are able to act as an expert system and generate step-by-step solution paths to a problem as sequences of production rule applications. By adding so-called buggy production rules, the system can also consider typical errors made by learners. An example of a buggy rule could be:

IF goal is addition of two fractions with unequal denominators

THEN individually sum up nominators and numerators

With correct and buggy rules, the tutoring system is able to create a generic Student Model that incorporates all states that can occur during problem-solving. When a learner works on an exercise in a Cognitive Tutor, an individual Student Model is created based on the learner's actions. By comparing the individual and the generic Student Model, Cognitive Tutors are able to determine at which point of the solution path the learner currently is. Thus, the system knows whether the learner is on the right track and what the next actions should be to reach the goal. This process of comparing learner and expert solution is called Model-Tracing. With this information, the system can provide feedback specific to the current state of problem-solving and give next-step hints if the learner is stuck. It also knows if the learner has made a typical error represented by a buggy rule and can react accordingly. In some of the Cognitive Tutors, e.g. the LISP tutor, another mechanism is implemented to track learner actions called Knowledge Tracing (Anderson, Corbett, Koedinger, & Pelletier, 1995). Upon observing the correct application of a production rule, the system does not conclude directly that the student has mastered the skill. Instead, probabilities for guess, slip and initial learning are considered to calculate a probability for mastering the skill using a Hidden Markov Model (HMM). Cognitive Tutors are considered as one of the most successful groups of ITSs and have been developed for different domains including the Geometry Tutor, the Algebra Tutor, and the LISP Tutor. In these domains, Cognitive Tutors were able to significantly improve learning results. For instance, an improvement of more than one standard deviation on test outcomes was measured for the Geometry Tutor, and learners using the Lisp Tutor were able to complete exercises 30% faster and 43% better compared to peers using a standard LISP environment. However, Cognitive Tutors also have limitations. The specification of production rules to cover as much as possible of the solution space is a difficult and time-consuming task. If a rule is missing, or if the student deviates from one of the predetermined solution paths, the tutor considers it as an error. Therefore, such tutors are especially hard to design for domains with many solutions to the same problem. Nevertheless, Cognitive Tutors with principles of Model and Knowledge Tracing are certainly the most influential class of ITSs.

2.2.4 Constraint-Based Tutors

Constraint-Based Tutors are based on the Constraint-Based Modeling (CBM) approach by Ohlsson (1994). Contrary to Cognitive Tutors, where cognitive skills are encoded in the form of production rules, CBM defines a number of constraints that the learner must not violate. Instead of analyzing each

step and comparing them to states in the solution space through Model Tracing, the CBM approach is only interested in the current state of problem-solving. As long as no fundamental principles of the domain are violated, the student is considered to be on the right track. While defining the constraints, it is also important to specify in which context the constraint applies to. Therefore, each constraint in a Constraint-Based Tutor is defined by a pair of two conditions: the relevance condition C_r and the satisfaction condition C_S. At every step of problem-solving, C_r is checked first, and if the condition is met, C_s will be checked subsequently. An error is detected if C_r is satisfied and C_s is not. An example given by Ohlsson and Rees (1991) for the domain of Algebra is as follows:

$C_r : \frac{(x+y)}{d}$ is given to the answer to $\frac{x}{d_1} + \frac{x}{d_2}$

$C_s : d = d_1 = d_2$

Constraint-Based Tutors have also been developed for a number of domains including the SQL Tutor (Mitrovic, 2003), CAPIT - an ITS for punctuation and capitalization in English (Mayo, Mitrovic, & McKenzie, 2000), KERMIT - an ITS for database design (Suraweera & Mitrovic, 2002) and NORMIT - an ITS for data normalization (Mitrovic, 2002). Studies by Mitrovic (2003) show the effectiveness of Constraint-Based Tutors and indicate an improvement of 0.5 - 0.75 standard deviations in exam scores for learners using the SQL Tutor.

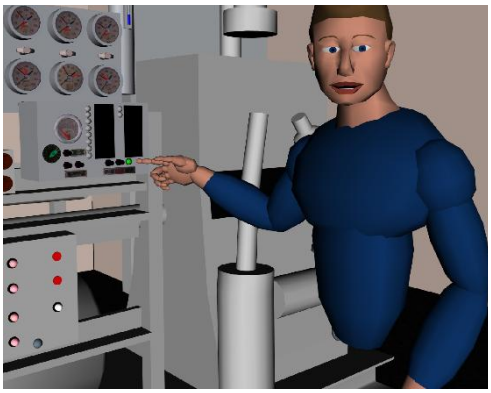
Cognitive Tutors and Constraint-Based Tutors both have their pros and cons, and a comparative analysis has been reported in the past (Mitrovic, Koedinger, & Martin, 2003; Kodaganallur, Weitz, & Rosenthal, 2006). Amongst the main differences are the implementation effort and also the supported form of instructions and feedback. While Cognitive Tutors are generally more difficult and time-consuming to implement due to designing and testing a large set of production rules, they can provide next step hints and understand typical errors due to the inclusion of buggy rules. Constraint-Based Tutors on the other hand are generally faster to implement, but lack the ability to provide next step hints and can only point out missing or erroneous elements of a solution. A Constraint-Based Tutor can also be more effective in domains with large solution spaces and many different solution strategies, were Cognitive Tutors tend to struggle with. For either type of ITS it is possible to extend or reduce the set of features at the cost of implementation effort. In summary, both types of ITS have proven to be feasible, and it depends on a domain, target features, and resources available whether a Cognitive Tutor or a Constraint-Based Tutor is more appropriate.

2.2.5 VR Tutors

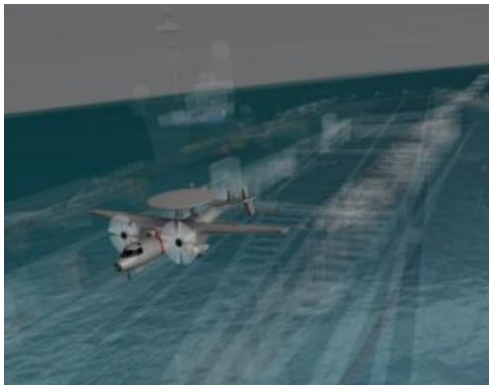
A combination of both of the above technologies can either be seen as an ITS that uses a VR based Interface Model or a VR learning environment that is extended by adaptive tutoring capabilities. Such systems, called VR-ITS from here on, have the potential to incorporate and join the benefits of both technologies to create tutoring systems that are not only highly realistic but can also provide assistance, feedback, and instructions based on learner needs. Besides the capabilities to realistically simulate and visualize learning content, VR offers several other possibilities. For instance, a virtual teacher or pedagogical agent could inhabit the virtual environment with the student and either collaborate or assist the learner. Hereby, the pedagogical agent can provide interaction in a natural, non-verbal way by utilizing gestures and mimic (Johnson & Rickel, 1997). In addition, VR provides the opportunity for pedagogical experience manipulation and stealth tutoring (H Chad Lane & Johnson, 2008). Through simulation alone,

VR already provides implicit feedback that mimics real-world feedback. For instance, if a golf ball is hit with a certain speed and force in VR, it could either enter the hole or land elsewhere. If the ball does not reach the hole, the learner knows that more force needs to be applied without the need for an explicit hint. This experience of implicit feedback can be manipulated by the ITS to optimize learning. An example of pedagogical experience manipulation can be found in an ITS that teaches intercultural awareness in VR (H. Chad Lane, Core, Gomboc, Karnavat, & Rosenberg, 2007). In this system, virtual characters overreact to errors of the learner, such as mentioning a taboo subject. Stealth tutoring on the hand goes beyond the manipulation of simulation rules and adds events in the environment that guides the learner in an indirect way. For instance, instead of providing a direct instruction, a virtual character could casually mention task-relevant information in a dialogue with the learner (Mott & Lester, 2006).

In addition to these benefits, VR also brings new ways and challenges for tracking and modeling the student. The Interface Models of the Cognitive and Constraint-Based Tutors mentioned above all utilize a standard desktop setting. This means keyboard and mouse as input devices and standard desktop user interfaces (UIs) consisting mostly of elements such as buttons, textfields, and checkboxes. By using such an interface, the learner's input is well defined. At each point of problem-solving, the system is aware of what input to expect and how it should be analyzed. Furthermore, the learner needs to submit an answer through an explicit interaction, and thus the system knows exactly when to evaluate the input. An Interface Model that uses VR on the other hand needs to cope with its real-time nature. On the one hand side, this means that the system receives user input continuously including information about movement



(a) STEVE (Johnson & Rickel, 1997)



(b) PEGASE in Gaspar (Buche, Bossard, Querrec, & Chevaillier, 2010)

(c) ELECT BiLAT (H. Chad Lane, Core, Gomboc, Karnavat, & Rosenberg, 2007)

(d) Military training (Livak, Heffernan, & Moyer, 2004)

Figure 2.9: VR-ITS examples

and gaze. On the other hand side, this also means that this information needs to be filtered and interpreted at an equally high frequency. The challenge is further amplified since VR itself is already computation intensive and thus the remaining calculation power for the ITS is limited. Especially for such domains that allow open movement and where movement is an essential part of problem-solving, the ITS needs to stay vigilant at all time.

A number of ITS have explored the opportunity of using a VR based Interface Model. Amongst the first of such systems is STEVE (Soar Training Expert for Virtual Environments) (Johnson & Rickel, 1997), an ITS for operating complex devices such as the gas turbine propulsion system aboard US Navy surface ships (Figure 2.9a). STEVE utilizes Model Tracing similar to the Cognitive Tutors but is build upon the SOAR cognitive architecture instead of ACT-R. Therefore, just as Cognitive Tutors, STEVE can provide next step hints and react on user questions. In addition, STEVE is embodied as a pedagogical agent that is able to demonstrate procedures. Another approach towards VR-ITS that focuses on a generic and domain independent Pedagogical Model is PEGASE (Buche, Bossard, Querrec, & Chevaillier, 2010). Instead of relying on production rules or constraints, PEGASE uses a UML (Unified Modeling Language) based description of tasks. The pedagogical rules are then formulated based on generic relations and events of the task description. For instance, instead of referring to concrete objects, a pedagogical rule could query objects related to the current task. Figure 2.9b right shows the application of the "simplify" rule in Gaspar, an application for training collaborative procedures on an aircraft carrier (Marion, Septseault, Boudinot, & Querrec, 2007). The rule turned the Aircraft carrier transparent so that the learner can focus on the airplane. Limitations of this approach are that human tutors are required for the final selection of the generic pedagogical rules at runtime and that the tasks need to be clearly structured, in order to be representable using UML. Another Standout VR-ITS is the above-mentioned approach by H. Chad Lane et al. (2007) for teaching the ill-defined domain of intercultural awareness in conversations (Figure 2.9c). Instead of classifying actions of the learner in a binary, correct or wrong way, this tutor also considers mixed results. These arise through actions that could be positive towards one, but at the same time negative towards another learning goal. An example would be a sentence during a dialogue that could have positive short-term but negative long-term effects. The classification of learner actions and provision of assistance are realized using an expert model. For these VR-ITS mentioned until now, movement in the Virtual Environment is not a critical part of domain skills and is thus not being accurately monitored by the ITS. On the other hand, correct movement is an integral part of this thesis' domain, pedestrian safety. As H Chad Lane and Johnson (2008) point out, VR-ITSs that precisely track learner movement are very rare and such systems face the challenge of deciding at which granularity movement should be tracked and when the system should react. One example of an open-movement VR-ITS is a system for tutoring soldiers on the use of military operations on urban terrain (MOUT) (Livak, Heffernan, & Moyer, 2004) (Figure 2.9d). For this system, the authors implemented a rule-based expert system that controls virtual teammates on the one hand side (through forward chaining) and diagnoses learner actions on the other hand side (through backward chaining). Movement is considered as a latent action that is only considered after the learner stops so that the path can be determined. Due to the fact that the system performs Model-Tracing at runtime, the authors have expressed concern about runtime when the complexity of the expert system increases. This concludes the overview of VR-ITS and, to the best of my knowledge, the most prominent practices and concepts of this small but interesting field of research. In summary, the potential and benefits of VR-ITS are well known, and a number of such systems have been developed.

Most of them target specialized domains with small user groups and thus, their effectiveness are not as well studied as non-VR ITSs such as Cognitive or Constraint-Based Tutors. While most VR-ITS make use of rule-based expert systems and Model-Tracing, this method could face limitations in a dynamic open movement domain, when this costly algorithm has to be run at high frequency.

2.3 Child Pedestrian Safety Education

2.3.1 General Overview

As stated in Chapter 1, pedestrians, especially children, are endangered traffic participants. Therefore, educational measures for teaching child pedestrians how to be safe in different traffic situations are crucial to prevent injuries. In fact, pedestrian safety education has a long-standing history in both research and practice and has been recommended in many countries, including high, middle and low-income countries (Duperrex, Bunn, & Roberts, 2002). In order to become safe pedestrians, children have to learn and train a number of concepts and skills, of which some are especially challenging for children. This includes for instance crossing the road at different types of crossing, identifying safe places to cross and to observe traffic from different directions. A more in-depth analysis of these skills and their structure are given in Chapter 4. Schwebel et al. (2014) conclude that "pedestrian safety is not a unique and single action but rather a multilayered complex cognitive-perceptual-motoric task". As such, pedestrian safety cannot be trained through classroom education alone but requires practical training of behavior and skills. Ideally, this training should be repeated until safe behavior becomes an automatism (Percer, 2009). It is also important, to design training in a way that suits the state of development and the special needs of children. In their elaborate, systematic review and meta-analysis of behavioral interventions to improve child pedestrian safety, Schwebel et al. (2014) included 25 studies spanning 5 decades and conclude that, although behavioral interventions are effective in general, there is still much room for improvement and further research. Besides children, pedestrian education programs also exist for other groups of people, including elderly and people suffering from certain disabilities. These groups, however, are beyond the scope of this section and thesis.

2.3.2 Forms of Training

There are many different forms of child pedestrian safety education. This includes different group sizes including classroom, small groups, and individualized education, as well as different instructional approaches including film and video, board games and training in sport halls or real roads. In the age of personal computers, specialized software has also emerged for the purpose of child pedestrian education. This includes puzzle games, multiple-choice and other quiz games and also pedestrian simulators where children are able to control a virtual avatar in a virtual environment. Virtual Reality training systems, such as the one presented in this thesis, are a particularly realistic subgroup of such simulators and are presented in more detail after introducing the challenges of child pedestrian education. Figure 2.10 shows examples of different forms of child pedestrian safety education. An additional important form is the training by parents to supplement school programs and to ensure the repetitive training and application of safe behavior in the long run. However, parents should not be let alone with this task, since they tend to overestimate the abilities of their children (Percer, 2009).

(a) Roadside training (WikiHow, n.d.) (b) Indoor training (Mike Koozmin, 2014)

(c) Izzy's road safety games (SDERA, n.d.)

Figure 2.10: Forms of pedestrian safety education

2.3.3 Challenges

As mentioned before, being a safe pedestrian is a challenging task for children due to a number of different reasons. First, their cognitive, perceptual and motor abilities are limited due to their state of mental and physical development. Crossing roads, especially without any regulations such as traffic lights or zebra crossings, is highly demanding with regard to these abilities. Children need to learn how to observe traffic from multiple directions, estimate speed and distances of incoming vehicles, keep their state in mind while looking the other side and analyze all this information with their own walking speed to determine when it is safe to cross the road. In addition, children aged 4-9 struggle to pay selective attention to those parts of traffic that are important for their safety while ignoring the other parts Hill, Lewis, and Dunbar, 2000. Further, it is difficult for them to see the traffic situation from a global perspective Underwood, Dillon, Farnsworth, and Twiner, 2007 and thus to understand that there are also hidden dangers. In experiments by Kwame Ampofo-Boateng and Thomson (1991), it has been shown that children aged 5-7 exclusively rely on visible presence of cars to determine whether a certain place is dangerous to cross. Factors such as obstacles obscuring the view are not recognized as a threat by them. As stated by Percer (2009), these difficulties can be related to developmental theories including Piaget's theory of cognitive development (Ginsburg & Opper, 1988) and Vygotsky's zone of proximal development (Shabani, Khatib, & Ebadi, 2010). According to Piaget's theory, children in the critical age of 5-9 are either in the pre-operational (2-7 years old) or the concrete operational stage (7-11 years old). One limitation of the pre-operational state, for example, is the inability to take another person's perspective, which translates into the difficulty of seeing traffic from a global perspective. In the concrete operational stage, children still struggle with abstract thinking which makes it difficult to apply learned skills to novel situations and to grasp concepts such as speed and timing. According to Vygotsky, the stages of development are however not bound strictly to a certain age but can be influenced by the surrounding society. For instance, Ampofo-Boateng

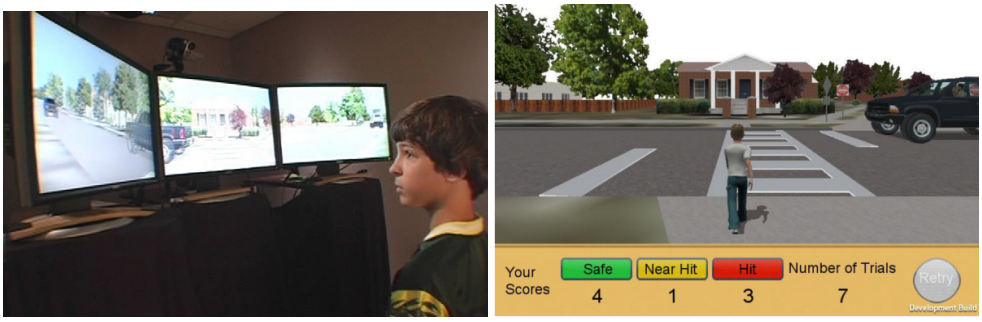
et al. (1993) showed that trained 5 year-olds are able to identify safe places to cross on the same level as the 11-year-old control group immediately after training and still on the level of 9-year olds 2-8 months later. Besides the challenges related to their state of development, children also lack experience and knowledge in traffic, which makes it difficult for them to react to unexpected situations or emergencies.

An effective training program needs to consider all these challenging factors. Because of the individual rate of cognitive development and differences in personality, small group or individualized training are favorable (Schwebel et al., 2014). In addition, since it might be too challenging for children to fully understand the traffic situation from a global perspective, skills have to be trained repetitively and in different situations to install an automatic reaction in children (Percer, 2009). For instance, even if children cannot fully grasp the dangers of crossing between parked cars, they can learn to recognize the situation and be trained to avoid them. However, this kind of training programs would typically require a large number of tutors and time spent on real roads, making them costly and time-intensive and thus hard for broad implementation in many schools or public education settings.

2.3.4 Pedestrian Safety Education in Virtual Reality

Virtual Reality is a promising alternative to training on real roads that is largely independent of location and time. In addition, it enables the creation of arbitrary traffic situations and reduces the need for human personnel to ensure children's safety during training. Further, it allows children to train individually and to make decisions by themselves, not based on examples of others. A number of previous studies have investigated the use of Virtual Reality (VR) as a tool for practical child pedestrian training.

In a study by McComas, MacKay, and Pivik (2002), 95 children from 4th to 6th grade attending urban and suburban schools went through a VR intervention that was intended to teach several pedestrian safety skills: walking on the sidewalk versus walking on the street; stopping at the curb; looking left-right-left before crossing and staying attentive while crossing the street. The VR system used consisted of a semi-immersive setup of three 21" computer monitors placed in a semicircle and a head tracking device to measure head movement. Thus, children could look left and right in a natural way by turning their head. The software guided the children through eight intersections with different types of crossing scenarios. This includes different traffic signs, number of lanes and distractions such as noise or other pedestrians. Although this requires more than the aforementioned skills, only those four were monitored. A feedback component was also included in the software that prompts children to stay on the sidewalk, stop at the curb and to look Left-Right-Left. Each child went through the software three times individually: pretraining,



(a) Semi-immersive VR environment (b) Desktop VR environment

Figure 2.11: Virtual road crossing environments of Schwebel et al. (2008, 2014)

during training and posttraining. Feedback was provided to the child only during a training session. The evaluation was performed by comparing pre- and posttraining performance within the system as well as actual roadside behavior, for which research assistants observed children crossing real roads. The results showed that children were able to significantly improve within the VR application, but only the children from suburban schools were able to transfer the improvement into the real world. The software of this study appears to be similar to that of this thesis as it offers a number of different crossings and includes a feedback component. However, the VR system and program were described only on a high level, and additional or pictorial information could not be found. Therefore it is unclear whether feedback was generated automatically by the system (and thus some type of Student Modeling was performed), how movement interaction was implemented and whether the training sessions relied on human or system feedback (according to Schwebel and McClure (2010), feedback was provided in verbal form by an experimenter). Nevertheless, this study it is a great example for the potential and diversity of VR for child pedestrian education.

Various other approaches focused solely on improving the skill of finding appropriate gaps in traffic for unregulated crossings. This includes a study by J. A. Thomson et al. (2005) with 94 children aged 7, 9 and 11. The VR system used a standard monitor and a single on-screen button for interaction. The software showed different busy roads from an elevated, semi-aerial viewpoint such that sufficient vision to both directions was provided from a single static viewport. The task was to help a virtual character to cross a number of roads by observing traffic and deciding when to cross by clicking the button. The program would then take over, simulated the crossing and give three different feedbacks: success, collision or tight fit. The trainers of this study were mothers of the participating children, and the training procedure consisted of four sessions of 30-40 minutes, held at roughly weekly intervals. The results of this study include that children crossed faster, missed less safe opportunities and improved in the estimation of time needed to cross.

A series of research by Schwebel et al. (2008, 2010, 2014) also target the skill of finding gaps in traffic for crossing. First, they investigated the general feasibility of a semi-immersive VR setup for training this skill with 102 children ages 7-9 and 74 adults (Schwebel et al., 2008). As in McComas' study, three monitors in a semicircular arrangement were used as display to show a moderately busy two-lane road (Figure 2.11a). The average traffic volume was 11.92 vehicles/min for adults and 10.65 vehicles/min for children, and the traffic speed was set to 30 miles/h (approximately 48.3 km/h). The objective was to observe traffic through by looking left and right and to decide the right moment for crossing. Unlike Thomsen et al., the interaction is triggered by a physical pressure plate that is activated by stepping on it. This simulates the actual crossing behavior of stepping from curb to road. After stepping on the plate, the virtual word changed from first- to third-person perspective, and the participant could observe a virtual character cross the road.

The evaluation utilized questionnaires and measurements to determine the validity of the VR system. The results indicate that the environment was perceived as very realistic and almost no symptoms of motion sickness were reported. Further, adults performed better in the system than children and behavior was in general similar to the real world. Thus the authors conclude that the system is valid as a tool to research and train children in pedestrian safety. This VR system was later used for a randomized controlled trial among 231 7-8-year-olds to compare the effectiveness of individualized street-side training, VR training, and training using videos and websites. The measured variables include hits and close calls,

(a) User with Eye-Trackers in front of the dome (b) Different crossing scenarios

Figure 2.12: VR dome projection of traffic scenarios (Oron-Gilad, Meir, Tapiro, & Borowsky, 2011)

attention to traffic and missed opportunities before, immediately after training and six months later. The result shows that VR training and individualized street-side training outperform the other variants with VR being the more cost-efficient method. In an effort to make their VR system more portable and more accessible, Schwebel, McClure, and Severson (2014) are also working on an online version of their system that can run in a browser and a single monitor (Figure 2.11b). In this setting, mouse movements are used to look both sides, and the space bar key replaces the pressure plate for interaction. The results show that the system offers a usable and feasible environment for pedestrian safety training and that children remained engaged and attentive during training.

Other uses of VR related to child pedestrian safety include a study Oron-Gilad, Meir, Tapiro, and Borowsky (2011) on deficits in hazard perception using a Dome Projection Facility (180 degrees spherical screen, Figure 2.12) as display and an eye-tracking device to record eye-movement data. The task and interaction are again determining an appropriate gap to cross (or in this case not to cross at all) and to indicate the decision by pressing a button. However, several parameters of the environment changed during the course of the study including the presence of zebra crossings, the presence of vehicles and restriction of the field of view. The results from 46 participants with 21 adults and 25 children showed that children were less aware of hazards, especially with regard to the limitations of the field of view in the form of road curvatures or parking cars.

Based on the above-mentioned cases and other similarly successful studies (e.g. Bart, Katz, Weiss, and Josman (2008); Congiu et al. (2008)), the value and potential of VR for child pedestrian safety have been clearly demonstrated. However, there is still much untapped potential of VR. Most of the current systems focus on the single skill of finding a right gap to cross, provide only very basic feedback and are restrictive in terms of interaction. A look into the entertainment industry unveils the potential of current computer technology to create realistic and complex artificial worlds with engaging open environments that could allow more authentic and diverse training. In addition, the current systems still rely heavily on human tutors to perform the teaching task which is still keeping the cost of VR training high and availability low. This could be changed through the use of an Intelligent Tutoring Systems (ITS) that provides individually tailored and autonomous training. This thesis contributes a step towards this next generation of VR training systems that could hopefully be assimilated into broad public education in the future and eventually lead to a positive effect on child pedestrian safety at a large scale.

Chapter 3

Behavior Tree based Tutoring

As described in Chapter 2, Intelligent Tutoring Systems (ITS) that use Virtual Reality (VR) have access to several special tutoring techniques such as embodied pedagogical agents, implicit feedback and stealth tutoring that standard user interfaces do not offer. On the other hand, such systems require Student Modeling methods that are compatible with VR's freedom of action and real-time nature. Current VR-ITSs use Model Tracing with rule-based expert systems, Constraint-Based Tutoring (CBT) or UML activity diagrams to model tasks and to evaluate learner performance (see Section 2.2). Although these methods have been successfully applied in previous work, the overall amount of VR-ITS and their target domains is still highly limited. One group of domains that is particularly hard for Student Modeling in VR are those with ill-defined tasks in dynamic environments, which includes the target domain of this thesis, pedestrian safety training. This chapter first introduces the characteristics of such domains and the challenges to current Student Modeling methods. Subsequently, a novel Domain and Student Modeling approach based on Behavior Trees will be presented as an alternative.

3.1 Ill-Defined Tasks in Dynamic VR Environments

One of the key challenges when developing an ITS for a novel domain is to solve the so-called Student Modeling problem (VanLehn, 1988). In general, this includes understanding the learner's actions within the system, putting them in context with skills and learning goals of the target domain and to evaluate their correctness. The target domain of this thesis, child pedestrian safety, is an unexplored domain for ITSs and combines two features that make Student Modeling difficult. First, the training environment is highly dynamic due to a necessity to simulate realistic traffic conditions. The state of cars and traffic signals are constantly changing and affecting the safety of pedestrians and the correctness of their actions. For instance, from one moment to another, a safe gap in traffic for crossing a road could become too narrow.

Second, the tasks of the domain are ill-defined. The definedness of a learning domain is not an absolute value and instead has many gradations. A categorization introduced by Mitrovic, Suraweera, Martin, and Weerasinghe (2004) separates the definedness of a domain itself from the definedness of it's tasks. On one end, well-defined domains with well-defined tasks (WDWI) include adding fractions in mathematics or balancing chemical equations in chemistry. On the other end, legal argumentation, essay writing and artistic design fall into the category of ill-defined domains with ill-defined tasks (IDIT). In between these extremes, lies the category of well-defined domains with ill-defined tasks (WDIT). Pedestrian safety

training falls into this category. Although there are well-defined rules to regulate traffic and behavioral patterns for safe road crossing behavior, the task of transferring and integrating this theoretical knowledge to actual situations is ill-defined.

Amongst others, the definedness of a task is dependant on start and goal state and the existence of multiple correct solutions (e.g. (Lynch, Ashley, Aleven, & Pinkwart, 2006; Mitrovic et al., 2004)). The general task of the pedestrian safety domain of walking safely from one place to another satisfies the conditions for being ill-defined with regard to these aspects. First, the start state of an ill-defined task does not provide sufficient information to derive the complete solution. At the start of a pedestrian journey, even if the person knows the destination and the way to get there, a number of unknown variables remain. This includes, for instance, the position of parked cars or other sight-blocking objects, the state of traffic lights, and the amount of traffic on the roads. All these factors could influence the actual path and sequence of actions that are required to complete the task. Second, reaching the goal state of an ill-defined task does not guarantee the correctness of the entire task execution. Registering the fact that a pedestrian arrived at the destination is not sufficient to judge the safety of the journey. In fact, the pedestrian could have ignored all safety rules and still arrived unharmed - by luck. Last but not least, an ill-defined task can have more than one correct solution. Indeed, there can be multiple different ways to get from one point to another safely, and it could be hard to decide on the best option. For instance, it is difficult to compare the safety of zebra crossings and traffic lights or to decide whether it is safer to cross at a small number of unregulated crossings versus a large number of regulated crossings. Besides pedestrian safety, there are other domains with similar characteristics including safe cycling or driving. In addition, the same holds for different kinds of sport, where there is a fixed set of rules and strategies, but the actual actions to win a specific game are ill-defined and dependent on the behavior of other players. Due to these similarities, the challenges and the novel Student Modeling approach described below are worth considering for these domains as well. This is however beyond the scope of this thesis.

3.2 Student Modeling Challenges

In order to perform Student Modeling for ill-defined tasks in dynamic environments, there are several requirements. This includes the capability to accept multiple correct solutions, a suitable model for the dynamic environment, frequent updates (due to continuous input and output of a VR interface) and mechanisms to handle randomized variables that are not known before the start of an exercise (for traffic safety this could include car speeds and traffic density to increase realism and diversity of the exercise). These preconditions are difficult to handle with conventional Student Modeling methods used in VR-ITSs. In the subsequent sections, the arguments for and against each of the current Student Modeling method will be provided.

3.2.1 UML Activity Diagram

In a UML activity diagram (used, for instance, by Buche et al. (2010)), nodes are used to represent states, activities, and decisions. In addition, connectors between these nodes, for instance, in the form of arrows, are used to define the control flow between nodes (Rumbaugh, Jacobson, & Booch, 2010). In general, the UML activity diagram provides an intuitive graphical representation for the structure of a task. With appropriate tools, this structure can also be changed on the graphical level by reordering and reconnecting

nodes. This form of visual programming enables designers without technical background to define and adapt task models. The problem with this method and ill-defined tasks in dynamic environments is that every transition and decision between states and actions need to be explicitly drawn. With the existence of multiple solutions and dynamic factors, this leads to a large number of nodes and edges, which eventually results in a cluttered diagram that is hard to read. This would reduce the advantage of visual readability and increase the effort required to design, modify and test such a task model. To illustrate the problem, the following Figure shows two example activity diagrams for a part of the road crossing process. In order to cross the road, the pedestrian needs to go to a suitable place to cross and, depending on the type of crossing, perform respective preparatory actions. In a straightforward approach, three decision nodes and four process nodes could be used to model this (first diagram of Figure 3.1). The problem with this approach is that it does not capture the case where a pedestrian decides to abort crossing at one place to cross somewhere else. This could, for instance, happen if s/he notices a better option to reach the

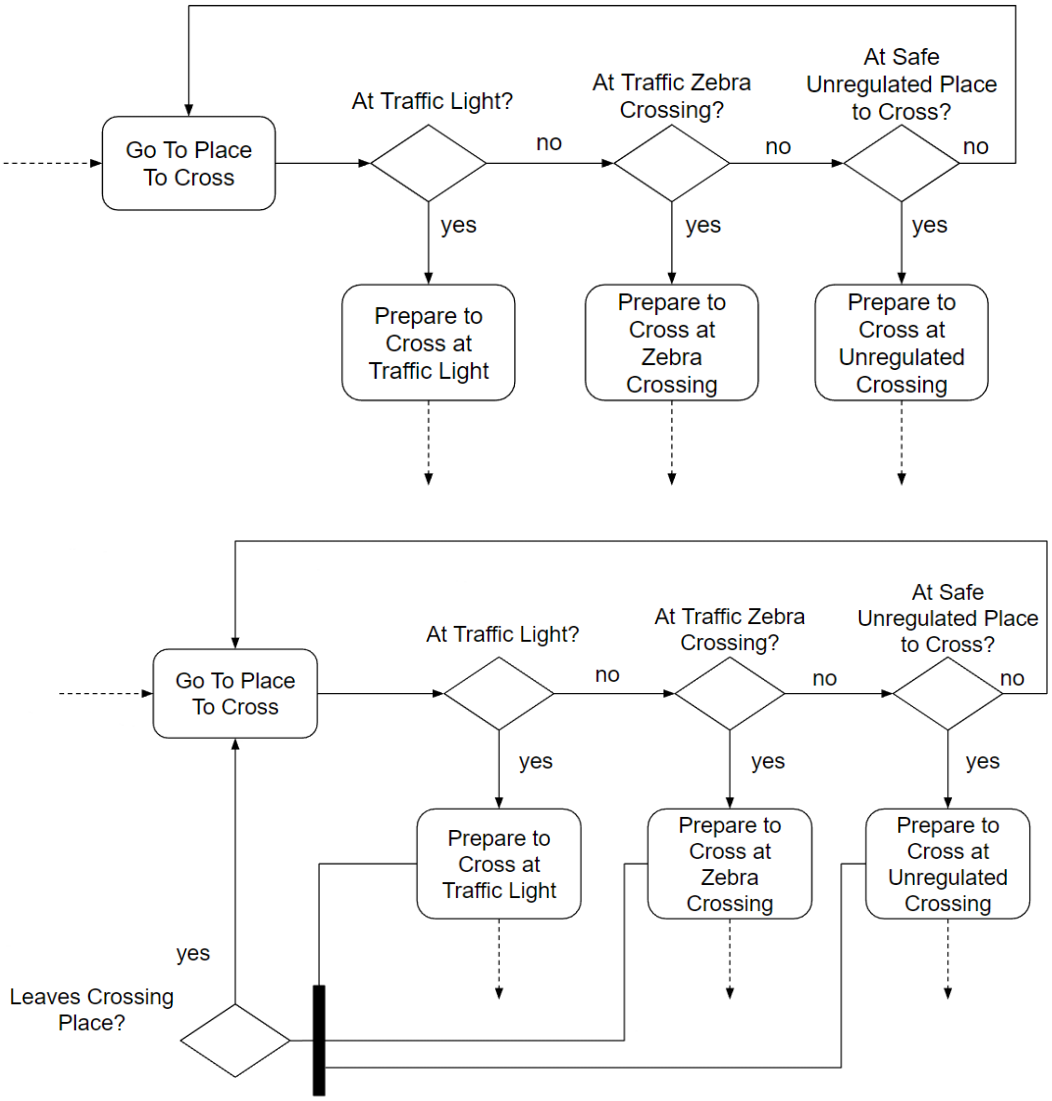


Figure 3.1: Activity UML diagrams for preparing to cross a road. The first picture shows straight forward model while the second model shows the enhanced version with backward links.

destination safely. In order to model this option, the model would need additional nodes and edges. As a result, the second diagram of Figure 3.1 becomes more complex. If erroneous actions would also be considered, the complexity of the diagram would grow even further.

3.2.2 Rule-Based Expert System

A rule-based expert system makes use of a so-called knowledge base to model the task and was used for instance by Johnson and Rickel (1997) for their VR-ITS. The knowledge base consists essentially of facts and IF - THEN rules (also called production rules) about the target domain. At runtime, the IF parts of the rules are matched against the current situation and, if a fitting rule can be found, then the action of the THEN part of that rule can be executed. If more than one rule matches the situation, a conflict resolution strategy needs to be applied to select a rule to execute. In contrast to the decision nodes of an UML activity diagram, the IF part of the rules can combine multiple variables and form complex formulas. Thus, these rules have high expressive power and a small number of well-designed rules can specify appropriate actions for a large number of different situations. This property is advantageous for dynamic environments and helps in building concise task models. However, this approach does not provide a native visual presentation. Thus, rule-based models are more difficult to design and debug. Especially for ill-defined tasks, creating and testing rules for all situations, including buggy rules, can be laborious and very time consuming (Mitrovic et al., 2003). Further, since rule matching can be computationally expensive and a VR needs to update at high frequency, the run-time of the expert system could potentially become an issue (see Section 2.2.5). An example rule set for the pedestrian crossing preparation task mentioned above is listed in Table 3.1 If a pedestrian would leave an appropriate place to cross without crossing, the system would notice that an action was executed that does not match any rule. The decision on how to interpret this case is up to the designer. In any case, the situation *Pedestrian not at crossing* would be restored, and the system can use its rules to guide the user's next actions.

IF	**THEN**
Pedestrian at traffic light	Prepare to cross at traffic light
Pedestrian at zebra crossing	Prepare to cross at zebra crossing
Pedestrian at unregulated crossing	Prepare to cross at unregulated crossing
Pedestrian not at crossing	Go to Place to cross

Table 3.1: Example rule-set for pedestrian crossing task

3.2.3 Constraint-Based Tutoring

Constraint-Based Tutoring (CBT) has been proposed as a suitable approach for WDIT domains (Section 3.1). It has been used in a VR-ITS for prospective memory training (Mitrovic et al., 2014). Instead of specifying correct actions for different situations, it tries filtering out incorrect actions using constraints. With other words, CBT models the boundaries of the solution space instead of paths through it. The constraints in CBTs consist of two conditions. A relevance condition that specifies when the constraint applies and a satisfaction condition that must not be violated if the relevance condition is met. By using CBT, the task modeling process is simplified in comparison to the above mentioned methods, especially for WDIT domains. On the other hand, the resulting model is also less expressive and precise. For

instance, it lacks the ability to determine which action to perform next and thus cannot be used to provide next-step hints. There is also no straightforward visual representation of the CBT task model. Table 3.2 shows example constraints for the pedestrian crossing preparation task.

Relevance Condition	**Satisfaction Condition**
You stepped on the road at a traffic light	You have performed the traffic light preparation before
You stepped on the road at a zebra crossing	You have performed the zebra crossing preparation before
You stepped on the road at an unregulated crossing	You have performed the unregulated crossing preparation before
You stepped on the road anywhere else	You made a mistake

Table 3.2: Example constraints for pedestrian crossing task

Conclusion

As a conclusion for this section, UML activity diagrams, rule-based expert systems, and CBT all bring benefits on the one hand but also face limitations on the other when dealing with ill-defined tasks in dynamic VR environments. Thus, an alternative Student Modeling method was developed for the VR-ITS of this thesis, which will be introduced in the remainder of this chapter. It is inspired by the concepts of all above mentioned methods and has the goal to join some of the individual advantages while reducing the drawbacks. The key concept of this approach is to map the learner's actions to the state of a Behavior Trees, and thus, it is given the name Behavior Tree Tracing (BTT). Chapter 4 is dedicated to modeling the child pedestrian safety domain using BTT.

3.3 Introduction to Behavior Trees

The term Behavior Tree is used in two different contexts. Originally, Behavior Trees emerged as a formalism for the process of requirement engineering during the development of complex software. In this context, they are defined by Dromey (2003) as "... a formal, tree-like graphical form that represents behavior of individual or networks of entities which realize or change states, make decisions, respond-to/cause events, and interact by exchanging information and/or passing control". In order to distinguish between the two types of Behavior Trees, this first type will be referred to as Requirement Engineering Behavior Trees (REBTs). The purpose of REBTs is to facilitate the translation of natural language requirements into formal specifications. It captures information that is traditionally spread across different other models in software engineering. This includes sequence diagrams, activity diagrams, class diagrams, and state machines.

The second type of Behavior Trees originated as Artificial Intelligence (AI) control model for non-player characters (NPCs) in modern video games (e.g. Isla (2005)). For the remainder of this thesis, the term Behavior Tree with its abbreviation BT should be associated with this second type of Behavior Trees. Similarly to REBTs, BTs are of tree-like graphical form, but instead of defining the behavior of a software system, they define the behavior of an AI agent. Each node in a BT represents a behavior that can be either elementary or composite. Elementary behaviors are the leaf nodes in a BT. This includes, for

instance, primitive actions such as performing a movement or retrieving specific information about the environment. Composite behaviors, on the other hand, are the inner nodes of a BT and have one or more sub-behaviors. At runtime, each node of a BT can be in one of four states: *Running*, *Success*, *Failure* and *Inactive*. When a BT is being executed, the states of its nodes change along with the actions performed by the agent and the dynamics of the environment. The general semantics behind the states are as follows: a node in *Running* state represents a behavior that is being carried out at the moment. *Success* and *Failure* indicate that the behavior was either completed successfully or failed. Finally, *Inactive* behaviors are not relevant to the current situation. The state of an elementary node is calculated directly based on interaction with the environment. For instance, the elementary task of moving forward five meters is in state *Failure* if there is a wall in front of the agent, *Running* if there is no wall but the agent is still walking and *Success* if the five meter goal is reached. Composite nodes, on the other hand, determine their state based on their type and the state of their child nodes. The most prominent composite nodes are Selector, *Sequence*, *Repeat*, and *Parallel*.

Selectors (also called *Priorities*) (Figure 3.2) evaluate their child nodes consecutively according to their order and enters the *Success* state upon finding the first child node in *Success* state. It only enters *Failure* state if all child nodes failed. This resembles the process of trying out different solutions to one problem and only finally giving up if there are no options left.

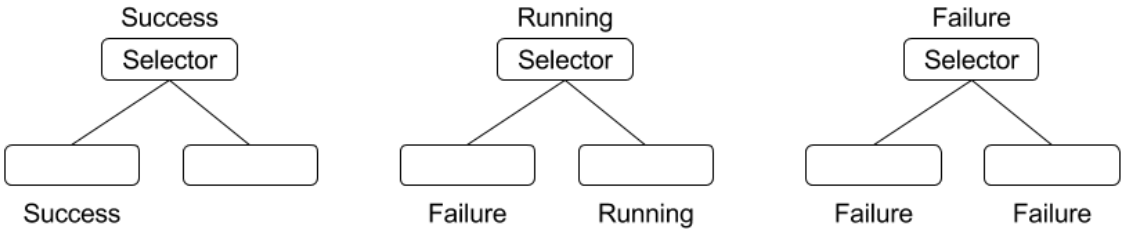


Figure 3.2: *Selector* configurations

Sequences (Figure 3.3) enter *Failure* state as soon as one of the child nodes are in *Failure* state on only enters Success state if all child nodes succeeded. *Sequences* are used to represent tasks where the sub-steps are dependent on the success of its predecessor.

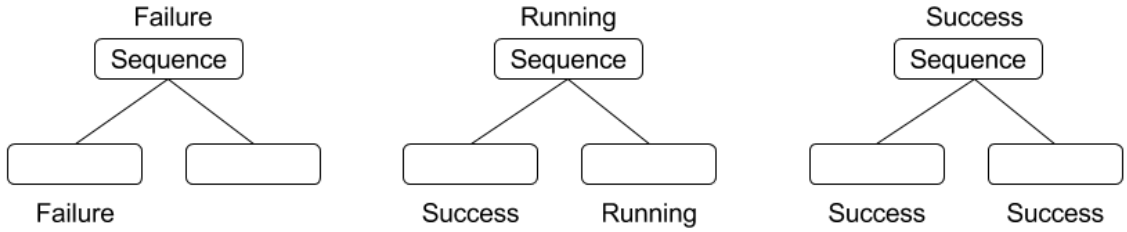


Figure 3.3: *Sequence* configurations

Repeat (Figure 3.4) will reset its child node to Running if it fails. The repeat node succeeds if the child node succeeds. In addition to the child node state, other conditions (for instance depending on environment variables) could be used as well to decide how long a behavior should be repeated.

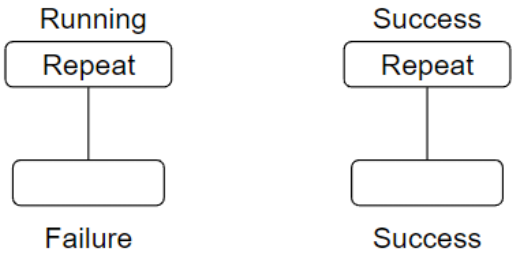


Figure 3.4: *Repeat* configurations

Parallel (Figure 3.5) sets its child nodes simultaneously to *Running* when it starts to run itself. It fails if one of the child nodes fails and succeeds if all child nodes are in *Success* state. Otherwise, it will stay in *Running* state. The order of child nodes is not relevant here, and variations to the *Success* and *Failure* rules are also possible

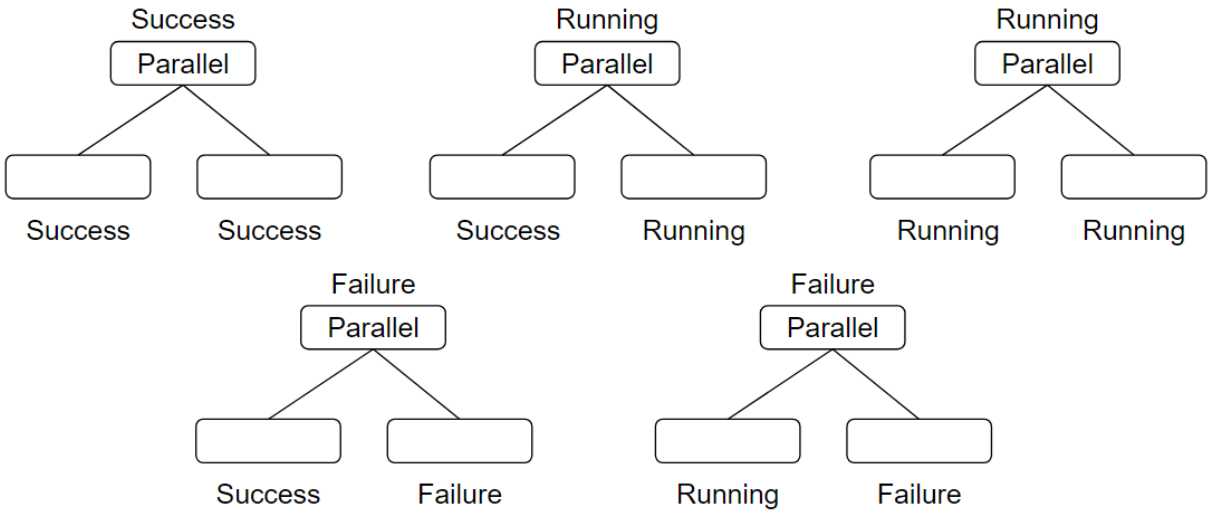


Figure 3.5: *Parallel* configurations

Using *Repeater*, *Selector* and *Sequence*, the following BT (Figure 3.6) can be constructed for the simplified crossing preparation task of Section 3.2. A detailed explanation of the structure will be provided in Chapter 4. Essentially, the BT expresses that the pedestrian needs to go to a place to cross and then select one of the preparation tasks. In addition, the whole process needs to be repeated until the pedestrian succeeded in reaching a place to cross and performing one of the preparation tasks.

In order to use a BT for Student Modeling, two key modifications are required. First, for the nodes at the elementary level, a mechanism needs to be added that checks the learner's execution state of that action. Second, a method for checking the correctness of task execution is necessary for error tracking. These modifications are presented in the following formalization.

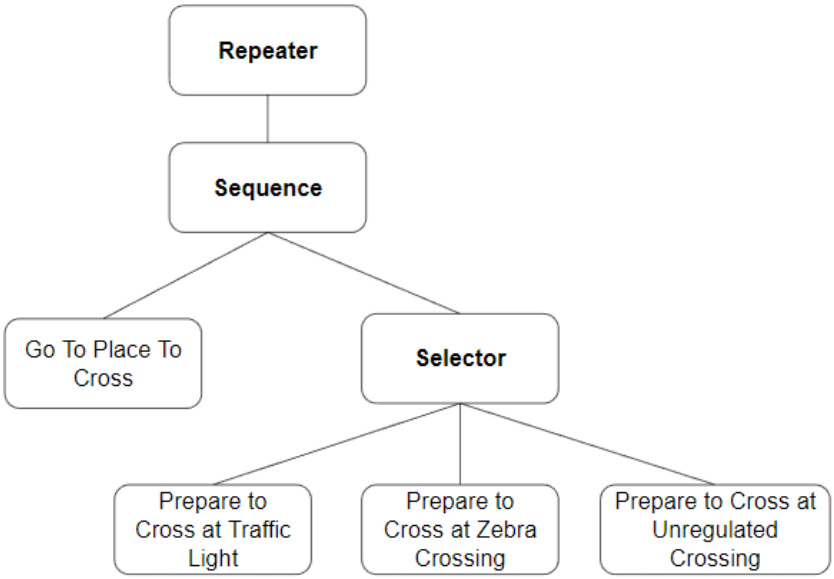


Figure 3.6: BT representation of crossing preparation task

3.4 BTT Formalization

Born as a model and tool from the domain of video games, BTs were not supplied with a formal mathematical framework (Isla, 2005). Since BTs have recently gained popularity in various scientific domains, including robotics (Marzinotto, Colledanchise, Smith, & Ögren, 2014) and Unmanned Aerial Vehicle (UAV) control systems (Ogren, 2012), there have been approaches to formalize BTs for these respective use cases. There is, however, to the best of my knowledge, no previous work on using BTs for Student Modeling. Thus, a novel formalism will be introduced for this purpose. This formalization for Behavior Tree Tracing (BTT) uses notations from propositional logic, set notation, and graphs, with the goal to strike a balance between clarity and accuracy. This formalization should not be considered as a rigid definition but rather a conceptual basis for both design and implementation of BTT. Depending on the individual needs of a domain, it can and needs to be further extended or optimized.

3.4.1 BTT Task Model

The BTT task model BTT_{TM} represents the rules, dynamics and relevant objects of a task (It can be interpreted as the (of part of the) Domain Model of a traditional ITS architecture) using three different layers that are built upon each other. A set of propositions about the virtual world in which training takes place, a set of conditions based on those propositions and a special type of BT called Expert Behavior Tree. Thus, the BTT_{TM} can be formally defined as a triple consisting of these elements.

Definition 1 (BTT Task Model)**.** Let W be a world space (Definition 2), C be a condition set (Definition 3) and EBT be an Expert Behavior Tree (Definition 4). Then a BTT task model BTT_{TM} is defined as the following triple.

$$BTT_{TM} = (W, C, EBT)$$

World Space

The world space W is the base layer of a BTT_{TM} and serves as the interface between the virtual world and the tutoring system. It consists of the basic properties and facts of the environment that are relevant for the task performance in the target domain. For this basic formalization, W will be considered as a set of abstract proposition that could, for instance, represent "The sun is shining", "The traffic light is green" or "The learner is on the road".

Definition 2 (World Space)**.** The world space W of a BTT_{TM} is defined as a finite set of propositions $W = \{P_1, \ldots, P_n\}$, where each proposition has a valuation to either true or false at any time during the task performance.

Condition Set

The second layer, the condition set C is used to represent constraints and rules of the target domain. They are designed in a task-oriented way based on W. For instance, they define under which circumstances a task or sub-task should be considered erroneous, completed or aborted. An example condition for preparing to cross at a traffic light is "has learner made sure that oncoming cars have stopped". This combines propositions about nearby cars, their speed and the trajectory of the learner's head movement. Each condition can be interpreted as a well-formed formula (wff[1]) based on propositions from W. There is no general rule whether relevant information from the training environment should be modeled as basic propositions or conditions. The design goal should be to achieve a high degree of re-usability for the world space propositions.

Definition 3 (Condition Set)**.** Let W denote the world space of a BTT_{TM} . Then, the condition set C of the same BTT_{TM} is a set of formulas:

$$C = \{c_1, ..., c_n\}, \text{ where } c_i : \text{is a } wff \text{ based on propositions from } W \text{ for all } 0 < i \leq n$$

Expert Behavior Tree

The Expert Behavior Tree EBT represents the topmost layer of a BTT_{TM}. It defines the hierarchy of tasks and subtasks using the same principles as presented in Section 3.2. Thus, there are also different types of nodes in an EBT. The essential difference is that nodes in an EBT can be annotated with different types of conditions. In a standard BT, a behavior finishes with either success or failure. Semantically, it represents whether the behavior or task was completed or could not complete. While this information is sufficient to control an AI agent, it is not expressive enough to model a human learner. He or she might complete a task but in an erroneous way. In such a case, it would not be a success in terms of mastering the task. For instance, the learner could reach the other side of the road and complete the crossing task but forget to monitor traffic while crossing. At the same time, aborting an action by choice is not necessarily a failure or evidence of a misconception. The learner could have chosen an alternative but also correct solution, which is perfectly possible for ill-defined tasks. As a result, it is not sufficient to only define conditions for completing or aborting a task. Instead, additional constraint conditions are required to describe the circumstances of correct task execution. From this point, completion and abort will be used

[1] For the sake of simplicity, common formation rules of propositional logic apply.

for the original BT concepts of success and failure to avoid confusion with correct and incorrect task performance.

Definition 4 (Node and Condition Types)**.** The node types N_{type} and condition types C_{type} of an EBT are defined as sets with the following members.

$$N_{type} = \{Sequence, Selector, Parallel, Repeat, Elementary\}$$
$$C_{type} = \{Abort, Constraint, Complete\}$$

Definition 5 (Expert Behavior Tree)**.** The Expert Behavior Tree EBT of a BTT_{TM} is the triple $EBT = (N, E, \gamma)$ where N denotes the set of nodes, E denotes the set of edges in the graph and γ is a condition assignment function. The tuple (N, E) forms a directed acyclic graph. Let N_{name} be a finite set of unique node names. Then, each node is represented by a tuple of its name and type. Edges are represented by triples consisting of parent node name, child node name and child index. The child index is unique per parent and used to determine the order of child nodes. Finally, γ assigns a set of typed conditions to a node. Thus, given a condition set C, the spaces of the EBT elements can be formally written as follows.

$$N \subset N_{name} \times N_{type}$$
$$E \subset N_{name} \times N_{name} \times \mathbb{N}$$
$$\gamma : N_{name} \to \mathbb{P}(C \times C_{type})$$

The amount and type of the conditions assigned to a node and the number of child nodes are defined by their type. The system described in this thesis uses the configuration as shown in Table 3.3 below.

N_{type}	# of Abort Conditions	# of Complete Conditions	# of Constraint Conditions	# of Children
Selector	0	0	0 or more	2 or more
Sequence	0	0	0 or more	2 or more
Parallel	0	0	0 or more	2 or more
Repeat	0	0 or 1	0 or more	1
Elementary	0 or 1	1	0 or more	0

Table 3.3: Condition assignment for different node types

Graphical Notation The following graphical notation (Figure 3.7) will be used to visualize EBTs. Nodes are represented by rectangles consisting of five segments. The left square shows the node type using different symbols. The right upper square contains the node's name, and the three right lower squares show the conditions assigned to the node. From left to right, the boxes are intended for $Abort$, $Constraint$ and $Complete$ conditions. If a node does not contain any conditions, the bottom squares are omitted for the sake of readability. In addition, arrows are used to represent edges. The order of child nodes is indicated by their horizontal position, increasing from left to right.

3.4.2 Example

To illustrate the BTT formalization, the following example shows a BTT_{TM} for the task of delivering a letter to a mailbox with a cover and two openings A and B (Figure 3.8). The natural language description

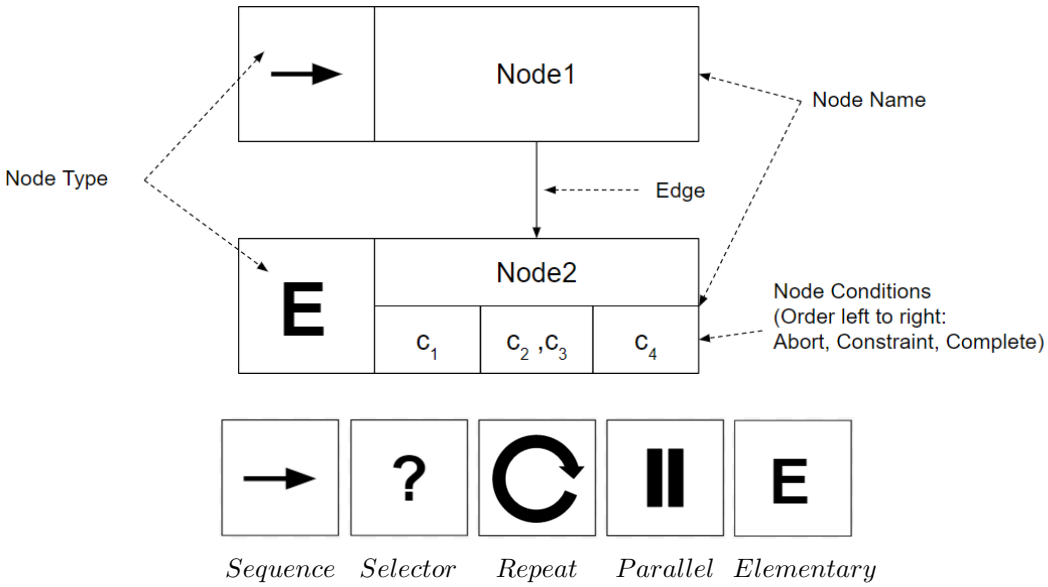


Figure 3.7: Graphical notation of EBTs

shall be written as follows: “First, open the cover. Then, depending on the recipient written on the letter, throw it either in opening A or B. Finally, close the cover”. Based on this description, the world space W, in this case, could consist of the following propositions in Table 3.4. Then, the conditions of C are defined as the formulas in Table (3.5).

Propsition	Semantics
$CoverClose$	The cover of the mailbox is closed
$CoverOpen$	The cover of the mailbox is open
$InspectedLetter$	The user read the recipient address
$LetterForA$	The letter is addressed to A
$LetterForB$	The letter is addressed to B
$LetterInA$	The letter was thrown in opening A
$LetterInB$	The letter was thrown in opening B

Table 3.4: Mail delivery World Space properties

Condition	Formula
c_1	$CoverOpen$
c_2	$(LetterInA \rightarrow (LetterForA \wedge InspectedLetter)) \wedge$ $(LetterInB \rightarrow (LetterForB \wedge InspectedLetter))$
c_3	$CoverClosed$
c_4	$LetterInA \vee LetterInB$

Table 3.5: Mail delivery Conditions

With W and C defined, the EBT is defined using the graphical notation in Figure 3.9. It consists of a $Repeat$ root node that keeps the tree running until the letter is inserted, a $Sequence$ node that specifies the order of actions that need to be performed and three $Elementary$ nodes representing basic actions

with their respective conditions. The example will continue with a detailed description of its runtime behavior after introducing the BTT Student Modeling approach.

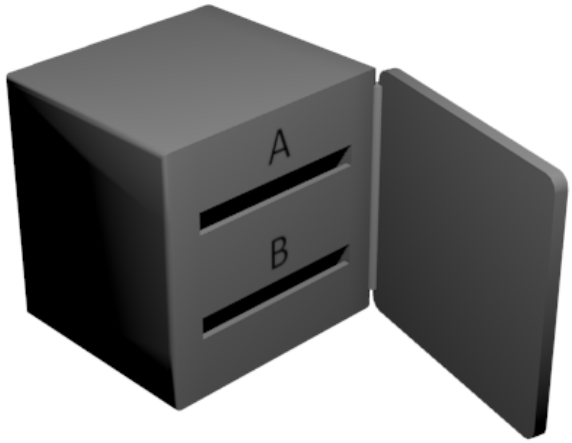


Figure 3.8: Mailbox

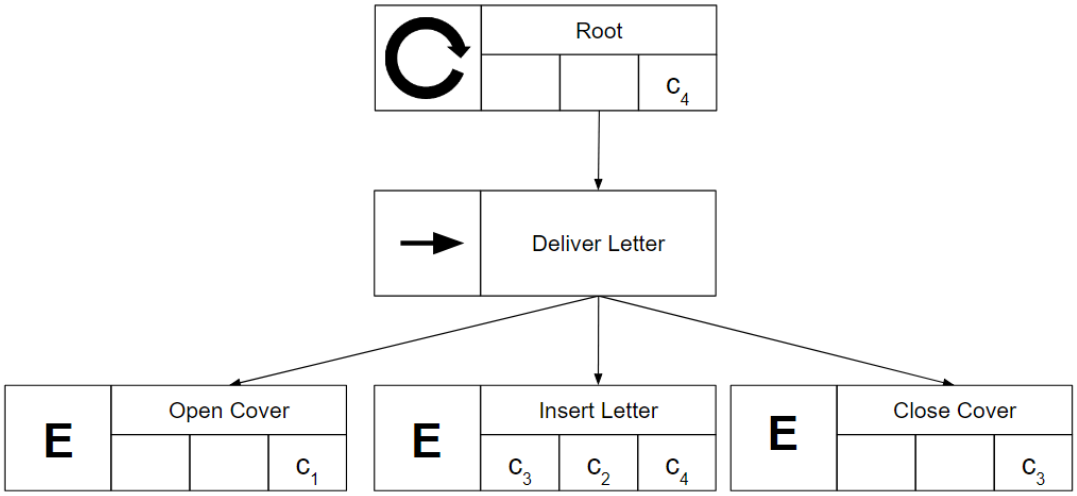


Figure 3.9: Mail delivery Expert Behavior Tree

3.4.3 BTT Student Modeling

Once a $BTT_{TM} = (W, C, EBT)$ is defined, a Student Model can be generated at every time frame during exercise execution. Essentially, this is achieved by assigning states to each node of the EBT and updating them in real-time based on generic algorithms. This Student Modeling process is a modified version of a BTs standard update mechanism (Isla, 2005). In addition to the run state of a node, an additional error state is introduced to track mistakes made by the learner.

Definition 6 (BTT Student Model)**.** Let $EBT = (N, E, \gamma)$ be an Expert Behavior Tree, $t \in \mathbb{N}$ denotes a naturally ordered timestamp during exercise execution and let the state space S be defined as a set of tuples consisting of an error and a run state.

$$S = (S_{error} \times S_{run})$$
$$S_{error} = \{false, true\}$$
$$S_{run} = \{Inactive, Running, Completed, Aborted\}$$

Then, the Behavior Tree Student Model is represented by a set of tuples that correlates nodes with their states at a given time frame. Using $s_t(n)$ as the notation for the state of node n at time t, the Behavior Tree Student Model at time t is defined as follows.

$$BTT_{SM}(t) = \{(n, s_t(n)|n \in N\}$$

Student Modeling Process

The algorithm for generating the $BTT_{SM}(t)$ consists of two phases. In the first phase, a so-called Tick signal is sent through the EBT in a top-down fashion. Starting from the root, each node that receives this signal is requested to update its own state and to return the result. While the error state of a node is directly updated by evaluating its $Constraint$ condition, the run state of the node is determined in a recursive way. Thus, composite nodes will forward the tick signal to their child nodes (the pseudo-code algorithms are provided below). When the Tick signal reaches the layer of $Elementary$ nodes, the second phase of the algorithm begins. Now, the run states of the nodes are updated bottom-up. $Elementary$ nodes update their run state directly through evaluation of its $Abort$ or $Complete$ condition. Whenever a node completes its own run state update, the result is returned to its calling parent. This allows the parent node to update its state as well. The overall algorithm is completed when the root node has updated its run state. This entire process is illustrated in Figure 3.10. It shows that the only interface between the VR simulation and Student Modeling algorithm is the set of relevant propositions defined in W. Apart from that, both processes can run independently from each other. During runtime, this Student Modeling algorithm should be run regularly and preferably at high frequency to enable the system to quickly react to errors or student actions. In practice, it was feasible for the system of this thesis to run the algorithm with each rendering frame, which resulted in a close to real-time update frequency.

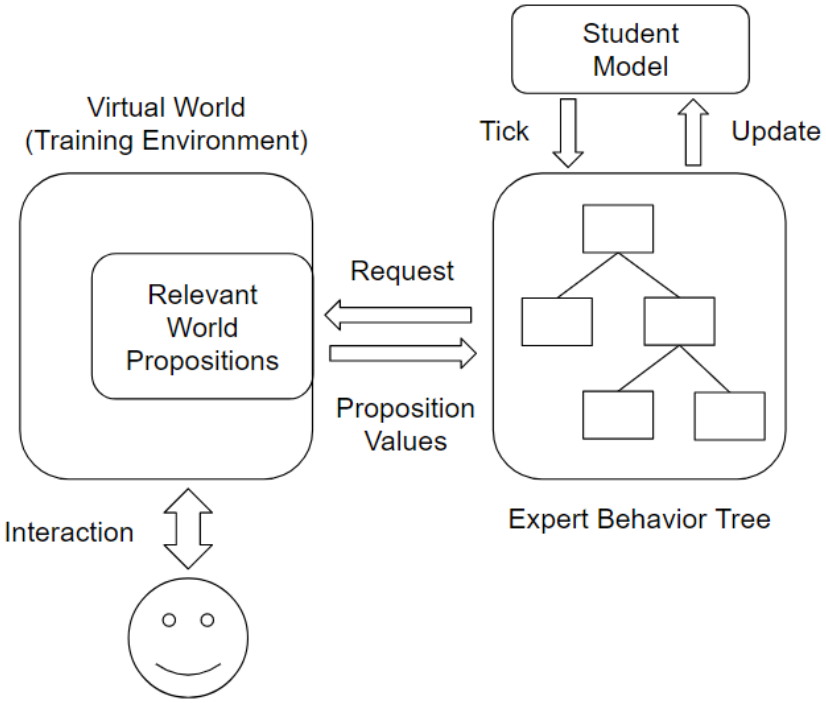


Figure 3.10: Student Model update process

Pseudo-Code Algorithms

Algorithm 1: Tick

```
Tick(node)
node.state ← Update(node,node.type)
if node.state = Running then
    foreach constraint of node.constraints do
        if constraint is violated then
            node.error_state ← Error //invoke appropriate handler
        end
    end
end
return node.state
```

Algorithm 2: Update for *Selector*

```
Update(node,Selector)
set i to 1 or the index of the first child in Running state
while i ≤ index of last child do
    child state ← Tick(child[i])
    if child state = Completed or Running then
        return child state
    else
        i++
    end
end
//if execution reaches this point, all children are in aborted state
return aborted
```

Algorithm 3: Update for $Sequence$

```
Update(node,Sequence)
set i to 1 or the index of the first child in Running node
while i ≤ index of last child do
    child state ← Tick(child[i])
    if child state = Aborted or Running then
        return child state
    else
        i++
    end
end
//if execution reaches this point, all children are in completed state
return completed
```

Algorithm 4: Update for $Elementary$

```
Update(node,Elementary)
if node.abort_condition is satisfied then
    return Aborted
end
if node.complete_condition is satisfied then
    return Completed
end
return Running
```

Algorithm 5: Update for $Parallel$

```
Update(node,Parallel)
tick all children
if there is one child in Running state then
    return Running
end
if one child state is Aborted then
    return Aborted
end
if all child states are Completed then
    return Completed
end
```

Algorithm 6: Update for $Repeat$

```
Update(node,Repeat)
child state ← tick(child node)
if child state = Aborted then
    reset all child node states to Inactive
    reset all child node error states to NoError
    return Running
else
    return child state
end
```

Example Continued

To illustrate the BTT Student Modeling process, the mail delivery example from above will be continued. At the beginning of the exercise (time frame 0), the run state of the *Root* node is set to $Running$ while all other nodes are $Inactive$. The error state of all nodes are set to $false$ (Figure 3.11). The cover of the mailbox is closed, both letter boxes are empty, the letter is addressed to A, and the learner has not inspected the letter yet.

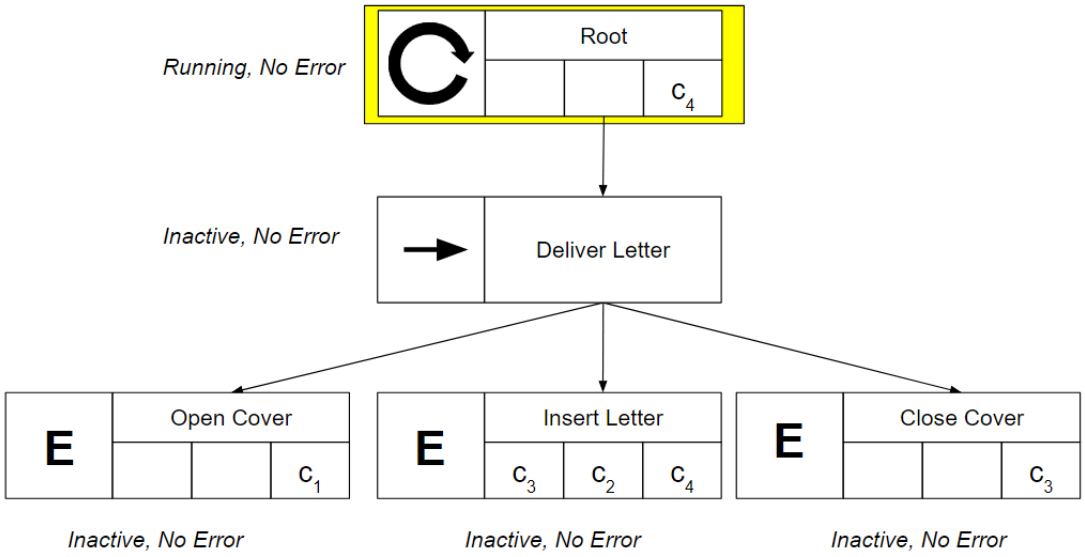


Figure 3.11: Initial Student Model. The yellow frame indicates a $Running$ state.

Immediately after the start of the exercise and the first Tick, the state of the nodes change to the situation depicted in Figure 3.12. Because the end condition is not met for the *Root* node, the *Deliver Letter* node propagates the Tick to its first child, and since none of its conditions is satisfied, it stays in $Running$ state. At this point, the learner is allowed to freely navigate and explore the environment. As long as the cover is not opened, the student model does not change.

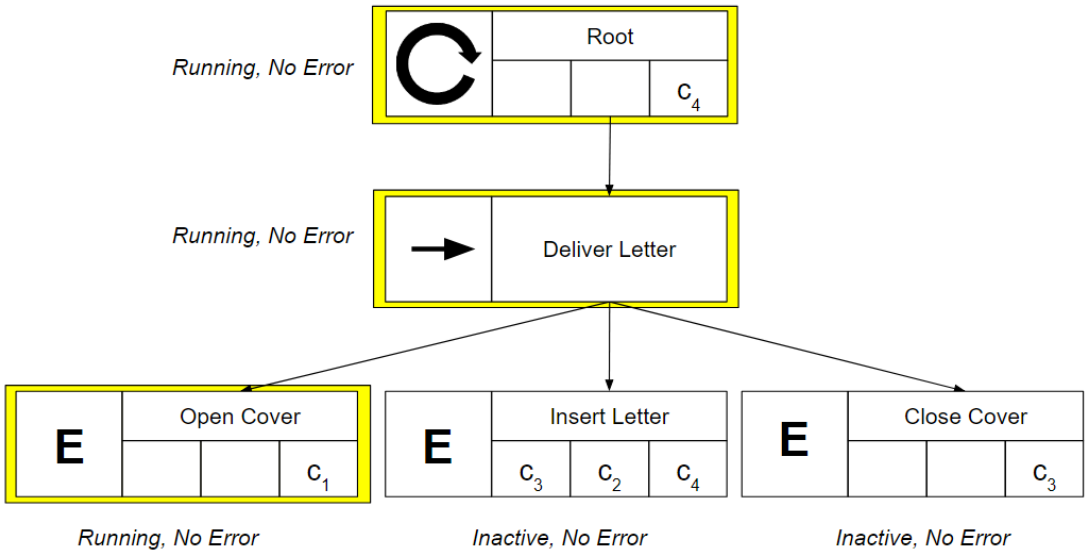


Figure 3.12: Student Model after the first update

If, at some time, the learner decides to open the cover, the states in the tree change again. Now, the *Open Cover* node's $Completed$ condition is met, and thus it changes into the $Completed$ state. Since the parent *Deliver Letter* node is of type $Sequence$, its next child *Insert Letter* will be set to $Running$ state (Figure 3.13). If the cover was open at the beginning of the exercise, the Student Model would turn into this state immediately. This is an example for BTTs flexibility to adapt to variations of an exercise.

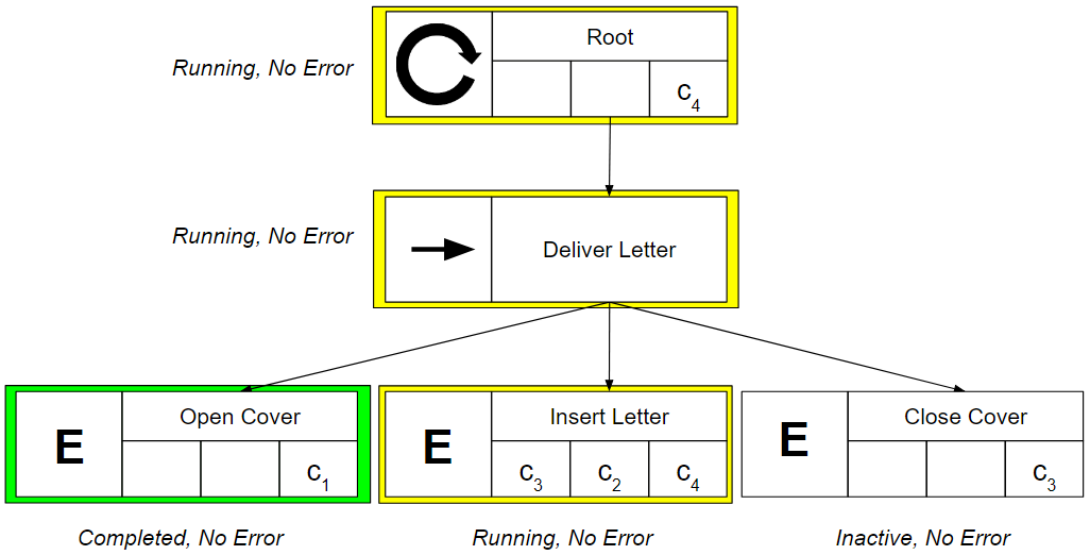


Figure 3.13: Student Model after opening the cover. The green frame indicates a $Completed$ state.

At this point, a number of different events could occur. The cover of the mailbox could be closed again, either by the learner or random events in the environment, such as blowing wind. This would satisfy the $Abort$ condition of *Insert Letter*. In this case, this node, as well as the parent sequence node, would be updated to the $Aborted$ run state. Eventually, the *Root* repeater node would evaluate its *Complete* condition, which is not satisfied, and reset its child nodes. In the end, this would result in the same situation as in Figure 3.11. Since the *Constraint* condition is however not violated by this action, the error state of the node stays false.

Alternatively, the learner could decide to insert the letter in one of the openings. In this case, the *Constraint* condition c_2 with the following formula

$$(LetterInA \rightarrow (LetterForA \wedge InspectedLetter)) \wedge$$
$$(LetterInB \rightarrow (LetterForB \wedge InspectedLetter))$$

becomes relevant. On one side, this condition requires the action of inspecting the letter to be performed before inserting the letter. Since the time of execution for this action is ill-defined (it could be done before, or after opening the mailbox), it is handled in this constraint instead of its own elementary node. On the other side, it requires the letter to be inserted into the right opening. If one of these requirements fails after inserting the letter, the $Constraint$ condition will be violated, and thus, the *Error* state of insert letter will be set to $true$, resulting in the situation shown in Figure 3.14. Else, if the task was performed correctly, the state depicted Figure 3.15 would be achieved. In both cases, the run state of the Insert Letter node updates to $Completed$.

After inserting the letter, the last node of the *Deliver Letter* sequence starts running until the mailbox cover is eventually closed. This would cause the *Root* repeater to positively evaluate its *Complete* condition

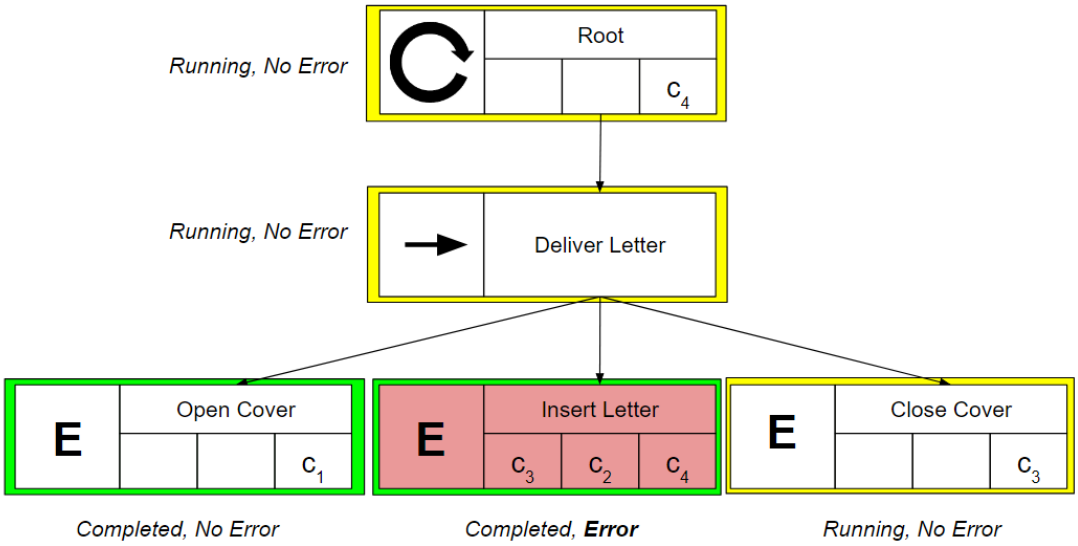


Figure 3.14: Student Model after inserting the letter incorrectly. The pink color indicates an $Error$ state.

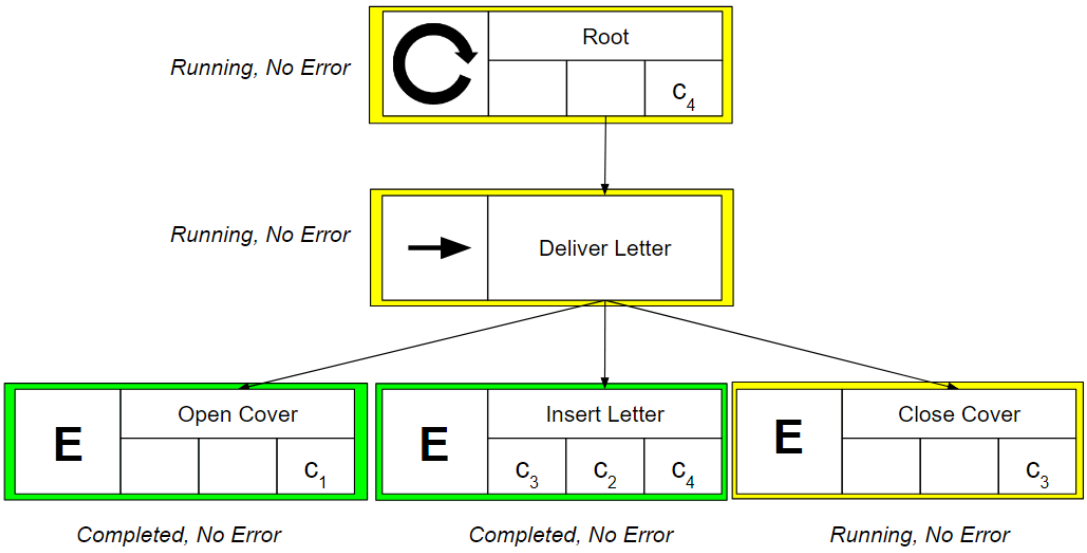


Figure 3.15: Student Model after inserting the letter correctly

and finish the mail delivery task. Assuming that the learner inserted the letter correctly, the Student Model at the end of the exercise would look like in Figure 3.16. This example demonstrated the Student Modeling process in BTT. It showed how the update mechanism is able to handle dynamics in the environment and how ill-defined aspects of a task can be expressed using constraints.

3.5 Utility for VR-ITS

After introducing the structures and algorithms of BTT, this section describes how it can be used by a VR-ITS to provide adaptive tutoring functionalities. This includes discussing its utility for providing instructions, feedback, and scaffolding during training, sequencing and selection of exercises and compatibility with the concepts of buggy rules and uncertainty.

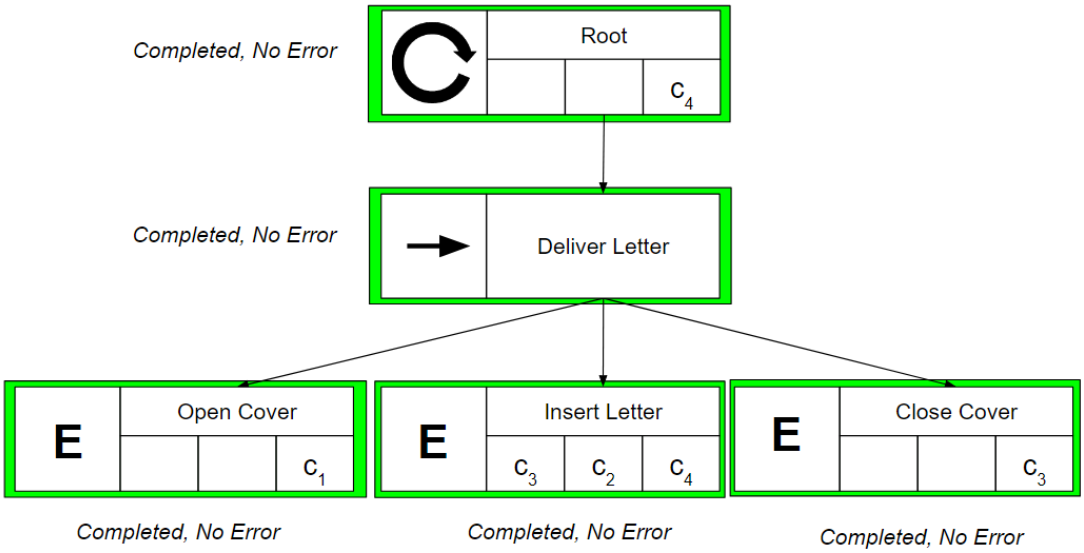


Figure 3.16: Student Model after successful completion of the task

3.5.1 Instructions, Feedback, and Scaffolding

Instructions, feedback, and scaffolding are three pedagogical actions that fall under the category of interactive problem-solving support with the goal to "...provide the student with intelligent help on each step of problem-solving - from giving a hint to executing the next step for the student." (Brusilovsky & Peylo, 2003). The general approach to achieve this in BTT is to annotate the nodes of the Expert Behavior Tree with these kinds of pedagogical actions. Then, at runtime, the Student Model would determine which of these actions to execute. For example, instructional text could be attached to a node that is displayed when the node is in running state. Feedback can be provided in similar fashion. In BTT, a task is considered to be completed successfully when the task's node switches from running to completed without error. On the other hand, a task execution is erroneous when the constraint conditions of the respective task are violated. In both cases, feedback messages associated with the node can be shown to the user. Scaffolding of virtual training environment can also be implemented in a straightforward way using BTT. Since the relevant world state propositions can be derived from the Student Model for each time frame, generic functions can be implemented, for instance, to highlight the related objects. The BTT Expert Behavior Tree structure can be further exploited to control the granularity of pedagogical actions. For instance, detailed and fine-grained support can be linked to elementary nodes and more abstract support to composite nodes. When the learner requests help at runtime, the system could start from the root node and go top-down through the list of running nodes to provide support with increasing level of granularity. Examples of these functionalities are provided for the pedestrian safety domain in Chapter 4. Last but not least, the Expert Behavior Tree also provides the blueprint for the implementation of Pedagogical Agents (PA) (See Chapter 2). Since BTs were originally developed for controlling AI agents, the structure and hierarchy of the Expert Behavior Tree can be reused for this purpose. Basically, functions need to be implemented for elementary nodes that instruct the agent to perform the required actions.

3.5.2 Exercise Sequencing and Selection

While adaptive problem support provides assistance within an exercise, the goal of adaptive sequencing and selection is to provide the learner with such exercises that fits his or her current skill level and learning goals. There are different selection strategies, such as selecting easier exercises when the learner struggles or such exercises that target specific weaknesses. An in-depth discussion would exceed the scope of this thesis but can be found, for instance, in (Vanlehn, 2006). Instead, this section discusses how BTT can support exercise sequencing and selection in general. Given a BTT Task Model, a tree coverage can be determined for each exercise. This coverage describes which subtasks are involved in solving the exercise and can be used to calculate the overall difficulty of a task based on the subtasks' individual difficulties. Beside this general measure, the Student Model provides information about errors made by the learner for each EBT node. By aggregating this information across multiple exercises, nodes that are error-prone as well as nodes that the learner has mastered can be identified. Both the general difficulty of an exercise and the individual skill level of a learner with regard to each node or sub-task can eventually be used for the implementation of adaptive exercise sequencing and selection strategies.

3.5.3 Buggy Rules and Uncertainty

Buggy rules and uncertainty in the context of ITS are two more advanced Student Modeling concepts that go beyond standard error recognition. While buggy rules represent common misconceptions, uncertainty stands for the fact that observations of user actions cannot always be interpreted in a deterministic fashion. For instance, the occurrence of errors or correct behavior does not reflect the user's skill level in a binary way. A mistake could be made due to carelessness and not due to a lack of ability (slip). On the other hand, a correct action could be made by accident as a result of random experimentation or luck (guess). Therefore, correct and incorrect actions need to be aggregated to calculate probabilities for mastery of individual skills. In order to support buggy rules, conditions that represent common misconceptions need to be added to the BTT Task Model. During the Student Modeling process, these conditions could then be evaluated whenever the constraint condition is violated. In order to handle uncertainty, Bayesian Knowledge Tracing could be applied (Corbett & Anderson, 1994). Another aspect of uncertainty could arise at the level of error recognition. In the basic version of BTT described above, error recognition is handled by the constraint condition. Thus, the error state is binary. In order to support uncertainty for error recognition, the constraint condition could be exchanged with a function that calculates a probability of an error. In contrary to the error state, there is no straightforward way to enable uncertainty for the abort and complete run state conditions. Therefore, the design goal is to define run state conditions of elementary tasks based on observable and deterministic facts in the virtual world. In order to support domains where this is not possible, a major revision of the BTT approach in future work would be required.

3.6 Chapter Summary

In comparison with the current approaches for Student Modeling presented in Section 2 of this chapter, BTT provides a number of advantages for VR-ITS dealing with ill-defined domains. First of all, BTT shares the benefit of a graphical model with UML activity diagrams. The difference is that the BTT model uses composite nodes, that are tailored for describing behavior in a dynamic environment, and that there are additional conditions attached to the nodes. Thus, the BTT model is less cluttered but also more

coarse. Next, the conditions of BTT can be used to represent complex formulas that involve multiple variables about the virtual environment. This trait further facilitates the handling of dynamic environments similar to the production rules of an expert system and conditions of Constraint-Based Tutoring (CBT). The special feature of BTT is that the set of conditions that need to be evaluated is restricted by the runtime Student Model. This reduces the computation time for evaluating conditions and enables BTT to run at a high frequency without affecting the real-time requirement of VR. An aspect that BTT shares with CBT is the handling of ill-definedness. Using constraint type conditions, the learner is given the freedom to implement different solutions within defined boundaries. These constraint conditions are however accompanied by completion and abort conditions that specify the transition between sub-tasks. As a consequence, BTT can be used to provide next-step hints in the form of intermediate goals. This additional feature comes however at the price of higher complexity, and therefore, the BTT task model requires more effort to design as compared to CBT.

In summary, BTT combines the structure and update mechanisms of Behavior Trees (BT) with the evaluation of task-related conditions to create a Domain and Student Modeling method that is compatible with ill-defined tasks in dynamic environments. It offers a readable graphical representation and provides information about the learner that can be used to provide adaptive tutoring functionalities. Although additional research and the application to other domains is required to confirm the usability of the BTT approach, the results of this thesis indicate great potential. Last but not least, VR software is developed using Game Engines nowadays, and since BTs are popular among game developers, tools for design and integration of BTs into VR software are available. From a practical point of view, this is another reason to explore the potential of BTs for VR-ITS further.

Chapter 4

Modeling Pedestrian Safety for Intelligent Tutoring

After providing a brief overview of the child pedestrian safety domain in Chapter 2 and the Behavior Tree Tracing (BTT) approach in Chapter 3, this chapter brings these topics together. It starts with an in-depth presentation of pedestrian safety skills with their cognitive demands and the resulting difficulties for young children. In addition, the relevant parameters and constraints of the environment will be defined for each skill. The subsequent section will then focus on the structure of a pedestrian journey and the order in which the skills need to be applied. These dynamic sequences are then represented using an Expert Behavior Tree (EBT).

The analysis of the pedestrian safety task in this chapter is mainly based on literature research and supporting interviews with parents and teachers. As discussed in the previous chapter, pedestrian safety is a domain with ill-defined tasks in a highly dynamic environment and thus challenging to formalize. In fact, most sources of information found in literature are informal and general guidelines of behavior rules (e.g. by Warwitz (2009)). The only exception, to the best of our knowledge, is a systematic modeling approach by Van der Molen, Rothengatter, and Vinje (1981). In their detailed analysis, the authors identified and modeled 26 pedestrian tasks that are divided into several layers of sub-tasks. Further, they also specified a total of 178 relevant system characteristics split into the four categories environment, traffic, person and social/personal context. There are several reasons why this fine-grained and elaborate formalization was not directly adopted for the ITS presented in this thesis. First, the model includes a number of tasks that were not intended to be trained with the VR-ITS. This includes, for instance, determining journey purposes or wearing headgear in a way that does not hamper vision. Then, there is also a set of tasks or sub-tasks that cannot be captured by the system, such as “Recognize the difference between the side of the street and elsewhere”. Furthermore, the task descriptions are formulated in a highly technical way that is difficult to understand, especially for young children. An example for a task description would be “Maximize not being on the street by carrying out Tasks 22, 11, 8 and 18 with preference for Task 19 (being elsewhere) about Tasks 14 (being on the side of the street)”. The Behavior Tree based model presented in this chapter is nevertheless inspired by the work of Van der Molen et al., but pruned and optimized it for usage by a VR-ITS.

4.1 Pedestrian Safety Skills

In order to be a safe pedestrian, there are a number of rules that need to be followed. This includes, for example, the rule of waiting for a green pedestrian light before crossing or keeping away from the road while walking on the sidewalk. Although some of these rules are very similar and common around the world, there are still variations depending on factors such as specific traffic regulations of different countries. As this research is based in Germany, the rules presented in this chapter are mainly derived from the Karlsruher Modell (Warwitz, 2009), a popular traffic safety education program in Germany. Given these pedestrian safety rules, the corresponding skills can then be defined as the ability to interpret the rules within the context of real-world traffic scenarios and to act accordingly.

4.1.1 Basic and Advanced Skills

The difficulty to carry out a pedestrian safety skill correctly varies greatly based on the respective rule and the state of the environment. While a skill such as wait for green pedestrian light before crossing is straightforward, the skill to find an appropriate gap within traffic to cross is much more demanding, especially, in dense traffic. The difficulty of some skills is further amplified by the cognitive development of young children (Toroyan, Peden, et al., 2007). J. Thomson, Tolmie, Foot, and McLaren (1996) defined four high-level skills that are mandatory for pedestrian safety: (I) Detecting the presence of (potentially dangerous) traffic; (II) visual timing judgments; (III) integrating information from different directions and (IV) coordinating perception and action. These high-level skills require a number of underlying abilities that can be highly challenging for young children in certain crossing scenarios. For instance for (I), it is essential to recognize the potential of hidden cars that might not be visible at the moment. In experiments described by Kwame Ampofo-Boateng and Thomson (1991), it has been shown that children aged 5 to 7 exclusively rely on the visible presence of cars to determine whether a certain place is dangerous to cross. Factors such as obstacles obscuring the view are not recognized as a threat by them. Another important skill for (I) is to pay selective attention to those parts of traffic that are relevant for one's safety. Hill et al. (2000) showed that children aged 4-9 have difficulty paying attention to relevant information and ignoring irrelevant one. For (II), the ability to calculate the time to impact of an oncoming car or the time required to cross a certain distance is well developed for adults but not so for children (J. Thomson et al., 1996)). In case of (III), it is important to be able to observe the road from a global perspective, which is also more difficult for young children (Underwood et al., 2007). Last but not least for (IV), children must learn to calibrate perception and movement (J. Thomson et al., 1996). Based on these findings from pedestrian safety literature, the skills of this domain can be categorized into two groups. A group of "advanced skills" that poses the above-mentioned cognitive challenges and a group of "basic skills" that are comparably easier and straightforward to implement.

4.1.2 Skill List

This section provides the complete list of the skills that are considered by the SafeChild system. In addition to the description of these skills, the relevant environmental variables and assessment rules will be provided as well. These correspond to the constraints and world states of the Behavior Tree Tracing (BTT) approach as described in the previous chapter. The values and parameters for the rules and constraints are based on typical car speed and braking formulas and improvised values when there was a lack of detailed

description in literature. This includes, for instance, the distance that a pedestrian must keep away from the road while walking on the sidewalk before it is considered as too close. In case a skill is considered an advanced skill, the respective reason will be provided as well. Since these skills are organized in an EBT, the constraints do not need to ensure that the pedestrian is in the appropriate situation because this is being taken care of by the run state of the respective node. For instance, the constraint for not to stop while crossing only needs to check whether the user has stopped moving and not whether the user is currently crossing the road. This is made sure by the fact that the crossing node has been set to the running state. An overview of the skills are provided in Table 4.1.

Rule	Traffic Light	Zebra Crossing	Unregulated Crossing
Keep distance from a road while walking on a sidewalk	×	×	×
Stop at a curb	×	×	×
Cross straight	×	×	×
Cross without stopping	×	×	×
Observe cars while crossing *	×	×	×
Make sure that cars have stopped *	×	×	
Cross at designated area	×	×	
Find a regulated area to cross	×	×	
Wait for the green light before crossing	×		
Look Left-Right-Left		×	×
Find a good unregulated area to cross *			×
Find a good time gap to cross *			×

Table 4.1: Traffic safety skills in SafeChild. Advanced skills are marked with *.

Skills to Perform on Sidewalk

These skills are related to determining safe routes to the destination and walking safely on the sidewalk. This includes recognizing regulated crossing locations if they are present (zebra crossing or traffic light) and safe unregulated places to cross otherwise.

Keep distance from road while walking on the sidewalk (all crossing types) The correct application of this skill forbids the pedestrian to come too close to the road or even to enter the road while there is no intention to start crossing at the current part of the sidewalk. The rule and assessment of the skill checks the pedestrian's position and calculates the shortest distance to the road. If the distance is smaller than half a meter, the skill will be considered as failed.

Relevant World States: Distance to road

Constraint: Distance to road must be greater than 0.5m

Find a regulated area to cross (regulated crossings) If there is a regulated crossing area nearby, it should be the preferred crossing option. Therefore, it is important to explore the environment to determine whether a regulated area to cross is nearby. In this work, zebra crossings and traffic lights are considered as regulated crossings. In the real world, crossing places regulated by police officers should be included

as well. The rule for this skill is that if there is a zebra crossing or traffic light nearby, the user should look around and make sure that it gets into his/her field of vision.

Relevant World States: objects seen by user, proximity of regulated crossings

Constraint: If a regulated crossing is within 100m, it should be used.

Find a good unregulated area to cross (unregulated crossing) If there is no regulated crossing nearby, and it is still necessary to cross the road, a suitable unregulated area needs to be found. The criteria for such a place are low traffic density and sufficient vision to both sides along the road to detect oncoming traffic in time. In real life, finding such a place requires paying attention to the curvature and slope of the road, parking cars, and other view obstacles. In addition, factors such as number of lanes and speed regulations have to be considered as well. In SafeChild, the complexity is reduced to parking cars and road curvature for the current implementation. Nevertheless, the skill is considered an advanced skill since it requires the pedestrian to have a holistic understanding of traffic and to make judgments about the line of sight and distances. The system assesses the skill by raycasting from the position of the pedestrian to determine whether the line of sight is sufficient. This means that cars out of the line of sight are far away enough to not threaten the crossing process.

Relevant World States: user position, user road vision, proximity of regulated crossings

Constraint: If no regulated crossing is nearby, the user has to move to a position that offers sufficient vision on the road.

Cross Preparation Skills

Before the pedestrian can safely cross the road, a number of preparatory steps are required. While some steps need to be carried out for all types of crossings, others are more specific. For instance, the skill of waiting for the green light before crossing is only applicable to traffic lights. All the following skills assume that the pedestrian has already successfully found and walked to the respective place to cross.

Stop at curb (all crossing types) This skill requires the pedestrian to come to a full stop near the curb of the sidewalk before starting the crossing process. This should ensure that the pedestrian, especially in the case of inexperienced children, has sufficient time to perform the appropriate preparations. The rule behind this skill demands the pedestrian to stop moving in a radius of two meters around a potential place to cross.

Relevant World States: user speed, user position, distance to road

Constraint: The user is at a place to cross and has stopped within 2m to the road.

Wait for the green light before crossing (traffic light) One of the most basic and common rules in traffic safety is the need to stop when seeing a red light. This rule applies to pedestrians as well, and it is mandatory to wait for the light to turn green before starting to cross. The situation becomes more complicated if the pedestrian arrives at a traffic light that has been green for a while. In this case, the light could turn red at any moment. Normally, there should be a so called protection time that ensures that the

pedestrian can safely reach the other side of the road, even if the light turns red after entering the road. Real life experience has however shown that this is not always reliable. For SafeChild, protection time is implemented, and the system considers the skill as correct as long as the light was green when the user entered the road.

Relevant World States: objects seen by user, state of nearby traffic light

Constraint: The user has seen the traffic light while it was green.

Look left-right-left (zebra crossing, unregulated crossing) At zebra and unregulated crossings, the pedestrian is supposed to perform the head movement pattern of looking left, right and left again. Similarly to waiting for the green light before crossing, this is a very common and often taught safety rule. While the head movement itself does not directly affect the safety of a pedestrian and is straightforward to implement (thus considered a basic skill), it facilitates two other advanced skills. In case of a zebra crossings, it helps with the skill of making sure that cars have stopped and in case of unregulated crossings it helps with finding a good gap to cross. For both of these skills, simply looking left-right-left once might not be sufficient. Depending on the density of traffic and the actual traffic situation, additional left and right looks have to be performed. The reason why there is an additional left look at the end of the pattern is due to the fact that in right-hand traffic, the car coming from left is closer to a pedestrian who is about to cross. Therefore, it is important to double check the left side before starting to cross. In order to assess this skill, the user's head movement is monitored, and the skill is considered successful if a sequence of left-right-left look is detected. (A left or right look is detected, when the viewing angle of the user surpasses 45 degrees to the left or right compared to the perpendicular line to the road)

Relevant World States: user view direction history

Constraint: User view direction history has to include a sequence of left-right-left.

Make sure that cars have stopped (regulated crossings, advanced skill) At regulated crossings, there are defined times when cars have to stop in order to let pedestrians cross the road. At a traffic light, the cars need to stop at red light and at a zebra crossing, cars are supposed to stop if there are pedestrians who want to cross. However, drivers could disregard these rules due to carelessness or on purpose because they are in a hurry. Therefore it is necessary to perform a visual check that cars are indeed stopping before starting to cross. This skill requires the pedestrian to be aware of hidden dangers and to pay selective attention to those cars that are relevant. Thus it is considered an advanced skill. The assessment of this skill is more complex than the above-mentioned skills. The system needs to determine which cars are relevant and whether the user has seen them. In the present implementation, the closest car approaching from left and right are considered as relevant, if they could potentially endanger the pedestrians crossing process. This means that if these cars would not slow down, they could hit the pedestrian. In order to check whether the pedestrian has ensured that these cars are stopping, the system checks whether these cars were in the user's field of vision during or after braking.

Relevant World States: objects seen by user, proximity and speed of approaching cars

Constraint: If the closest car from left or right could endanger the pedestrian, s/he should have seen it slow down.

Find a good time gap to cross (unregulated crossing, advanced skill) In order to cross an unregulated crossing safely, it is necessary to find an appropriate gap in traffic that provides sufficient time for the pedestrian to cross. This skill is very challenging for young pedestrians and requires proficiency in and coordination of a number of cognitive and perceptual abilities. First, the pedestrian needs to predict the time needed to cross based on the width of the road. Second, it is also necessary to identify oncoming cars and to estimate their distance and time of arrival. Last but not least, it is also important to remember and project the state of cars from one side, while looking towards the other side. The assessment of this skill relies on a conjunction of several constraints. When the pedestrian starts to cross the road, no car should be within a threatening distance. Also, the pedestrian should be aware of this safe state by having looked both sides recently. The sequence of looking left and right is not constrained for this skill as it is already taken care of by the look left right left before crossing skill. As with the other perception related skills, the system can only assess whether the pedestrian showed the necessary actions for performing this skill correctly, without knowing for sure whether all required cognitive processes took place.

Relevant World States: user view direction history, proximity of approaching cars

Constraint: If there are cars from left/right, they should be at least 80 meters away, and the pedestrian should have looked left/right within the last 4 seconds.

Crossing Skills

Cross at a designated area (regulated crossings) For regulated crossings, there is a marked area where the crossing should take place. Cars are supposed to stop outside this area, and pedestrians need to stay within this area during the entire crossing process. The correctness of this skill is assessed by constantly monitoring the user's position during the crossing process.

Relevant World States: user position, current crossing area

Constraint: user position is within boundaries of crossing area.

Cross straight (all crossing types) While crossing the road, the pedestrian needs to cross in a straight line to minimize the time spent on the road. Especially children need to learn to resist the temptation of crossing the road diagonally in order to shorten the total distance to the goal position. The success of this skill is assessed by constantly monitoring the angle difference between the perfect crossing line and the current crossing trajectory of the pedestrian.

Relevant World States: user crossing angle

Constraint: The deviation of user crossing to perfect crossing angle must be below $30°$.

Cross without stopping (all crossing types) In addition to crossing straight, it is equally important not to stop while crossing as it would also delay the time spent on the road and expose the pedestrian to greater threat. For this, the system monitors the pedestrians speed while crossing and reports a failure in this skill if a full stop is detected.

Relevant World States: user speed

Constraint: The user must not stop.

Observe cars while crossing (all crossing types, advanced skill) Although the preparation phase before crossing should make the crossing itself as safe as possible, it is still necessary to stay vigilant while being on the road. Due to the unpredictable nature of traffic, there is no guarantee that other traffic participants will always follow the rules. In order to implement this skill correctly, the pedestrian is supposed to look both sides while crossing. This is considered an advanced skill due to its dynamic nature. The action of looking left and right has to be performed under time pressure while the pedestrian is on the road and in combination with the two above-mentioned skills, crossing straight and crossing without stopping.

Relevant World States: user view direction history

Constraint: User view direction history has to include left and right looks before reaching the opposite side.

4.2 Pedestrian Safety Task Structure

The subset of relevant skills, as well as the order to apply them, is dependant on the specific crossing types and traffic conditions of a pedestrian's journey. In SafeChild, the respective logic of these dependencies is represented using an Expert Behavior Tree (EBT) as introduced in Chapter 3. The general idea is that the pedestrian alternates between walking on the sidewalk and crossing roads until the goal position is reached. Consequently, the root node of the EBT is a Repeater node that keeps the EBT running as long as the pedestrian is not at the goal position. Directly under the root node is a sequence node with two children. The first child is the root node of the sub-tree that represents behavior on the sidewalk while the second child is the root node of the crossing behavior branch. The following Figure 4.1 shows the first two levels of the EBT.

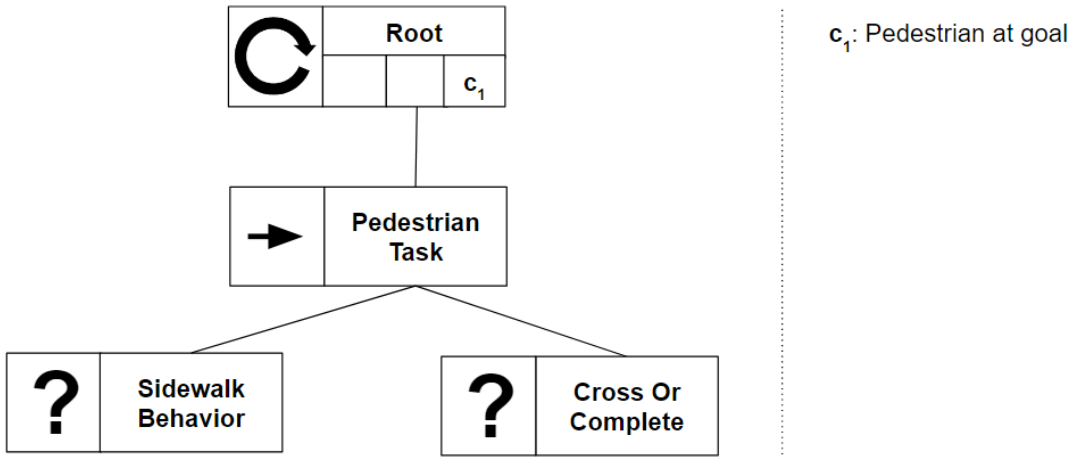


Figure 4.1: Pedestrian safety Expert Behavior Tree (EBT) top-level structure

4.2.1 Sidewalk Behavior

While being on the sidewalk, the pedestrian could be following one of two behaviors. If the goal position is reachable without crossing, s/he should be walking directly towards the goal. Else, the intermediate

destination should be an appropriate place to cross the road. To reflect this, the root node of the sidewalk behavior is a selector node with two children representing the behaviors of walking towards the goal and walking towards a place to cross. In both cases, a sequence of two *Elementary* nodes (one for finding and one reaching the target location) is used to describe the respective behavior. While performing these behaviors on the sidewalk, the user needs to keep distance to the road. A third child of the sidewalk behavior root node is used to capture those cases where the user has unexpectedly (and erroneously) entered the road before reaching a place to cross. Figure 4.2 shows the sidewalk behavior sub-tree.

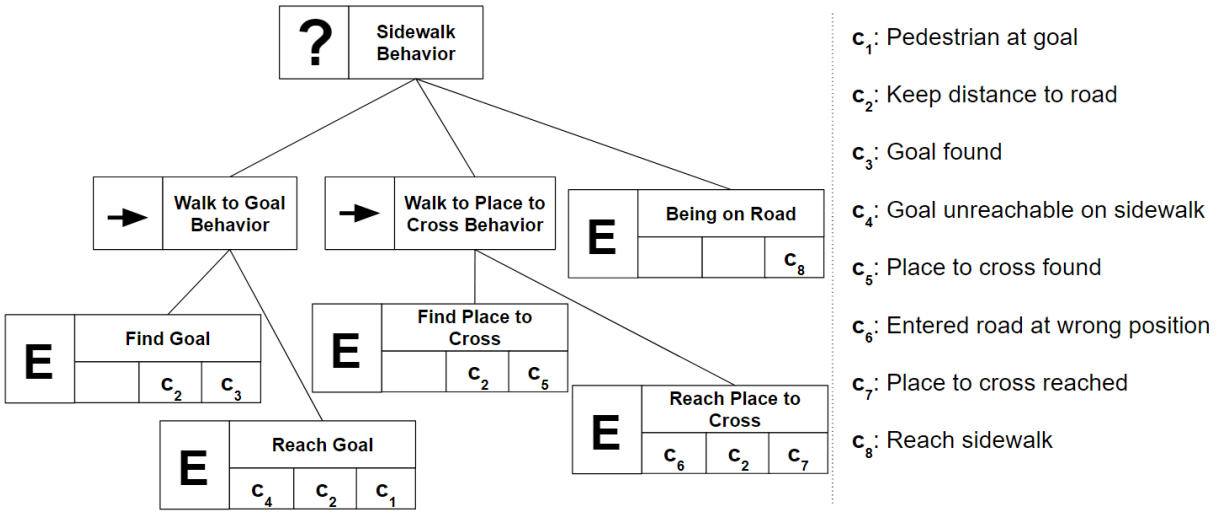


Figure 4.2: Sidewalk behavior sub-tree

4.2.2 General Crossing Behavior

With the structure of EBT so far, there are two possible scenarios when the *Crossing* behavior gets activated. Either the pedestrian has reached a place to cross or s/he has already reached the goal. In order to model this (Figure 4.3), the root node of the crossing behavior is a *Selector* where the first child is an *Elementary* node that checks whether the goal is reached (in this case, the entire task will be completed without the need to activate the crossing related part of the EBT). The second child represents the actual crossing behavior.

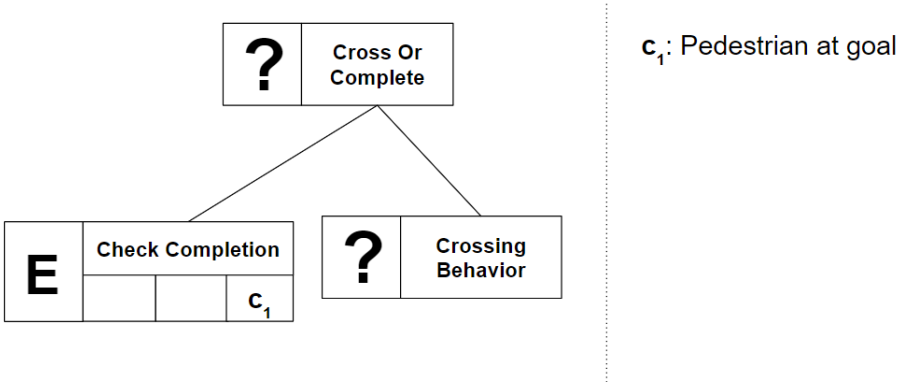


Figure 4.3: Crossing behavior top-level tree

Individual Crossing Behaviors

In order to cross a road, the pedestrian has to select and identify the type of crossing to use. Then, appropriate preparations have to be performed to ensure that it is safe to cross at the chosen location. Finally, the crossing action itself has to be carried out. To represent this (Figure 4.4), the EBT uses a *Selector* node with three children; one for traffic light, one for zebra crossing and one for unregulated crossings. These child nodes are sequences of preparation and crossing behavior and will be presented in more detail below.

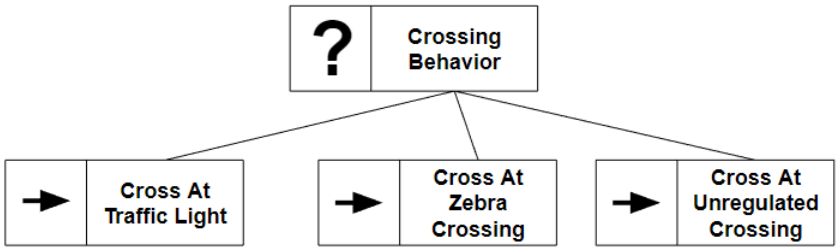


Figure 4.4: Crossing behavior second-level tree

Cross at Traffic Light Both the preparation and crossing behaviors are represented as *Elementary* nodes in the EBT with a number of constraints representing the different skills mentioned above. For the preparation behavior in the traffic light case, there are three constraints that correspond directly to the skills: *Stop at Curb*, *Wait for Green* and *Make Sure That Cars Have Stopped*. In addition, the completion condition monitors whether the pedestrian has entered the road while the abort condition checks if the traffic light area has been left. For the *Cross at Traffic Light* behavior, there are also several constraint conditions representing the skills of *Observe Cars While Crossing*, *Cross Without Stopping*, *Cross at Designated Area* and *Cross Straight*. The completion condition for this node is fulfilled, when the pedestrian leaves the road and enters the sidewalk again (Figure 4.5).

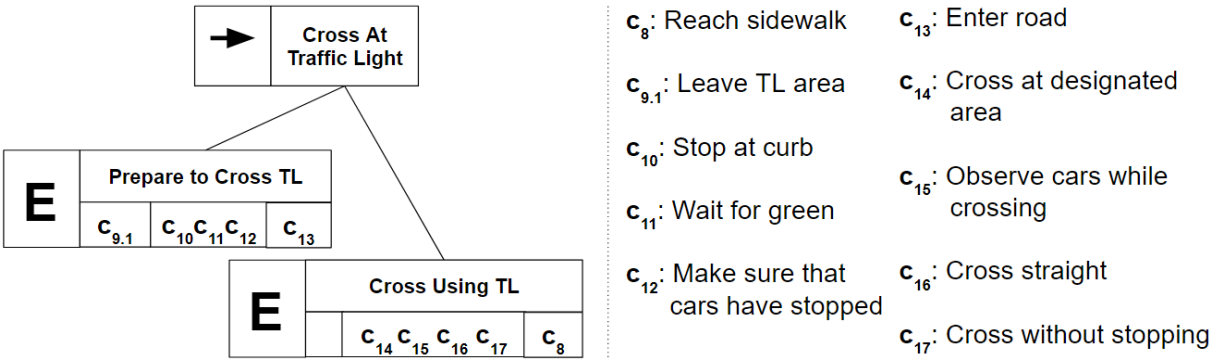


Figure 4.5: *Crossing at Traffic Light* Behavior Tree

Cross at Zebra Crossing The behavior sub-tree for crossing at a zebra crossing is similar to the traffic light version. The only major difference is the exchange of the *Wait for Green* constraint with the *Look Left-Right-Left* constraint. Since there are no lights at a zebra crossing, it is more important to check the surrounding for potentially dangerous cars (Figure 4.6).

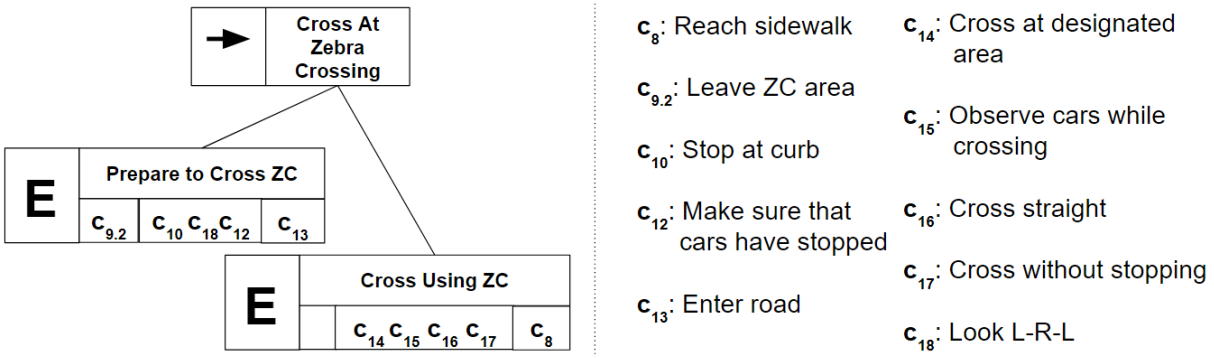


Figure 4.6: *Crossing at Zebra Crossing* Behavior Tree

Cross at Unregulated Crossing The behavior at an unregulated crossing also follows a similar template as the other two crossing types. Unlike those however, cars are not required to stop for a pedestrian at any time, and there is also no designated crossing area. Instead, the pedestrian has to adjust the crossing time to the flow of traffic. Thus, the skill (and respective constraint) of *Making Sure that Cars Have Stopped* is missing from this behavior. In its place, the skill of *Find a Good Time Gap to Cross* is added to the preparation part of this behavior. The result is shown in Figure 4.7 below.

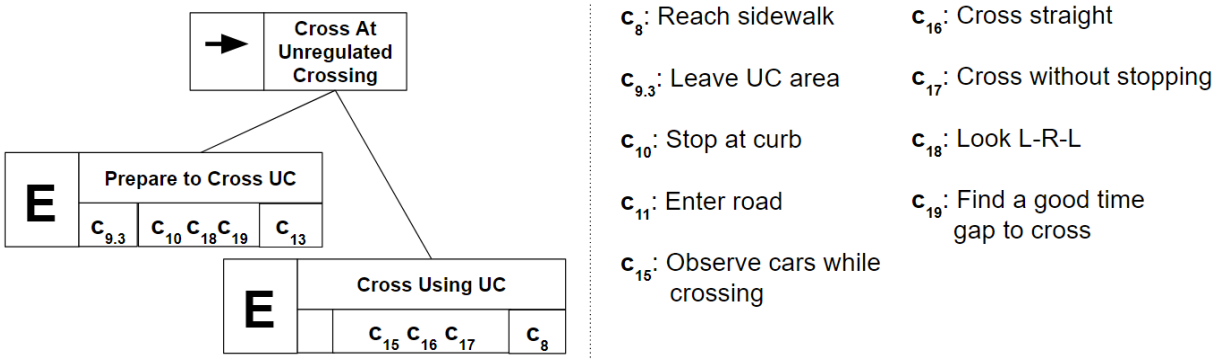


Figure 4.7: *Crossing at Unregulated Crossing* Behavior Tree

4.3 Application of the Complete Model

After modeling the skills and structure of the pedestrian task, this section shows how Student Modeling is applied in practice during an exercise within the SafeChild system. It starts with a complete walk-through for a traffic light based crossing scenario and then describes the deviations for the other two crossing types. In order to keep the EBT readable in the following examples, the tree will be presented in a simplified way by only showing the node types and state for each node besides the leaf nodes. For the leaf nodes, their type will be stripped, and only their name will be shown. Finally, color encoding will be used to represent the run state of a node. Figure 4.8 shows the complete EBT in the simplified version and the color coding for the states. The detailed version of the complete model can be found in Appendix A.

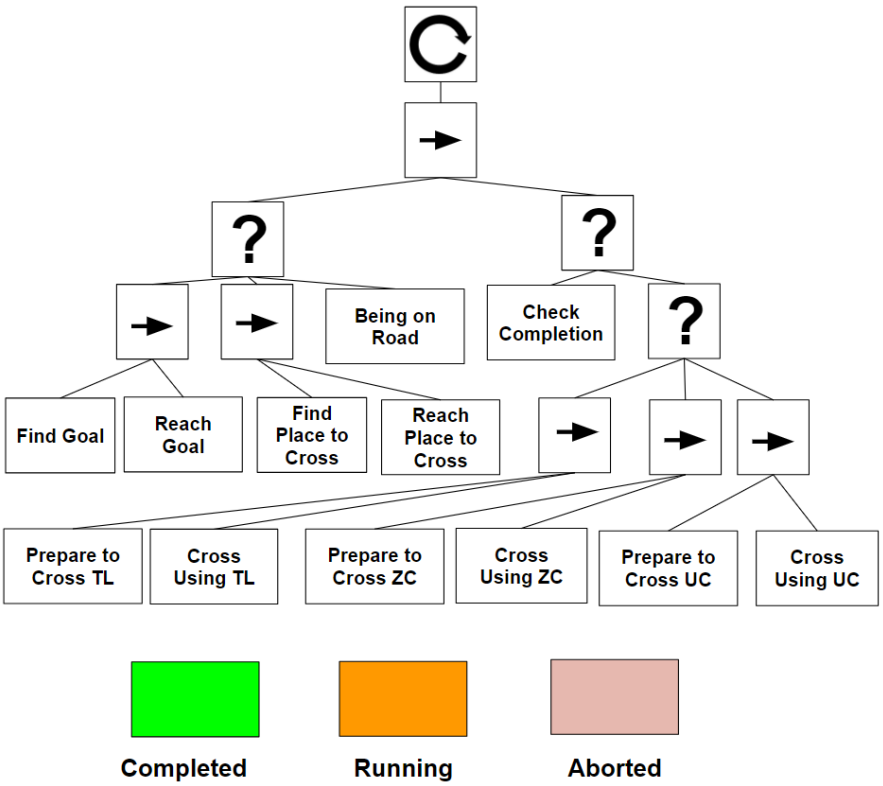


Figure 4.8: Simplified EBT for the pedestrian safety task

4.3.1 Traffic Light Crossing

In this scenario, the pedestrian starts near a traffic light but without seeing it. The goal is on the opposite side of the road but also not within the field of vision of the pedestrian. As a result, the EBT will go from the root into the sidewalk branch and end up on setting the *Find Goal* node to running. Figure 4.9 shows both the EBT and the first person view of the pedestrian for this initial situation.

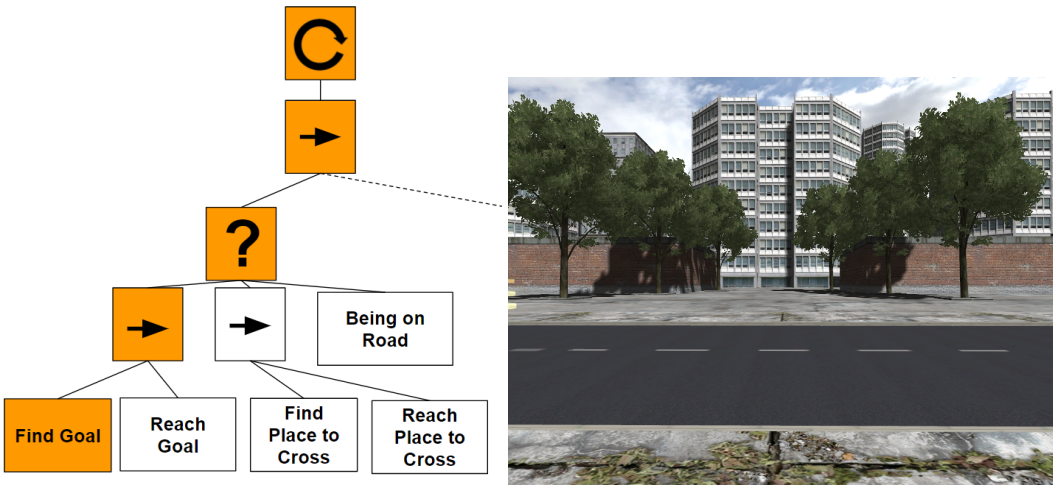


Figure 4.9: Traffic light crossing initial situation

The first thing that the user needs to do is to explore the environment visually until the goal is found. This is accomplished in this specific scenario by turning the head to the left. Once the goal is within the user's field of view, the *Find Goal* node will complete successfully. This activates the *Reach Goal* node which

will immediately abort due to the fact that the goal cannot be reached without crossing the road. As a consequence, the *Find Place to Cross* node will be activated and, since the goal is next to a traffic light, this node will also complete successfully. Eventually, the *Reach Place to Cross* node will be activated, resulting in the state depicted in Figure 4.10.

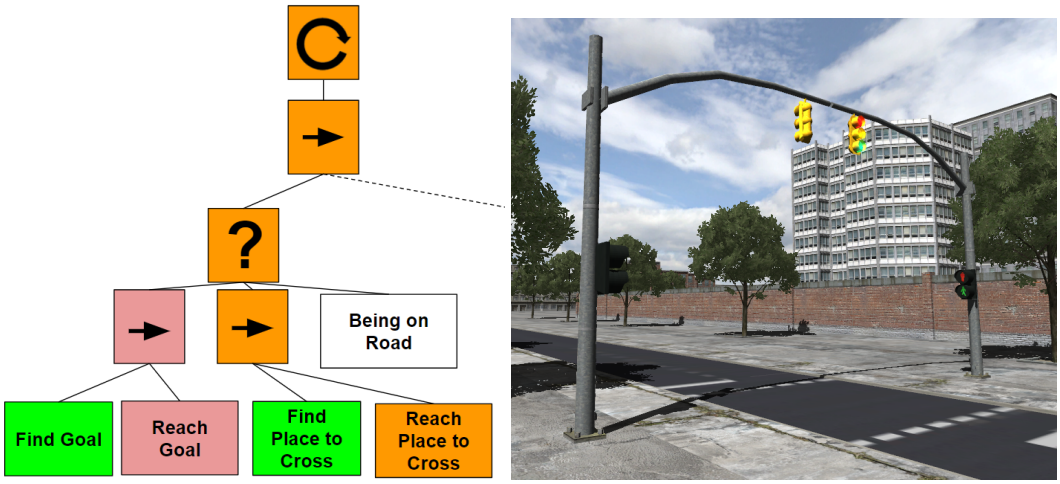


Figure 4.10: Traffic light crossing after finding goal and place to cross

Now, the user should walk towards the traffic light to begin with the crossing preparation. If s/he comes too close to the road before entering the *Crossing* behavior node, the respective constraint will cause the system to register an error. If the pedestrian even enters the road, all nodes in the *Sidewalk* branch would fail except the *Being on Road* node. In this state, the user should return to the sidewalk as fast as possible. When the original or the opposite sidewalk is reached, the system reevaluates the situation in order to determine whether a road crossing is still required or the pedestrian has already (erroneously) crossed the road. By assuming that none of the above-mentioned errors were made and the traffic light is reached by the pedestrian, the *Sidewalk* behavior would be completed and the *Crossing* branch would become active. Since the pedestrian is still on the opposite side of the road from the goal, the *Check Completion* node would abort and eventually lead to the *Prepare to Cross TL* node to become active (Figure 4.11).

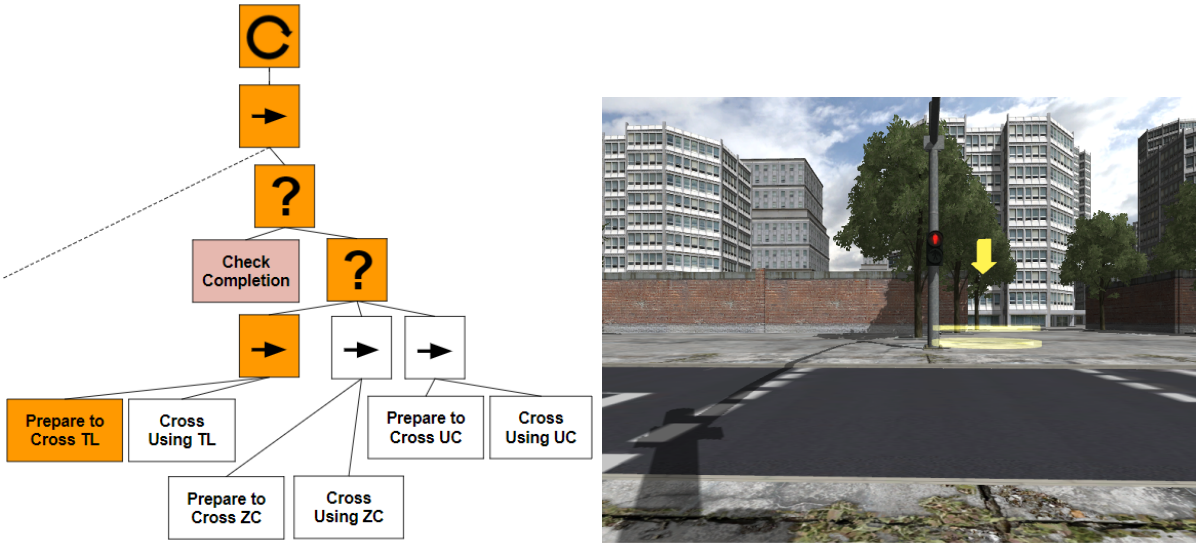


Figure 4.11: Traffic light crossing after reaching place to cross

In order prepare correctly for crossing at a traffic light, the skills mentioned in Section 4.1.2 have to be carried out according to the respective constraints. If any of these skills are not implemented correctly, an error will be registered when the pedestrian enters the road, which is considered as completion of the *Preparation* and starting of the *Crossing* behavior. If the pedestrian decides to not enter the road and to leave the pedestrian light area instead, (which could happen in a real life, for instance, due to forgetting something along the way) *Prepare to Cross TL* node would abort and, since the pedestrian is neither at a zebra crossing nor an unregulated crossing, the BT would eventually be in the same configuration as in Figure 4.10. Assuming that the user completes the preparatory actions and enters the road to cross, the situation as depicted in Figure 4.12 would occur.

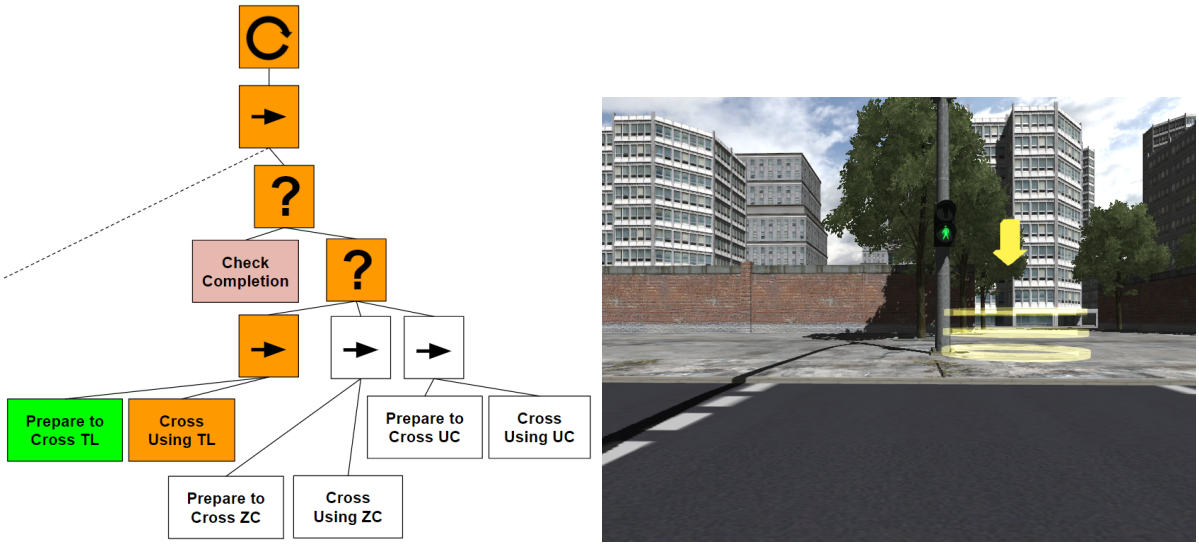


Figure 4.12: Traffic light crossing while crossing

During the crossing process, the skills for crossing at a traffic light listed in Section 4.1.2 need to be applied. As before, skills that were not implemented correctly would violate the related constraints, and the system can react accordingly. In order to complete this behavior, the pedestrian needs to transition from road to sidewalk again. This can either be the sidewalk on the opposite side of the road or the sidewalk from which the crossing was started. In the latter case the system would reset to the state in Figure 4.11. At the moment of reaching a sidewalk, the *Crossing* branch would complete, and, since the user is not at the goal yet, the EBT would switch back to the *Sidewalk* behavior. Assuming that the user correctly reached the opposite side of the road, the goal would become reachable, and thus the *Reach Goal* node would become active. The resulting situation is depicted in Figure 4.13. The last step to complete this scenario is to reach the indicated goal position. Once the pedestrian arrives there, the *Reach Goal* node completes and the *Check Completion* node of the *Crossing* branch will now also complete. Eventually, the root *Repeater* node will be evaluated and, since the constraint of the user being at the goal is satisfied, the entire tree will complete, finishing the crossing at a traffic light scenario. The final state is shown in Figure 4.14.

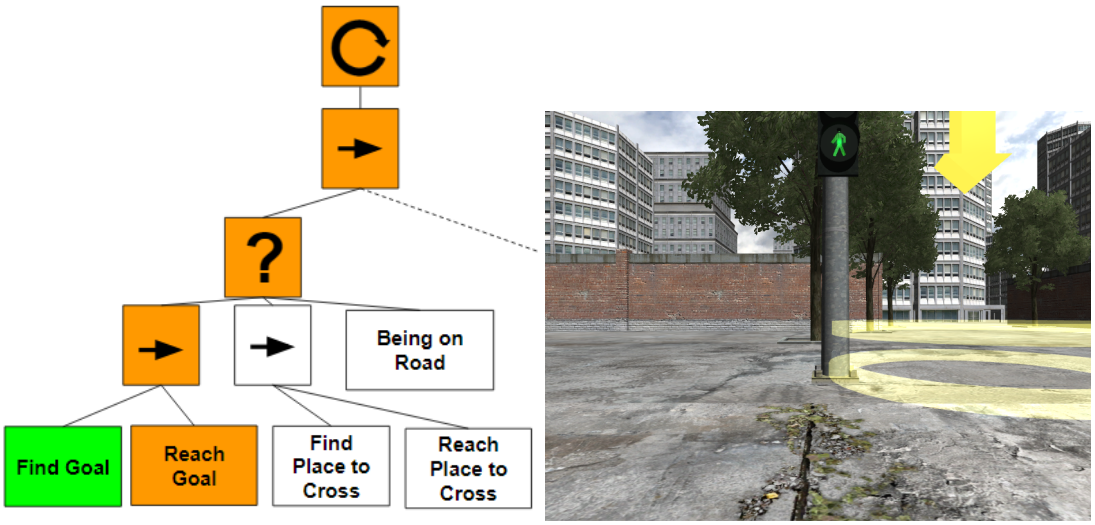


Figure 4.13: Traffic light crossing situation after crossing

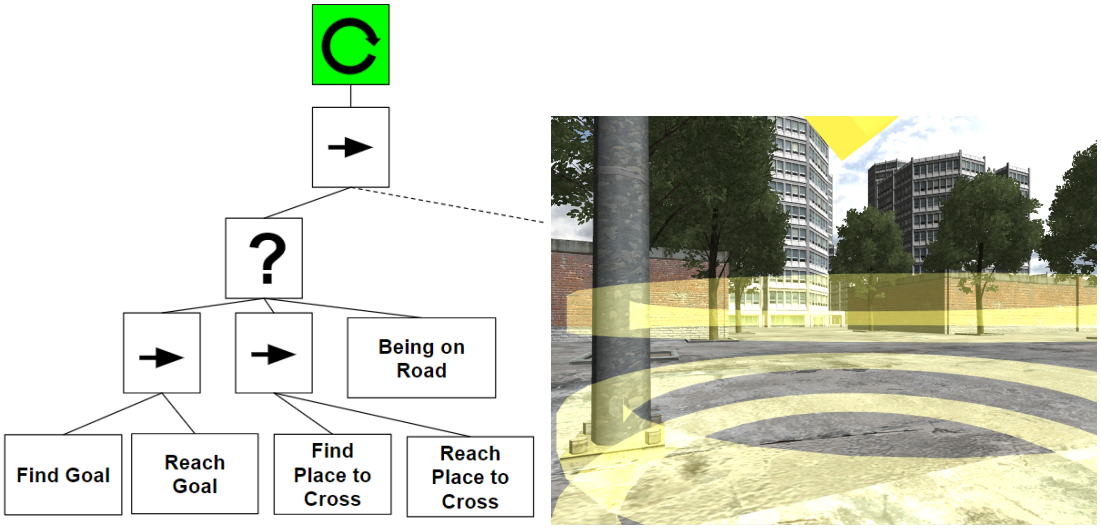


Figure 4.14: Traffic light crossing situation after reaching the goal

4.3.2 Zebra Crossing

The zebra crossing scenario is similar to the traffic light scenario described above. First, the pedestrian goes through the same steps on the *Crosswalk* branch. When the pedestrian arrives at the zebra crossing, the EBT would be temporarily in the same configuration as in Figure 4.10, where the *Prepare to Cross TL* node is active. This is due to the structure of the EBT and the update logic of *Selector* and *Sequence* nodes as described in Chapter 3. The node would however abort at the subsequent update tick due to the fact that the pedestrian is not in at a traffic light. This leads to the situation as depicted in the following Figure 4.15, where the *Prepare to Cross ZC* node becomes active.

4.3.3 Unregulated Crossing

If there is no regulated crossing nearby and the pedestrian is forced to cross at an unregulated place, the task of finding a suitable place to cross becomes a demanding task. In fact, it is necessary to evaluate the field of vision and road visibility to make sure that oncoming cars can be seen from a sufficient distance.

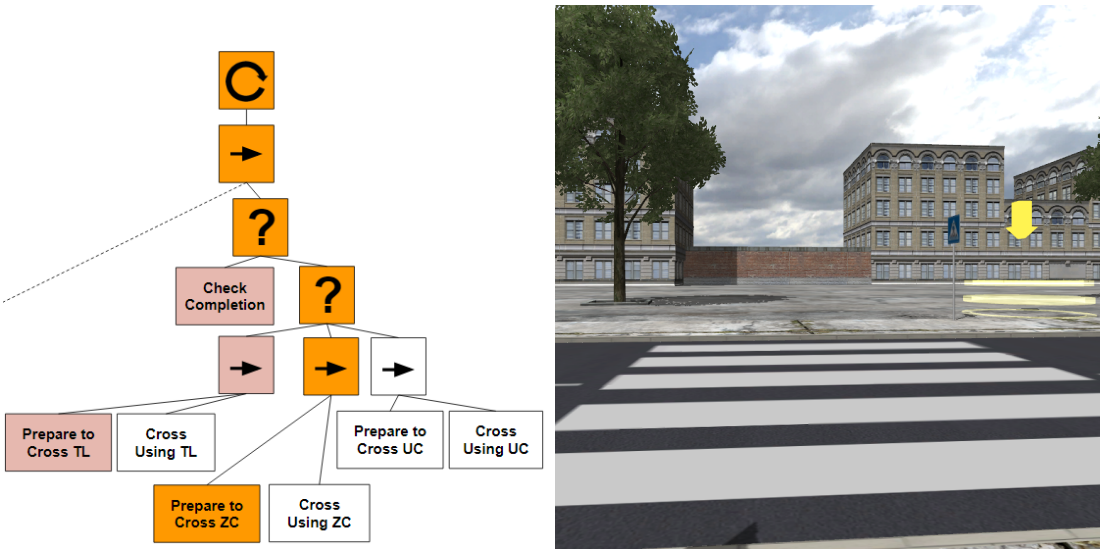


Figure 4.15: Zebra crossing preparation

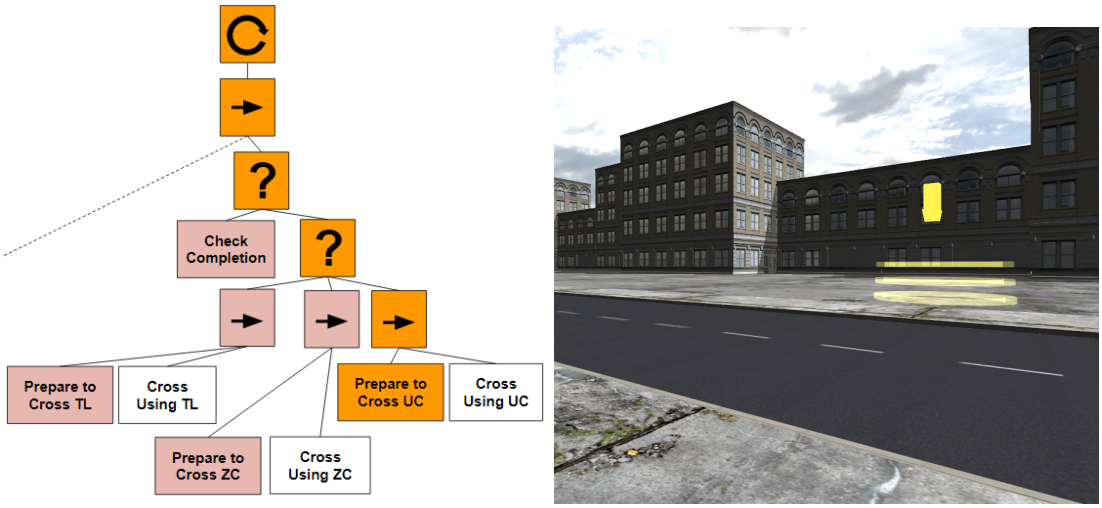


Figure 4.16: Unregulated crossing preparation

To reflect this, the *Find Place to Cross* node does not check whether a certain object has been seen by the user but instead requires him/her to walk up to a place that is suitable for crossing. Once the user achieves this, the *Find Place to Cross* and *Reach Place to Cross* nodes both complete. On the *Crossing* Branch, the *Prepare to Cross TL* and *Prepare to Cross ZC* node will both abort, which leads the *Prepare to Cross UC* node to become active (Figure 4.16). From there, the user needs to perform the relevant skills described in Section 4.1.2 to complete a safe crossing for unregulated crossings. Once the opposite side of the sidewalk is reached, the EBT will be traversed as in the other crossing scenarios.

4.4 Chapter Summary

This chapter described in detail how the BTT Approach is used to design and develop a model for the domain of pedestrian safety education. It presented the individual skills of the domain with their constraints and how the different stages of a pedestrian task can be structured using a EBT. Finally, the completed domain model was shown in action for Student Modeling in all three crossing scenarios. This

approach of Domain and Student Modeling for the pedestrian safety domain has room for expansion and improvements for future work. For instance, the current model does not yet include traffic regulating police officers, pedestrian bridges or weather conditions, which all affect pedestrian safety in the real world. Thanks to the modular setup of the EBT, adding these factors can be done on the basis of the current implementation without restarting from scratch. In addition, the current skills and constraints can be further improved. Due to the lack of precise definitions, a number of variables and constraints are based on intuition and subjective interpretations of pedestrian safety rules. Interdisciplinary research and longtime studies are required to fine-tune these parameters and the related constraints. Since there are also differences in traffic safety rules and driver behavior around the world, these parameters need to be further adjusted depending on local circumstances.

Chapter 5

VR Interfaces

The user interface (UI) of a Virtual Reality based Intelligent Tutoring System's (VR-ITS) is an immersive environment that looks and behaves as in the real world and thus allows the learner to train and perform tasks in a natural way. On top of that environment, further UI elements can be added. This includes classical 2D elements such as buttons, labels or text and also 3D elements that can be integrated into the environment. For instance, 3D markers or arrows can be used to guide the user's attention. The key elements for child pedestrian safety education are roads, traffic and different forms of road crossings. Such an environment was created for the SafeChild system and will be presented in this chapter. In terms of hardware, there are different options to present the virtual environment to the user and to provide appropriate means of interaction (see also Chapter 2). These options vary in the degree of immersion, ease of setup, availability, and costs. The optimal selection and combination of VR hardware components is therefore dependent on the specific requirements of a use case. A VR-ITS such as SafeChild could be used in different settings. This includes usage at home for highly available self-training, usage in a classroom setting for traffic safety lessons and even usage at specialized VR centers that provide a fully immersive experience for specialized training or examination scenarios. Therefore, SafeChild was designed with the support of a wide variety of hardware setups in mind. This chapter presents three different setups. A semi-immersive version with three stereoscopic displays and a Microsoft Kinect[1] camera, a fully immersive version with an Oculus Rift DK2 [2], a Leap Motion[3] sensor and a Myo[4] wristband and last but not least, a desktop VR version using a regular monitor and gamepad input.

5.1 Environment

In real life, pedestrian safety skills are most important in areas with regular traffic. Given the same traffic laws and the same level of driver compliance with those laws, areas with greater traffic density, complexity or car speeds are more dangerous. In Germany, schools are often located in urban areas with high traffic density but low-speed limits. Such an environment was the target of the SafeChild system. More specifically, this virtual learning environment consists of typical urban structures such as buildings, walls and trees, different types of crossing (traffic light, zebra crossing and unregulated crossings) and cars

[1] https://developer.microsoft.com/de-de/windows/kinect, accessed 12-08-2018
[2] https://en.wikipedia.org/wiki/Oculus_Rift#Development_Kit_2, accessed 12-08-2018
[3] https://www.leapmotion.com/, accessed 12-08-2018
[4] https://www.myo.com/, accessed 12-08-2018

Figure 5.1: SafeChild environment and objects. The left side shows a birdview screenshot of the environment while the right side shows the goal position indicator.

moving at around 30 kilometers per hour (which is the typical speed limit for areas around elementary schools in Germany).

5.1.1 Building and Objects

The left-hand side of Figure 5.1 shows a birdview screenshot that depicts buildings and objects from the virtual environment. The models are mostly generic and resemble typical western architecture. Due to limited time and resources, the environment's degree of realism is neither photorealistic nor comparable to high-end computer games. Nevertheless, the user study presented in Chapter 7 shows that the graphical quality was well perceived by the target user group of young children. Also, since these components of the system are mostly static, improving or changing the environment would be straightforward to implement in the future. The only purely artificial object in the environment (that does not represent any real object) is an indicator for the goal position of an exercise. This is represented by animated yellow circles with an arrow on top. The right-hand side of Figure 5.1 shows this indicator during an exercise.

5.1.2 Traffic and Vehicles

In SafeChild, traffic takes place on a road network consisting of a road ring with an additional T- junction (Figure 5.2). Along the road network, there are two traffic lights and two zebra crossings (Figure 5.3). Between the curves of the road ring, there are sufficiently long straight segments that can be used to train unregulated crossings.

On top of the road network is an additional layer that is used to guide cars and to control two-way traffic. More details about the implementation will be provided in Chapter 6. In terms of appearance, there are three generic car models for moving traffic. One model for small cars, one model for medium-sized cars and one model for minivans (Figure 5.4). Each time a car is generated during an exercise, the model of the car along with its top speed is selected randomly. This is done to increase diversity and realism of traffic. In any case, a car will not deviate more than 10 km/h (plus or minus) from a specified speed limit. They will also slow down before hitting a car in front of them and follow the traffic rules at regulated crossings. This means that they will try to stop at a zebra crossing if pedestrians are standing nearby and at yellow or red light at traffic lights. If the braking distance is too short however (for instance if the light

turns yellow and the car is already directly in front of the traffic light), the car will not try to slow down but continue driving. This reflects typical driver behavior.

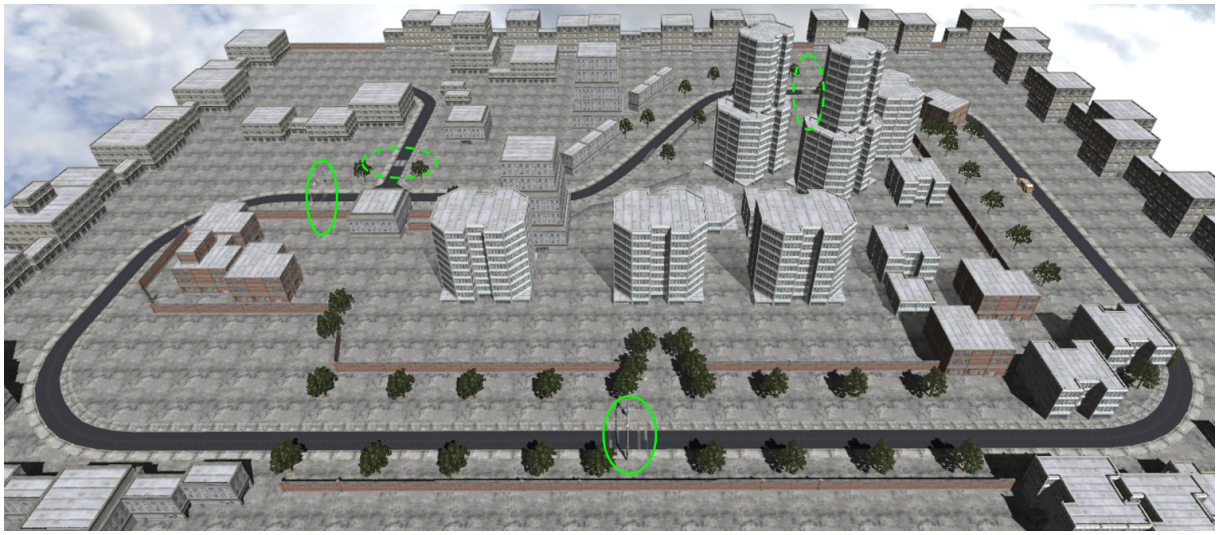


Figure 5.2: SafeChild road network. Traffic lights are marked with solid circles and zebra crossings are marked with dashed circles.

(a) Traffic light with pedest. light (b) Zebra crossing with signs

Figure 5.3: Regulated crossings in SafeChild

Minivan Small car Medium car

Figure 5.4: Cars in SafeChild

Figure 5.5: Semi-immersive interaction setup of SafeChild (consisting of three active stereoscopic monitors and a Microsoft Kinect motion sensor)

5.2 Semi-Immersive Setup

After describing the virtual environment itself, this section describes the first interaction setup that allows a user to walk and look around in the environment. A semi-immersive interaction setup provides more realism in terms of presentation and interaction as compared to a traditional desktop setting. However, it does not fully enclose the user in VR as Head-Mounted-Displays (HMDs) or CAVEs (see Chapter 2) would do. In SafeChild, the semi-immersive setup (Figure 5.5) is achieved through widening the field of view through the use of multiple displays and gesture tracking through a motion sensor.

5.2.1 Display

The display component of the immersive setup in SafeChild consists of three active stereoscopic monitors arranged in a semi-circle which enables a 3D view into the virtual world in a visual angle of about 180 degrees. The monitors on the sides are placed at an angle of roughly 45 degrees, which allows the user to use natural head movements to perform skills such as look left-right-left or exploring the environment. In order to avoid distortions at the edges of the combined screen space, three virtual cameras are used to render the 3D scene to the user. Figure 5.6 shows the difference between a one camera setup versus a three camera setup.

5.2.2 Input

The input device for the semi-immersive setup is a Microsoft Kinect camera. This interaction device uses structured light for real-time 3D reconstruction and skeleton mapping. Gesture recognition, such as waving a hand or raising a leg can then be implemented on top of this information. In case of SafeChild, a total of four gestures have been implemented to allow the user to move and turn in the virtual world. One

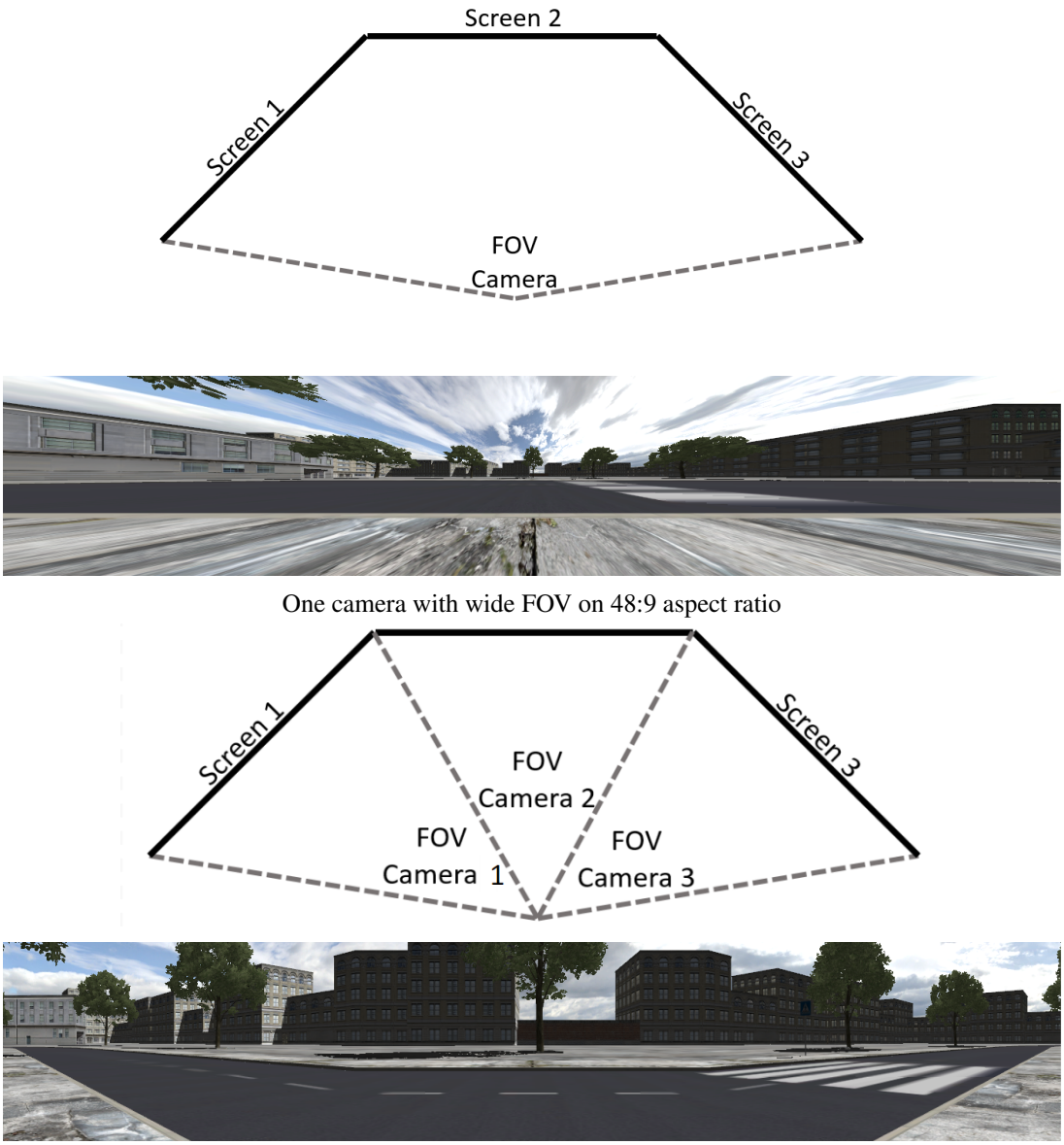


Figure 5.6: One camera versus three cameras

for rotation and three for translational movement. These gestures should feel natural to the user and thus share similarities with real-life walking. An overview of these gestures is provided below while details about implementation and evaluation can be found in the master thesis of Patrick Rump (2014).

Rotation

The forward orientation vector is orthogonal to the line that goes through both shoulders of the user. Therefore, by turning the shoulders, rotation in the virtual world can be achieved. As depicted in Figure 5.7, the degree of rotation is controlled by the angle λ between the original orientation v_0 and the new orientation v_1

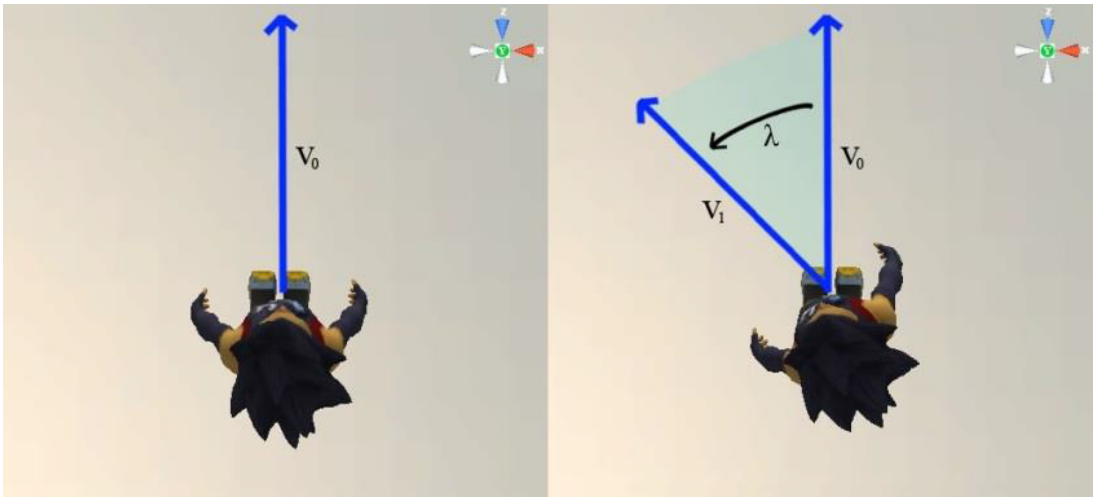


Figure 5.7: Rotation gesture in SafeChild (Rump, 2014)

Translation

The first gesture for translational movement is LEAN. As depicted in Figure 5.8, forward movement is achieved by leaning forward, and speed of movement is controlled by the magnitude of leaning angle λ. A negative value for λ results in backward movement.

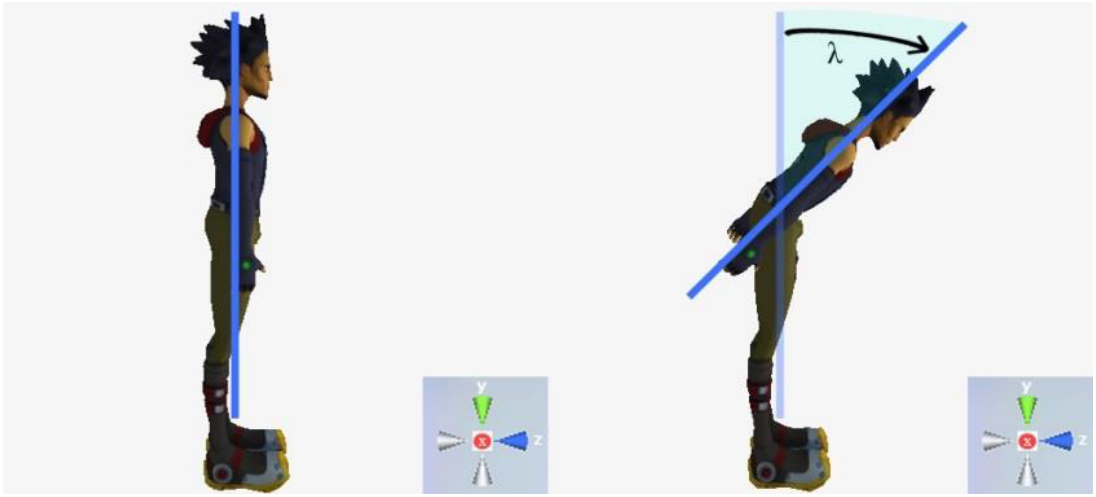


Figure 5.8: LEAN gesture for translational movement (Rump, 2014)

The second Gesture is Walking-In-Place. Here, the user walks in place by his/her raising left and right alternately above a certain threshold to move forward (Figure 5.9). A higher frequency results in faster movement. Although this gesture is close to real-life walking, it is not possible to differentiate between forward and backward movement based on this gesture alone.

The last Gesture for translational movement is called Distance-To-Velocity: For this gesture, the user needs to define and calibrate an initial position first. Afterward, movement is controlled by stepping away from the initial position as indicated by vector v_0 in Figure 5.10. Distance and direction of the vector controls the actual movement in the virtual world.

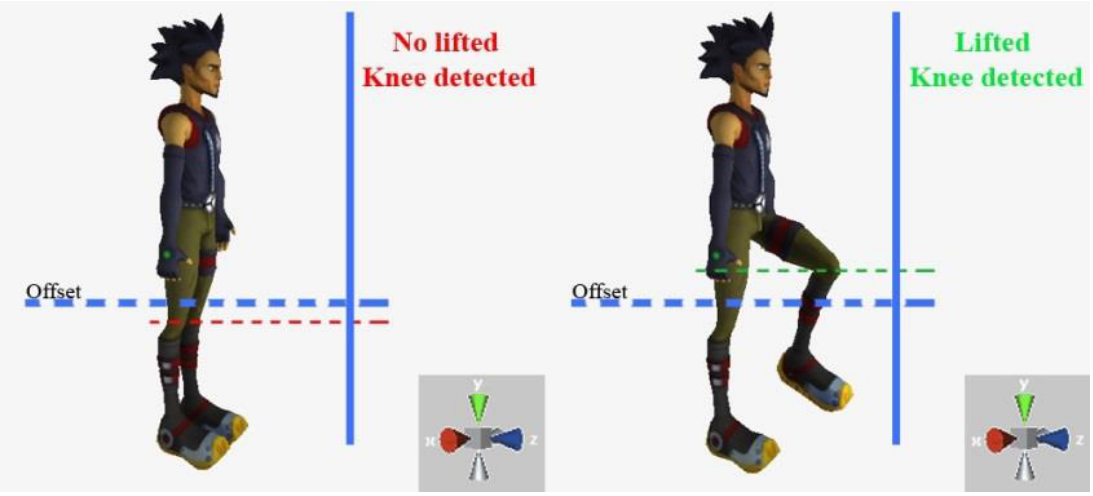


Figure 5.9: Walking-In-Place gesture for translational movement

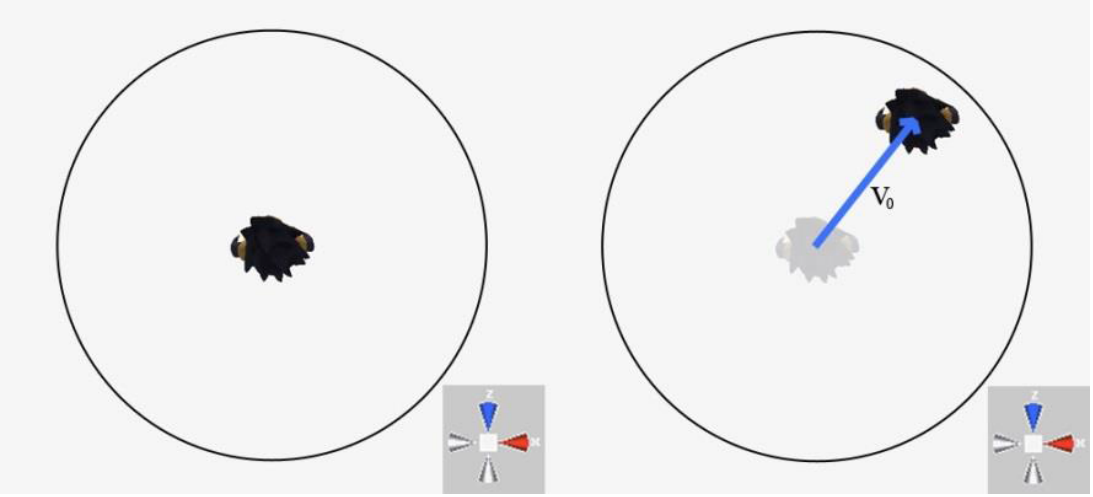


Figure 5.10: Distance-To-Velocity gesture for translational movement (Rump, 2014)

Potential for SafeChild

A similar display setup by Schwebel et al. (2008) was already successfully evaluated for the purpose of pedestrian safety training. In their publication, the authors claim that this semi-immersive setup provides a realistic view into the virtual world and also offers sufficient static visual stimuli to prevent (or at least reduce) feelings of nausea. In SafeChild, the gesture-based interaction was added to move and look freely in the virtual world. In order to evaluate the potential of these gestures, a small-scale user study was conducted by Rump (2014). Under the main hypothesis that a natural gesture should result in precise movement and low amounts of perceived cognitive load, the (highly diverse group of 22) experimentees had to complete a path following task and answer a questionnaire afterward. By using the NASA TLX Load Index (Hart & Staveland, 1988), Distance 2 Velocity produced the lowest cognitive load (38.61 of 100), followed by LEAN (48,98 of 100) and lastly Walking in Place (58,29 of 100). Without a baseline, these results can only be used to compare the gestures with each other and not to determine the general usefulness of any of these gestures for SafeChild. However, by looking at the logged paths of the participants, a high variance in the ability to accurately follow the given waypoints could be observed for all gestures. This shows that there could be a significant amount of people for whom this interaction

method is not natural and hard to master. Another factor against this semi-immersive setup is the high cost of the hardware (three stereoscopic monitors and a powerful computer to run the application). In conclusion, this setup does not fit a day-to-day usage scenario at regular schools or households in its current state. It has however potential, given an improvement on the gesture-based interaction, to serve as an alternative to fully immersive for users who are prone to nausea.

5.3 Fully Immersive Setup

In this setup, the user is fully immersed in the virtual world. The user becomes completely detached from the real world (at least visually), and his/her head movements are directly translated into the virtual environment. This provides a highly natural way to look around in VR that surpasses the semi-immersive setup. Thus, the user is also able to perform typical traffic safety tasks such as look left-right-left as in real-life. In order to walk around in the fully immersive variant of SafeChild, arm swings are registered using an extra arm worn device. The following section is based on joint research, mostly with the colleagues J. Orlosky and M. Weber (2015)

5.3.1 Display

The immersive display used for this setup is the Oculus Rift DK2 HMD (Figure 5.10), which provides stereoscopic images to the user at a resolution of 960 x 1080 pixel per eye. Although the grid between pixels is still visible, it is sufficient for the SafeChild use case since all relevant objects are large enough. The field of vision of the DK2 is roughly 100 degrees. The DK2 conducts six-degree-of-freedom low latency head tracking and can reach a refresh rate up to 75 Hz. Given a sufficiently powerful computer, there is almost no noticeable lag and therefore a low risk of nausea. Due to the narrow field of vision and additional tracking data provided by the HMD, it allows fairly accurate tracking of the user's field of vision in comparison to the semi-immersive or desktop setup. This information could be used in SafeChild to assess perception related skills more precisely. For instance, whether the user has looked at the right cars for the skill of ensuring that cars are stopping at regulated crossings. The other display option for fully immersive VR would be a Cave Automatic Virtual Environment (CAVE). These VR rooms use projectors to turn each wall into a screen to enclose the user in the virtual world. This option was not considered for SafeChild due to its impracticality (e.g. high costs and space requirement) to be used at regular schools or households.

5.3.2 Input

As in the semi-immersive setup, the user must have a method to engage walking in order to move in the virtual environment. To accomplish this in the fully immersive setup, the Myo armband was used. This armband contains sensors to measure movement and muscle activity (Figure 5.12). Since the device is arm worn, it can be used in a non-invasive manner and take advantage of the user's natural arm movements as a form of input. More specifically, the natural swinging of the arm is used to facilitate movement. Since a person's arms move naturally during walking, this becomes a much more familiar way of interacting than by pressing keys on a keyboard. Faster arm swinging corresponds to faster movement within the environment. In addition to walking, other interaction experiments were conducted. For instance, in order to physically interact with virtual elements in the simulated world (for example pedestrian light buttons),

Figure 5.11: Oculus rift DK2 HMD

the LEAP Motion sensor was used (Figure 5.13). This device was mounted to the HMD and provides near-field hand tracking and is ideal for on-demand interactions in the environment.

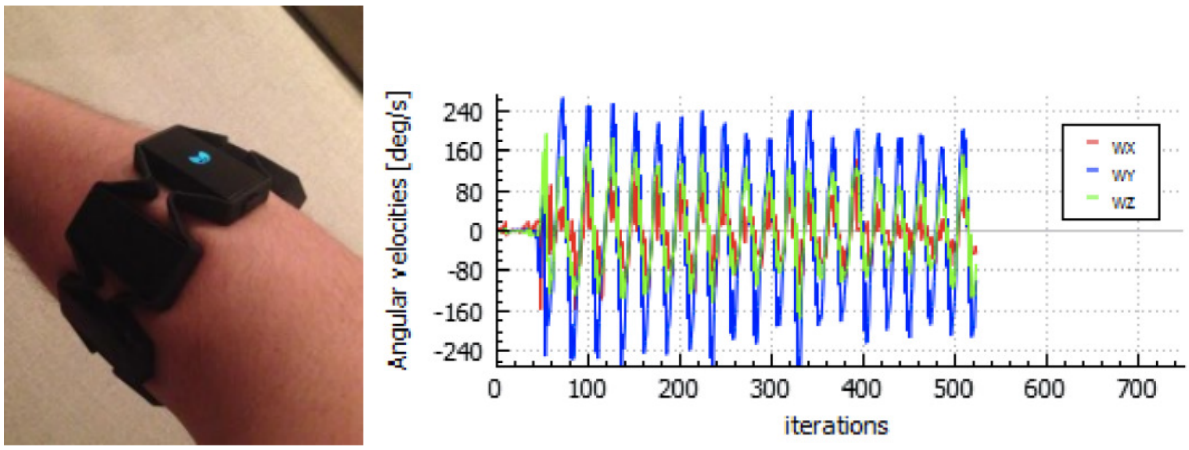


Figure 5.12: Myo wristband (left) with angular velocities during arm swing (right)

Figure 5.13: Hand model in VR obtained through LEAP Sensor.

The last input device added to SafeChild is a special SMI[5] eye tracking prototype that works inside the Oculus Rift. By using information provided by this device, the system is able to determine with high

[5]https://www.smivision.com/, accessed 12-08-2018

certainty whether objects (especially those that are relevant for training) have been seen by the user. This information is useful to track skills and/or to provide adaptive feedback and instructions based on gaze. Figure 5.14 shows an experimental exercise that requires the user to perform different tasks, including finding the traffic light button and checking traffic. Although eye tracking would be very useful for SafeChild, the high price and prototypical stage of the hardware would exclude important stakeholders such as regular schools or households. Another problem with the interaction methods mentioned above is the inability to walk in one direction while looking to another. This makes it impossible to perform tasks such as crossing the road straight while continuing to look for traffic.

Figure 5.14: Eye tracking in VR. The green circle indicates that the user has focused on the traffic light button. This is also reflected by the green highlight of the text in the top left corner.

5.3.3 Potential for SafeChild

The fully immersive setup of SafeChild provides, at least from a visual point of view, the most realistic and natural experience. The size ratio of the environment feels close to the real world and seeing an approaching car in stereoscopic 3D can cause real feelings of danger. One main problem with this setup is the lack of longtime research with HMDs and young children (especially with regard to potential harm). Oculus, for instance, recommends an age of 13 or older for their devices. This is above the age of elementary school children in Germany, who are the main target group of SafeChild. Other problems include the inability to walk and look in different directions at the same time and the high effort to install and run the application in the fully immersive setup. With the rapid development in this area at the moment of writing this thesis, it is nevertheless a highly promising approach for future pedestrian safety training.

5.4 Desktop Setup

The desktop setup for SafeChild relies on a standard monitor as display and a standard keyboard or gamepad as input device. Monitor and keyboard are mandatory hardware to operate a PC and, in case of notebooks, already built in. Therefore, this version of SafeChild can be installed and run in the same manner as normal desktop applications. In addition, it could even be provided as a web application and

thus run inside a web browser without the need of an installation process at all (given the browser is properly setup). With this ease of deployment and the fact that no special hardware is required, the desktop version of SafeChild has the potential to be disseminated to a great number of stakeholders.

5.4.1 Display

The environment is shown on a single 2D monitor using a perspective camera and a field of view of 60 degrees. By using a standard 16:9 aspect ratio, this yields an image that feels natural without too much distortion at the edges of the image (Figure 5.15). Although a 2D monitor does not provide stereoscopic 3D, the distances of objects can be estimated by their size.

Figure 5.15: Single screen first-person perspective using a 60 degree FOV on a 16:9 aspect ratio

5.4.2 Interaction

Interaction in this setup requires a number of buttons that can be either mapped to keys on a regular keyboard or buttons on a gamepad. As shown in Chapter 7, a gamepad is more convenient to use for young children. In total, six buttons are used for movements in the virtual world. Two buttons for moving forward or backward, two buttons for turning left and right and finally, two buttons for turning the head left and right. Head-turning and body-turning are separated to enable the user to walk in one direction while looking in another. This is mandatory to fulfill the skill of crossing straight while observing cars (Chapter 4). In terms of implementation, head and body-turning are different as well. The body-turning keys slowly turn the user's virtual body while they are pressed and keep the body's rotation after being released. This enables the user to turn to the desired direction with high precision. The head-turning keys on the other hand quickly turn the field of vision by 90 degrees and snap back to the original direction after being released. Thus, the user is able to quickly look left and right, even while walking. The standard key mappings are depicted in Figure 5.16.

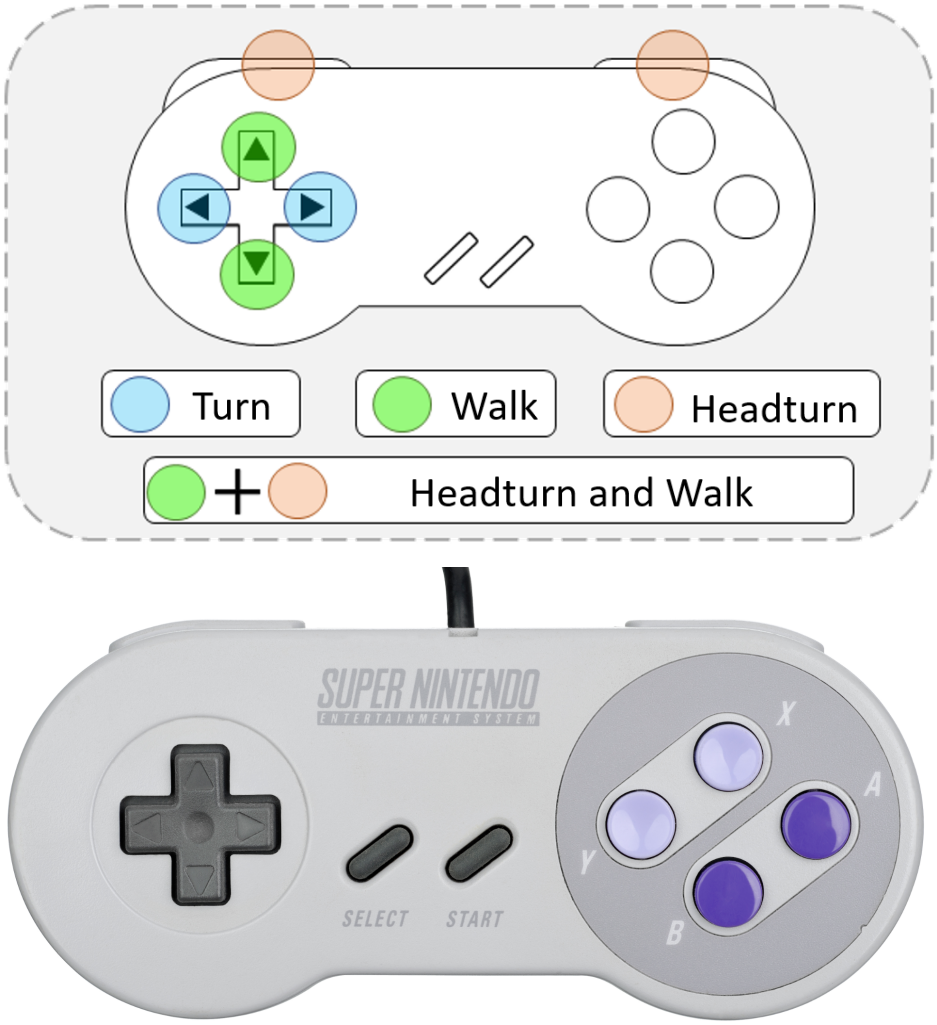


Figure 5.16: Gamepad controls. The top figure shows the key assignment while the bottom image shows a photograph of the actual gamepad used for the study in Chapter 7.

5.4.3 Potential for SafeChild

Based on discussions with elementary school teachers and parents, it is still not feasible (as of 2017) for regular German schools to equip their classrooms with the hardware required for semi or fully immersive VR training. This leaves the desktop version as the only realistic option for large scale usage of SafeChild (or similar VR training applications) in the near future. Therefore, this setup was selected for user studies mentioned in Chapter 7. Although it lacks in realism in terms of visual presentation and control compared to the other setups, it is still superior in terms of interactivity and level of engagement compared to video or text-based education. Further, preliminary studies by Schwebel et al. (2014) using a similar setup had promising results as well. In summary, the desktop-based interface setup has the greatest immediate potential for SafeChild. Schools and households could benefit from this type of VR education right away and transition to more immersive interfaces once the hardware becomes a commodity.

5.5 Educational Instructions and Feedback

As mentioned in Chapter 2, learners in an open training environment need appropriate guidance to focus on learning tasks. Therefore, educational instructions and feedback are required. In SafeChild, information about the learner's current performance in an exercise is provided through the Behavior Tree based Student Modeling framework described in Chapters 3 and 4. This includes the current task as well as successful or failed application of related skills. Since the desktop setup of SafeChild was the only one used for studies with the target user group, the respective UI elements were only implemented for this setup. Both instructions and feedback are implemented in a straightforward text-based way. Although VR would allow more immersive ways of educational interventions including voice and gestures, they bring additional effort, challenges and research questions that exceed the frame of SafeChild and this thesis. Instead, the goal was to determine how effective text-based instructions and feedback would be for this specific use case. The results should then serve as a baseline for future work in this direction.

5.5.1 Instructions

The system is able to provide instructions about what to do next. For instance, the system would tell the user to wait for green when s/he prepares to cross the road at a traffic light. In case that there are multiple actions to perform within a task (or technically speaking, multiple constraints that are active in parallel), they are sorted by their common order (for instance for traffic lights, the pedestrian should normally first wait until the light is green and then check whether the cars have stopped. It is however not a strict order since seeing that the cars are stopping first and then checking the light would also be acceptable). Then, only the topmost action is shown to reduce the time needed to process this information. This filtering is primarily done to reduce the amount of text on the screen since reading is a challenging task for young children. The background for the instructions is a light blue panel with a thick black border that sets itself apart from the rest of the screen. On that background is a white arrow symbol and bold black instructional text (Figure 5.17).

Figure 5.17: Instruction in SafeChild. The text reads "continue observing traffic" in German.

5.5.2 Feedback

Feedback is provided to the users to inform them about their performance. In SafeChild, there are two variants of feedback that differ mainly in timing. While the immediate feedback variant would directly inform the user whenever s/he succeeds or fails a skill, the delayed version waits until the end of an exercise before providing a list of results for all skills related to that exercise. A discussion about the advantages and shortcomings of both feedback variants from an educational point of view is provided in Chapter 7.

Immediate Feedback

In this feedback variant, the feedback occupies the same area on the screen as the instructions shown in Figure 5.17. As soon the user successfully completes the task shown currently in the instructions, the panel turns to green, the symbol turns from the arrow to a green happy smiley icon. This success panel would flash up and then fade. Subsequently, a new instruction will be shown in that place. In case the user fails a skill, the panel turns red, and a sad smiley icon appears instead (Figure 5.18).

Figure 5.18: Immediate feedback. The instructions shown in Figure 5.17 transforms to the left image when the user has correctly waited for the light to turn green and to the right one if s/he failed to do so and entered the road prematurely.

An important difference between success and error feedback is that success is only shown for skills that are being displayed as instructions. As mentioned before, several skills could be required at the same time. If the user completes another skill that is not currently displayed (for instance making sure that cars are stopping while waiting for green), the success panel for that skill will not pop up. This was done to prevent confusion and disruption of the exercise flow since the text would change (and the panel would flash) very often. The same does not hold for the error panel which is shown whenever an error is detected. In case that it is not the same skill as currently displayed as instruction, it would overwrite the text and then stay visible for a longer time (Figure 5.19). Being aware of errors made is critical to learning and are thus always shown, even if it disrupts the flow of the exercise.

Delayed Feedback

In the delayed feedback mode, no error or success panels are shown at all during an exercise. Instead, all skills that were relevant during an exercise are annotated with success or failure and listed after the completion of an exercise. This occurs either when the user reaches the goal or when s/he comes too close

Figure 5.19: Error panel pop-up. The left image shows the instructions to *keep observing traffic while crossing*. If the user stops mid road and thus failing the skill of *cross without stopping*, the instruction will be replaced by a red error panel that shows the message of not to stop while crossing.

to a moving car on the road. Figure 5.20 shows an example of delayed feedback after the completion of a traffic light exercise.

Figure 5.20: Delayed feedback. This panel is shown after a traffic light exercise and indicates that the user stopped at the curb, waited for green and continued to observe traffic while crossing (items 1,2 and 5 on the list). It also shows that the user has failed to ensure that cars are stopping and also made the mistake of stopping while crossing (items 3 and 4). The button on the bottom of the panel would lead the user to the next exercise.

5.6 Chapter Summary

This chapter introduced SafeChild's VR based UIs. This includes the Virtual Environment itself (consisting of an urban neighborhood with roads and traffic) as well as three different interface setups: desktop, semi-immersive and fully immersive. Based on their respective advantages and shortcomings, the desktop setup was selected as the main platform for the user studies presented in Chapter 7 due to its low cost and ease of installation. In combination with SafeChild's Student Model, instructions and feedback are

Figure 5.21: Virtuix Omni (Goetgeluk, 2013) 360 Degree Treadmill

provided in textual form. With regard to feedback timing, SafeChild supports immediate as well as delayed (end of exercise) feedback. An evaluation of their effects on learning is provided in Chapter 7.

UIs, especially those for VR, are evolving rapidly under the influence of the gaming industry. As a consequence, more realistic virtual worlds can be created with less effort and resources. It is both a great opportunity and challenge to translate these improvements in technology to better learning results through close to real life practice and training. In terms of locomotion within immersive VR, stationary 360 degree treadmills such as the Virtuix Omni (Figure 5.21) could provide a natural walking experience that allows the user to walk in one direction while looking to another. At the same time, higher resolutions and portable light-weight solutions for fully immersive VR might make fully immersive VR appropriate for children in the near future.

Finally, appropriate pedagogical instructions and feedback are required to prevent the learners from getting lost and to keep their focus on the learning task. In order to fully benefit from the potential of VR, future research should focus on immersive and natural educational intervention such as voice and gestures of fully embodied virtual teachers or peers.

Chapter 6

SafeChild Implementation

This chapter presents the implementation of SafeChild. Amongst others, it describes how the Behavior Tree based Student Modeling concept from Chapters 3 and 4 as well as the different VR interfaces from Chapter 5 are technically realized. The chapter starts with a discussion of requirements and design decisions. Afterward, key frameworks and tools are presented followed by an in-depth look at the system's architecture. The chapter wraps up with a section dedicated to the anatomy and setup of exercises within SafeChild and the chapter summary.

6.1 System Requirements

The requirements to the SafeChild system are derived from the use cases. Users (most importantly children) should be able to freely navigate in an immersive, interactive and real-time 3D environment for traffic safety training, including crossing at pedestrian light, zebra crossing, and unregulated crossings. This environment needs to represent an urban setting with traffic, which includes moving cars and crossing related signs and signals. In addition, the user should get educational instructions and feedback in a fine-grained and adaptive way based on the user's actions within the context of a training task and the state of the environment. Since SafeChild should not only serve as a training application but also as a research and analysis tool, additional use cases have to be supported. This includes logging user performance with the ability to replay them as well as the ability to set up system parameters such as different instructional models. The use cases are summarized in Figure 6.1.

6.2 Design Decisions

There are a number of design choices to be made for the implementation of a system that has to satisfy the requirements stated above. First of all, the decision needs to be made about how to realize the interactive 3D part of SafeChild. The options range from low-level 3D application programming interfaces (APIs) such as OpenGL[1] to a great number of 3D frameworks and engines as well as editors for extending existing 3D games. Both extremes were not considered for SafeChild. Since SafeChild does neither target the highest graphical quality nor the best graphical performance, the immense programming effort to work with a low-level API would not be justified. On the other hand, editors for existing games are too

[1] https://opengl.org/, accessed 13-08-2018

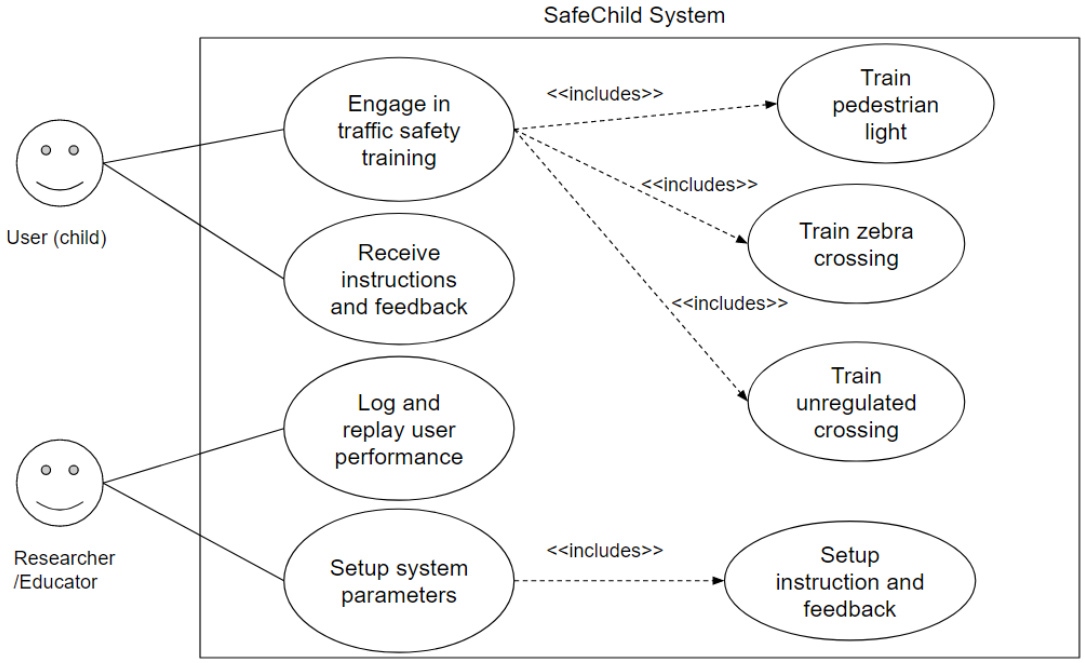


Figure 6.1: Use-case diagram

restrictive to support the research and experimental aspects of the system. Eventually, a game engine was chosen as the main development platform for SafeChild (presented in the section below). It combines a rich toolset for composing and deploying interactive 3D software with a programming interface for customization of existing and addition of new functionalities. The next design choice is how to fill the game engine with 3D content. One possibility would be to build the models from scratch with full control over the models' looks and geometry. Another one is to get or buy models from third-party that fits the setting. To save time and effort, the second option was used for most of the models in SafeChild, and only those that could not be found from other sources were created from scratch. The third design choice was about traffic simulation. Again, the question was whether to use a third-party implementation or to develop a new one. Here, the latter was chosen due to the fact that full control over the traffic simulation was required. Last but most importantly, the design decision about student and domain modeling for the intelligent tutoring part needed to be made. The reasons for the decision here to use Behavior Trees (BTs) as the main data structure for this purpose was discussed in detail in Chapter 3.

6.2.1 Key Frameworks and Tools

This section presents the specific frameworks and tools used for the implementation of SafeChild based on the design considerations above. In particular, this includes the game engine, the programming language, and various tools for 3D and 2D content creation.

Game Engine

The game engine used for SafeChild is Unity[2]. It includes a visual editor for the composition of 3D scenes, allows live preview and debugging of the application and provides a wide range of game development tools. This includes animation, lighting, user interface (UI) creation, and physic simulation. On top of

[2]https://unity3d.com, accessed 19-04-2018

that, Unity comes with an asset management system. Assets are for instance 3D models, 2D images, sound files, scripts or bundles of those. Unity provides mechanisms to import and export assets and an asset store that contains a great number of (commercial and free) ready to use content. This makes it easy to outsource time-consuming content creation and enables developers (especially small teams) to focus on program logic. The two main building blocks of a Unity scene are game objects (GOs) and components. GOs represent entities in a scene and are organized in a hierarchical way. For instance, there could be a car GO with four child GOs for the wheels. Components, on the other hand, represent attributes or functionality that can be assigned to a GO. Each GO has at least one transform component that specifies its 3D position, rotation and scale in the scene with respect to its parent GO (or globally, if the GO is at the root of the hierarchy). Other components include colliders for physics calculation, mesh renderers to display 3D models or custom behavior scripts. Figure 6.2 shows an Overview of the Unity editor's user interface (UI).

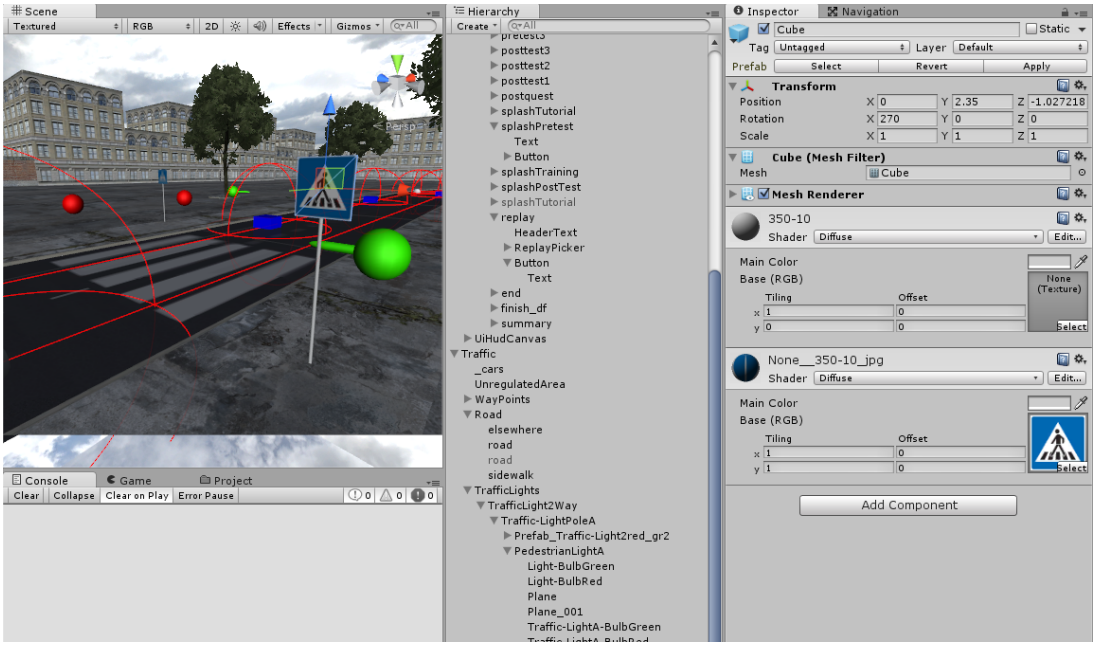


Figure 6.2: Unity editor

6.2.2 Programming

The above-mentioned behavior script components in Unity grant access to a wide range of functionalities and allows (amongst others) to manipulate the start and update behavior of a GO. These scripts are created by directly writing program code. The choice of programming language is restricted by Unity. More specifically, Unity supports Javascript (in a customized version), C# and Boo. For SafeChild, C# was chosen mainly due to the familiarity and preference of the developers. In terms of functionality within Unity, the other languages could produce the same result. Unity provides a custom version of MonoDevelop[3] as default integrated development environment (IDE) for programming. Other editors such as Visual Studio or Visual Studio Code from Microsoft[4] could be used as well.

[3] https://wwww.monodevelop.com, accessed 22-08-2018
[4] https://visualstudio.microsoft.com, accessed 22-08-2018

6.2.3 Content Creation

Besides programming, a major part of an interactive 3D application such as SafeChild consists of 3D models. Although the majority of models for SafeChild was obtained through the above-mentioned Unity asset store, not all required models could be found there. Especially traffic signs and signals such as the zebra crossing sign needed to be created. For this purpose, the open source 3D modeling software Blender[5] was used. Blender allows mesh and vertex based 3D modeling as well as UV mapping and animation. Another benefit of using Blender is the compatibility with Unity, which is able to directly read and import native Blender files. Figure 6.3 shows a screenshot of Blender's UI.

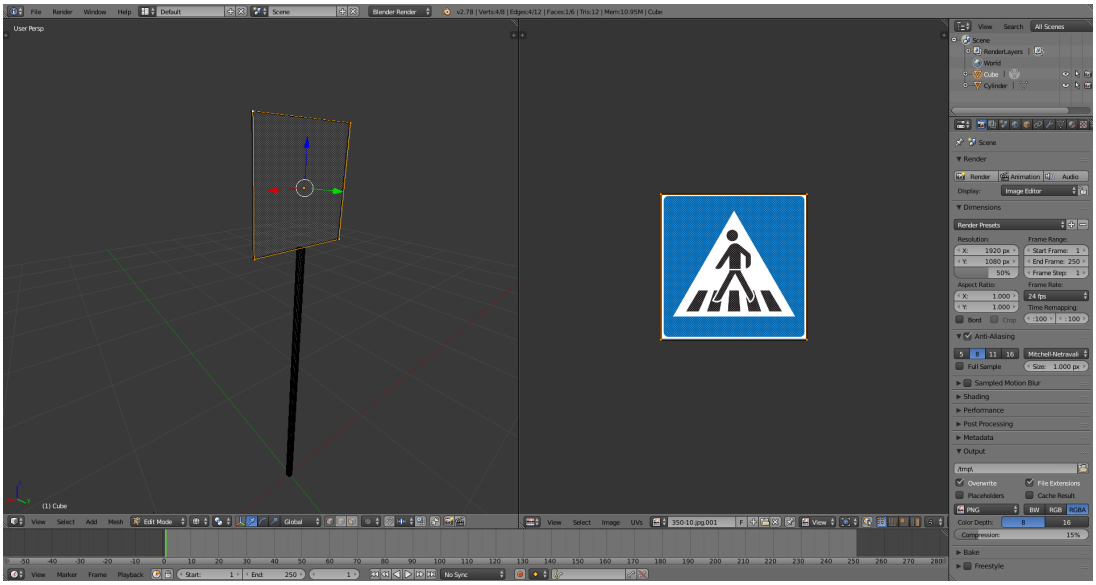


Figure 6.3: Blender 3D modeling software

In addition to creating 3D models, 2D images are required as well for texturing the 3D models and 2D UI elements. In order to create and edit these 2D images, the (also freely available) program Paint.net[6] was used. This program provides standard image editing features including scaling, moving, cutting and turning selections of an image and also more advanced features such as layer-based editing, adjusting color channels and finally, the import and export of different file formats. Figure 6.4 shows a screenshot of Paint.net. Blender and Paint.net are not the only options for 2D and 3D content creation. Instead, there is a wide range of software for this purpose with different feature sets and different costs. For a research project such as SafeChild, the choice of software was however fully sufficient.

6.3 System Architecture

SafeChild's system architecture consists of six major modules that are represented by top-level GOs in the hierarchy of a single scene. This comprises of the scene manager module, the 2D UI module, the player module, the traffic module, the environment module and finally the Behavior Tree (BT) tutor module. The scene manager module is at the core of the architecture and serves as glue and hub of the other modules. It bundles administration and control functionalities that affects the entire scene. The UI module lies

[5] https://www.blender.org, accessed 22-08-2018
[6] https://www.getpaint.net/ , accessed 22-08-2018

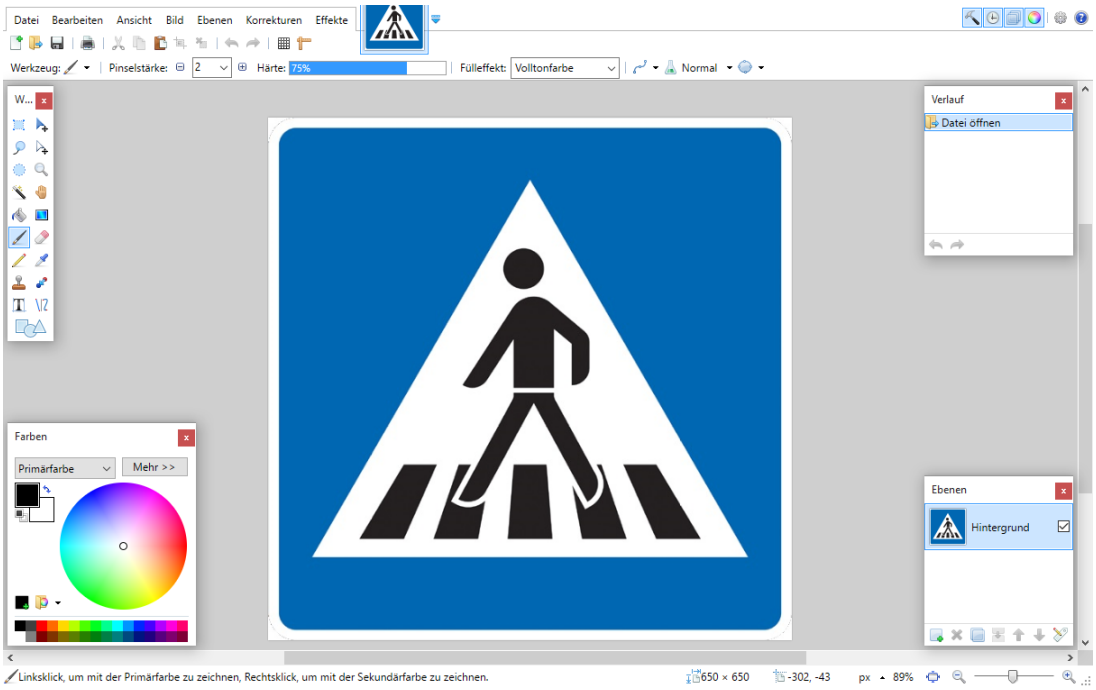


Figure 6.4: Paint.net image editing software

between the user and the scene manager. It is responsible for directing user input on the 2D UI (e.g. button clicks) to the scene manager and to present 2D information to the user. Also between the user and the scene manager lies the player module. This module processes input that directly affects the user's current location and view direction in the virtual world (e.g. gamepad input). It is then responsible for updating the scene accordingly. The traffic module is not directly affected by the user and contains models, artificial intelligence (AI) routines and logic for all traffic-related entities including cars and traffic signs. Further, the environment module aggregates the remaining 3D models and lighting to complete the visual representation of the scene. Finally, the BT tutor implements the Domain and Student Modeling concepts of Chapters 3 and 4 to provide intelligent tutoring capabilities. Figure 6.5 shows a top-level overview of the architecture with managed interfaces. An interface is considered as managed if they contain SafeChild specific functionalities. On the engine level, all GOs are connected and exchange data (e.g. for physics calculations). The architecture figures in this section are supposed to show the structural concept of each module and thus do not include all details for the sake of readability.

6.3.1 Scene Manager

The scene manager module contains several behavior scripts to setup and administrate variables that affect one or multiple other modules. This includes traffic variables such as average car speed and traffic density or environment variables such as weather conditions and lighting. The logging and replay related functionalities are attached to the scene manager as well. In order to log the user's performance, the position and rotation information from the player module is written continuously into a log file together with events from the BT tutor. This logfile can be either stored in an online database or offline as JSON file on the hard disk. Since the log entries contain timestamps, the performance of a user can be replayed by applying the position and rotation to the player module at the right time (and interpolating between them). For the cars, it is sufficient to save the time when they were generated together with their maximum speed.

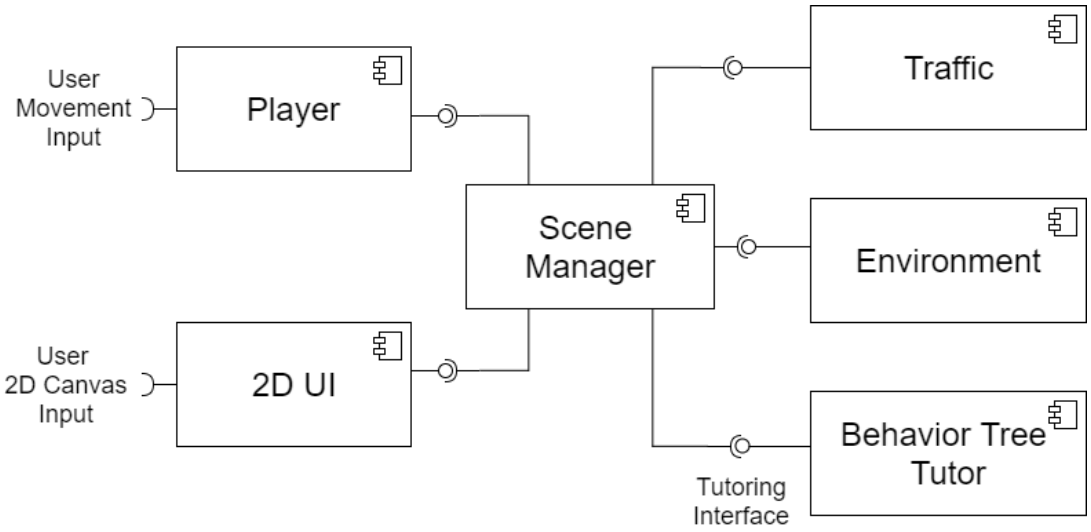


Figure 6.5: UML Component Diagram of SafeChild top-level architecture (the term component refers to the type of UML diagram and not to Unity components.)

In addition to the script components, the scene manager also includes the exercise module as a child GO. This module is responsible for loading and switching between different exercises. This includes setting the start and goal position at the beginning of an exercise as well as loading exercise specific objects. This includes view obstacles, such as parking cars, and exercise borders that restrict the area that the user is allowed to move in. Each specific exercise is implemented as a separate GO and attached as child to the exercise module. Figure 6.6 depicts the architecture of the scene manager module.

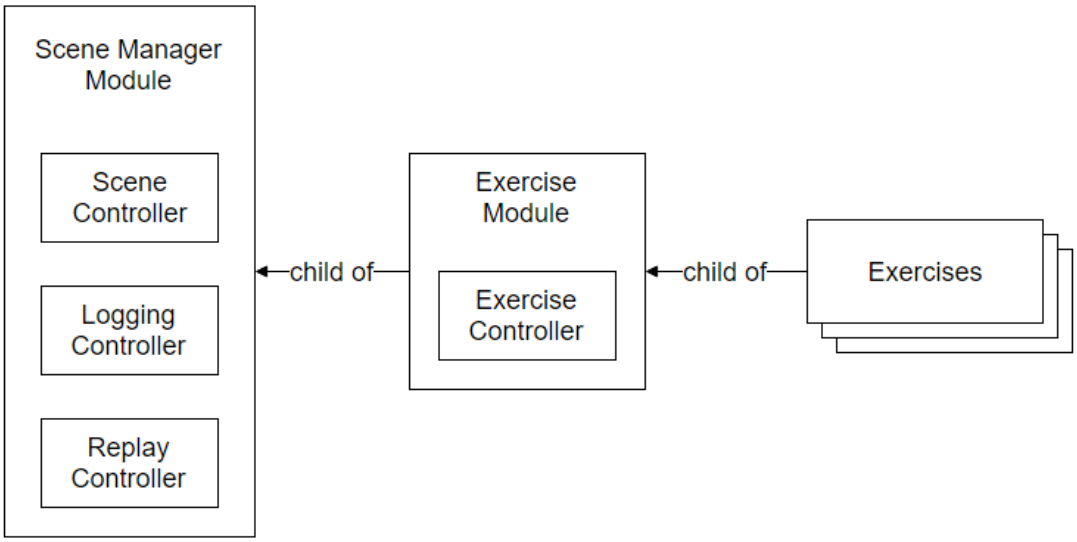


Figure 6.6: Scene manager architecture

6.3.2 2D UI Module

The 2D UI module draws and manages 2D UI elements such as panels, text, and pictures that reside on a so called canvas, which is the native container of Unity for 2D UI content. This canvas is rendered by a dedicated UI camera and overlaid above the rendering of the 3D world. The 2D UI consists of the

two parts, head-up display (HUD) and menu. While the HUD shows information during runtime of an exercise (e.g. instructions or immediate feedback), the menu elements are only shown when the exercise is paused and occupy the entire screen. This includes panels that show the controls, summarizing feedback or disclaimers. Figure 6.7 shows the composition and hierarchy of this module and Figure 6.8 shows examples of HUD and menu in SafeChild.

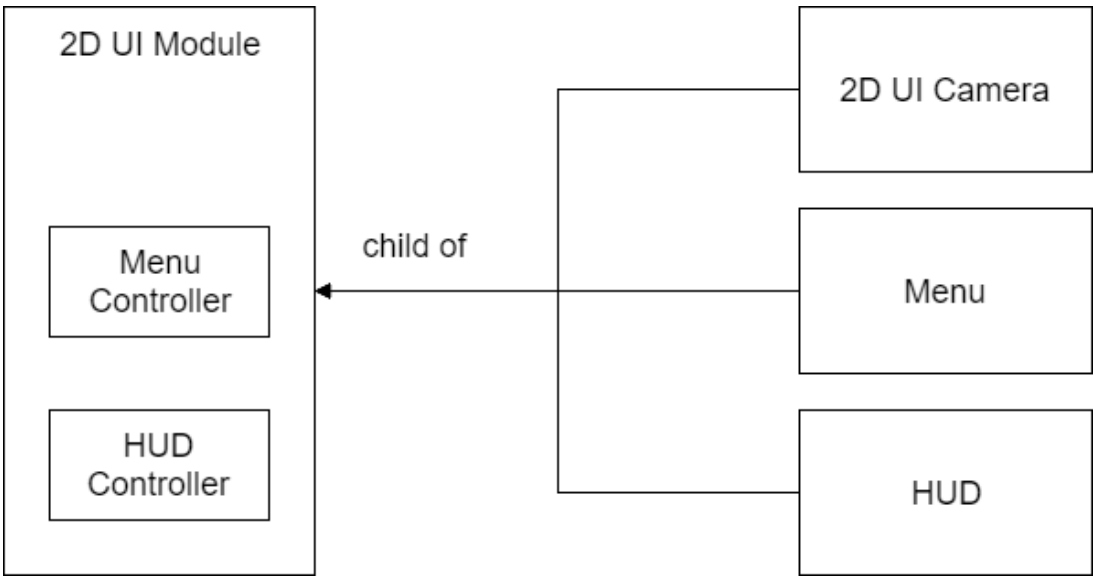


Figure 6.7: 2D UI module architecture

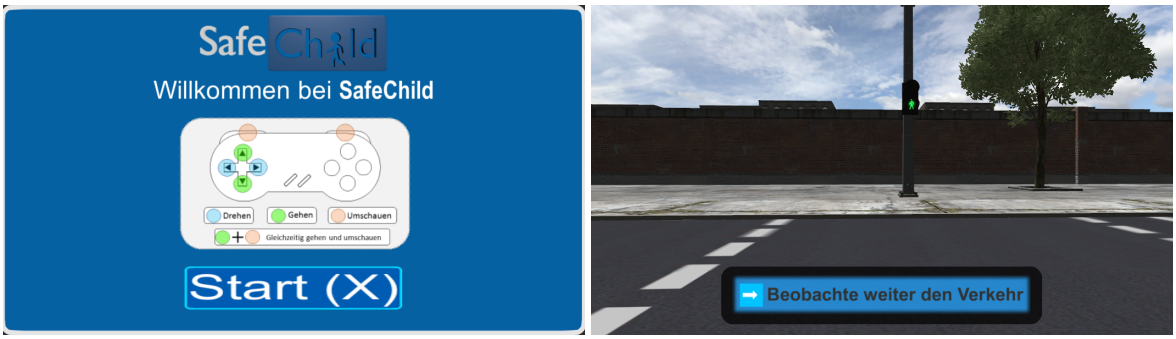


Figure 6.8: 2D GUI in SafeChild. The left-hand side shows a menu panel that explains the controls while the right-hand side shows a HUD panel that is overlaid above the 3D view of the scene.)

6.3.3 Player Module

The player module manages the user's virtual presence in the artificial 3D environment. User input for locomotion and head-turning are translated (in near real-time) into positional and rotational changes of this module. The player module has a vision sub-module as child GO which contains a camera component that moves with its parent. This camera is responsible for rendering the main 3D view of the scene. The vision sub module also contains a perception script which uses a technique called ray casting to approximates the visibility of traffic relevant items in order to estimate whether the user could have seen it. To do that, virtual rays are cast from the user's position to see whether they intersect with the corners and center of the target object's bounding box. This method is not exact and could cause problems for very large

objects or objects that deviate strongly from a rectangular form. For the use case of SafeChild however, most relevant object (signs, traffic lights, and cars) are close to rectangular, and the method turned out to be efficient enough to also run lower end machines. In the future, this method could be replaced by accurate eye tracking based methods when hard- and software for this purpose becomes more common. The player module also incorporates the user's physical presence using a so called collider component. This collider allows the system or other objects (such as cars) to detect collision with the player module. In order to switch between those different VR setups discussed in Chapter 5, only this module has to be changed. The subsequent figures show the functionality of the player module at runtime (Figure 6.10) and its architecture diagram (Figure 6.9)

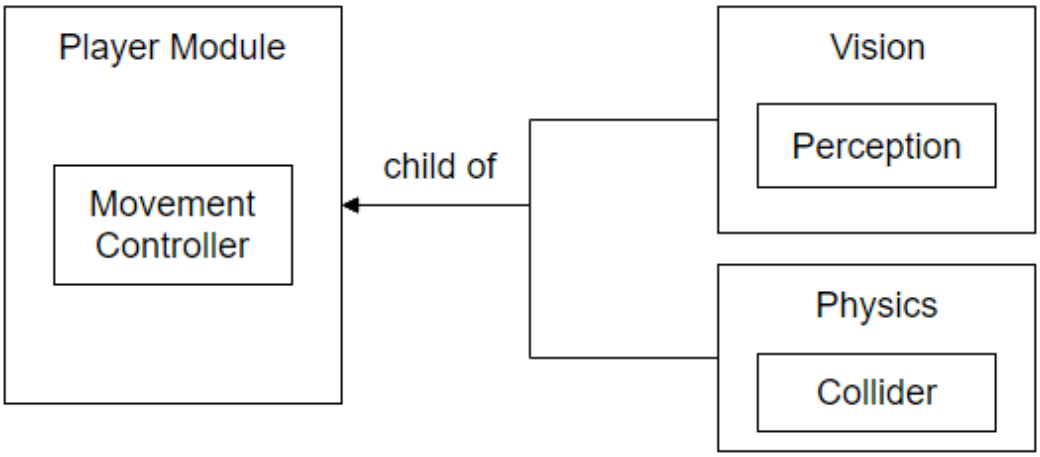


Figure 6.9: Player module architecture

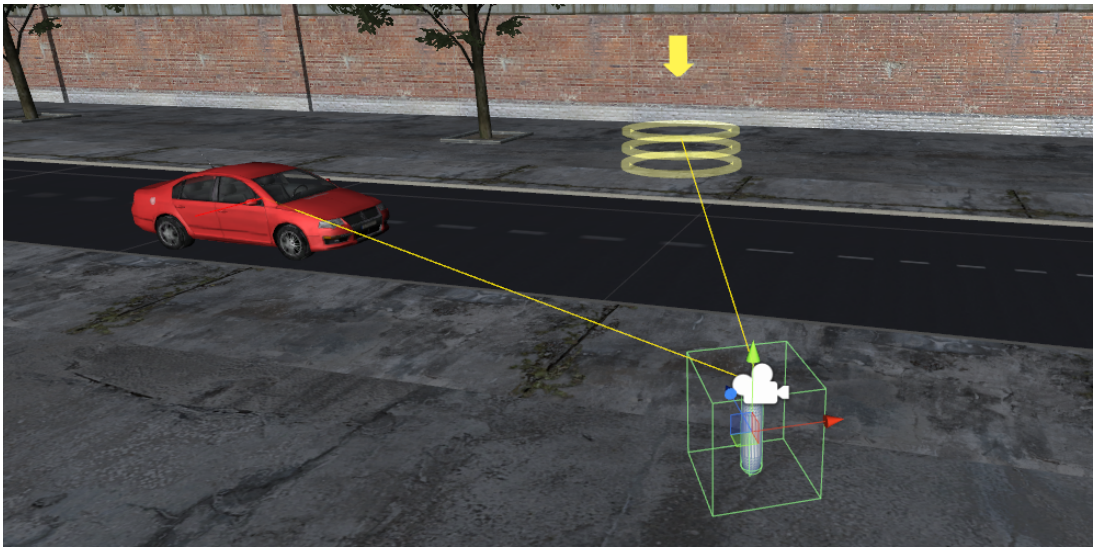


Figure 6.10: Player module at runtime. The player module is shown within the rectangular box for collision detection. The rays shooting out from player to car and goal object symbolize the raycast based perception calculation. Only the center ray is drawn for debug purposes in the screenshot although additional rays are cast towards the corners of each relevant object as well.

6.3.4 Traffic Module

The traffic module aggregates all GOs and functionalities for the traffic simulation in SafeChild. This includes cars, roads, traffic lights, zebra crossings, and waypoints. Waypoints form a directed graph that spans across the road and are used to guide and control cars (Figure 6.11). Under normal circumstances, the car steers from one waypoint to another while trying to keep its desired speed, which is set randomly between two thresholds (in the study described in Chapter 7, the thresholds were 30 and 40 kilometers per hour). Waypoints also control the creation and deletion of cars. This means that a waypoint can generate cars at their position or remove cars from the scene when reached by them. Thus, the section on the road where traffic is active can be changed for different exercises to prevent calculation overhead from cars that are too far away and thus not relevant for training. In addition to the functionalities above, waypoints can also instruct a car to stop at their position. This is used to implement the traffic light and zebra crossing functionalities. For traffic lights, a car is instructed to stop if the light is yellow or red and for zebra crossings, if the user is nearby. Each car is steered by an AI behavior script that moves the car based on Unity's wheel collider system. This system simulates physical wheel movement based on variables such as friction and torque. These parameters were tuned in SafeChild to resemble typical traffic behavior. As a consequence, cars brake and accelerate in a realistic way. If a waypoint instructs the car to stop, but the braking distance is too short, the car will not brake but continue to drive. The threshold for this is calculated based on the following approximation formula for emergency braking taught in German driving schools[7].

$$(\frac{\text{speed in km/h}}{10} * \frac{\text{speed in km/h}}{10})/2$$

As in real traffic, this can lead to cars running over a yellow light. In addition to following waypoints, cars use ray casting to prevent hitting cars ahead of them (they will adapt their speed instead) and to detect whether a collision with the user is inevitable. Figure 6.12 shows the traffic module's architecture.

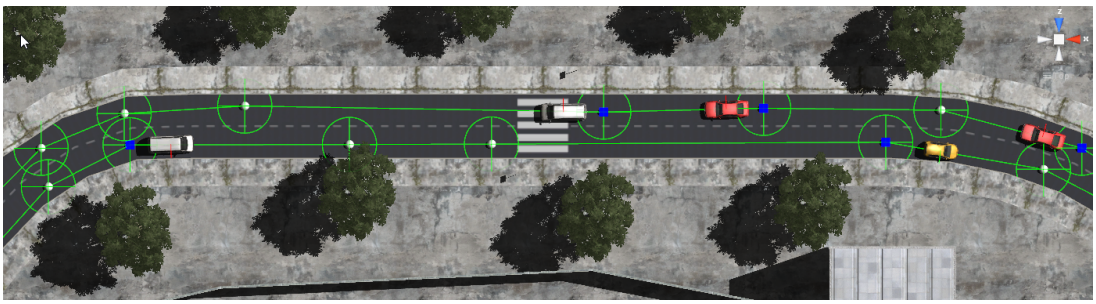


Figure 6.11: Waypoints for traffic simulation depicted as green circles

6.3.5 Environment Module

The environment module is essentially a container for those entities of the scene that are static in terms of their position in the 3D world. It includes the 3D models of buildings, vegetation, and the road. In addition, it also incorporates a lighting GO that represents a virtual sun. Altogether, the environment creates a urban look and feel at daylight. This module does not make use of any custom behavior scripts and its hierarchy is depicted in Figure 6.13.

[7] https://www.stvo.de/79-information/305-fuehrerschein-regeln-faustformeln, accessed 22-08-2018

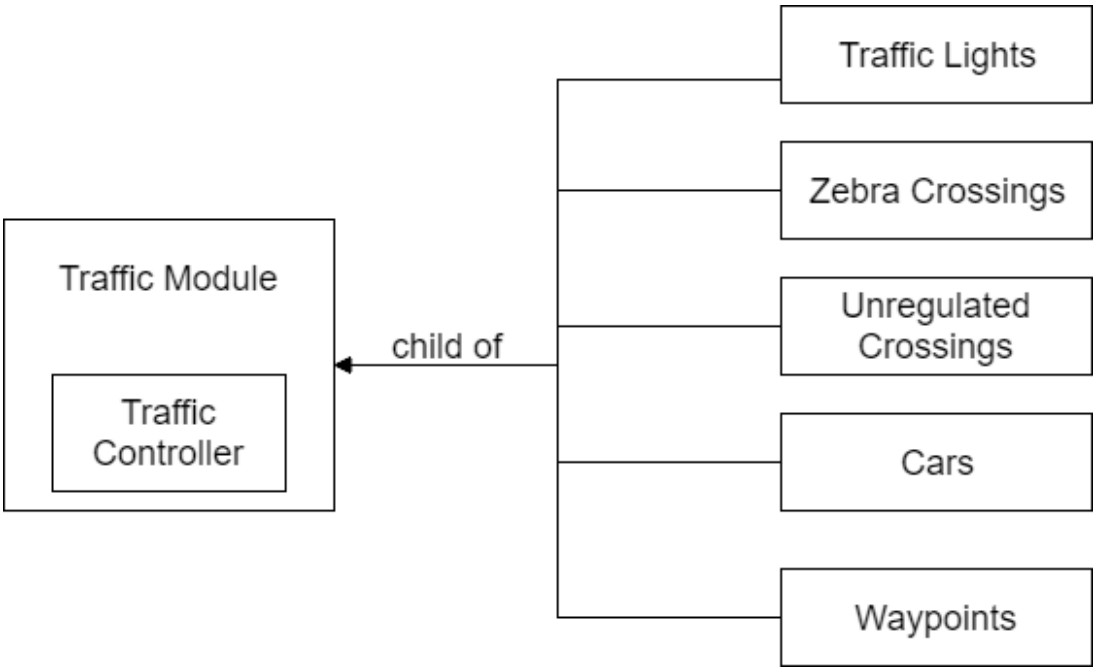


Figure 6.12: Traffic Module Architecture

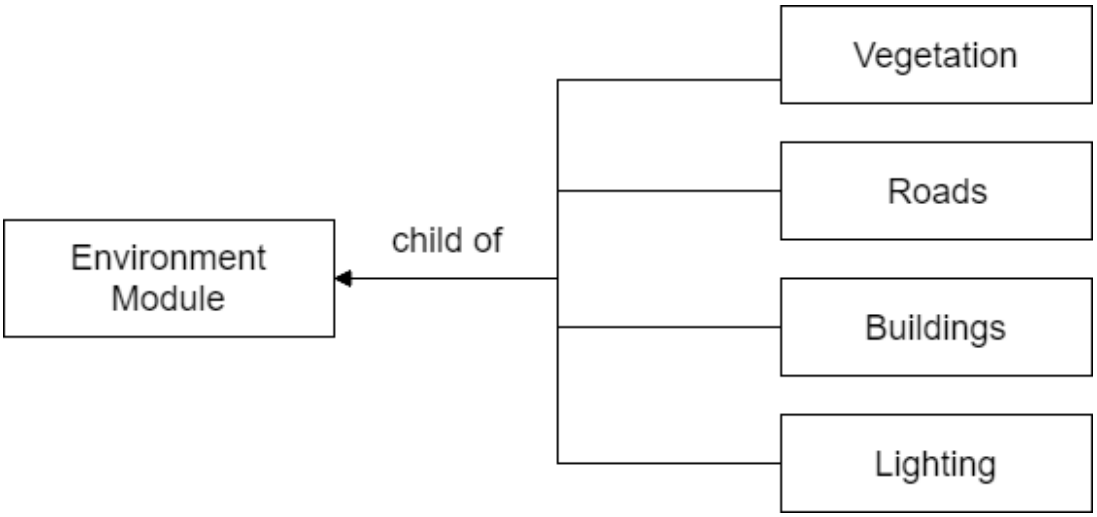


Figure 6.13: Environment module architecture

6.3.6 Behavior Tree Tutor module

The behavior tree tutor module is responsible for providing adaptive instructions and feedback based on the current task and the actions of the user. It implements the concepts described in Chapter 3 and thus follows a similar structure. On the top-level, the module contains the tutoring manager and the world state adapter as custom script components. The tutoring manager provides high-level tutoring related functionalities to the scene manager module based on information gathered through Student Modeling. This includes querying the currently relevant instructions and the performance of previous tasks. In addition, it also offers setup options for the behavior tree tutor module including the timing and mode of feedback (see Section 5.4). Although the educational functionalities of SafeChild are still limited, the tutoring manager script corresponds to the pedagogical model of a typical Intelligent Tutoring System (ITS) architecture. The world state adapter draws information about the scene (player, traffic, environment)

from the scene manager module to determine or calculate the training relevant world states. This includes for instance the shortest distance of the player to a moving car and whether a crossing is required to reach the goal (calculated based on navigation meshes). One level further, there are two child modules, one for constraints and one for the Expert Behavior Tree (EBT). Each constraint in SafeChild is implemented as a custom script component of the constraints module. The EBT module, on the other hand, serves as the root of the EBT implementation. Each node is represented by a GO with a BT node script, and the EBT structure is constructed using the scene hierarchy. As mentioned in Chapter 3, the EBT corresponds to the Domain Model of an ITS. Since these GOs do not have geometry or rendering components, they do not noticeably affect the graphical performance of the system. The EBT manager script component of the EBT module implements the traversal and update algorithms presented in Chapter 3. In addition, it registers and processes state changes and emitted events by the EBT nodes at runtime. Since this information essentially represents the student's current performance, the EBT manager can be seen as runtime Student Model of an ITS. The last of the four typical ITS models, the Interface Model, is the entire interactive 3D environment implemented through the other modules of in SafeChild's architecture. The following diagram (Figure 6.14) shows this module's architecture.

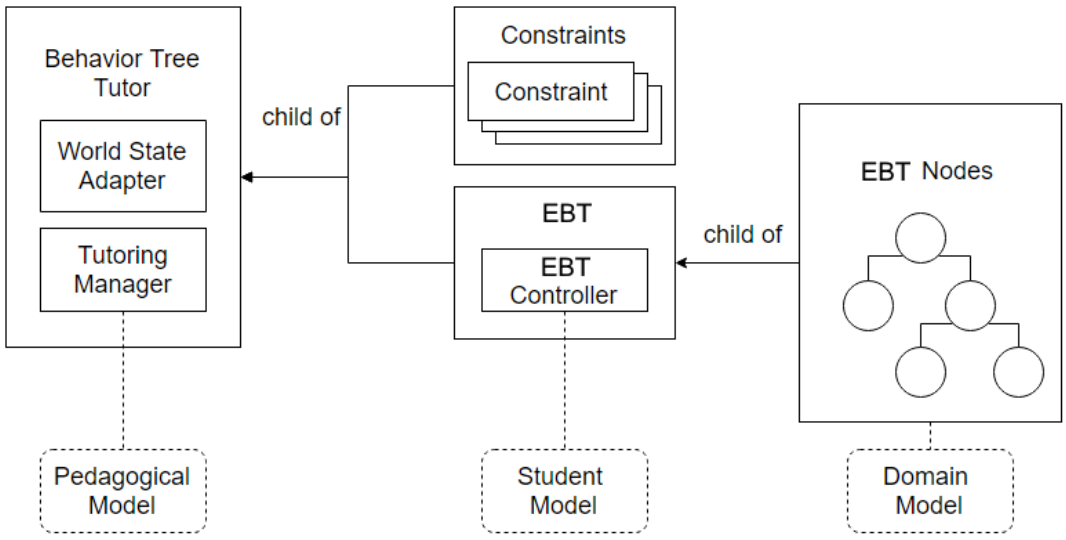


Figure 6.14: Behavior Tree Tutor architecture

6.4 Exercise Anatomy

This section describes the implementation of traffic safety exercises in SafeChild. Each exercise has the same basic structure and only differ in start position, goal position, exercise borders, exercise waypoints and (optional) exercise models (Figure 6.15). When an exercise is loaded, the user is placed at the start position, and the goal indicator object is set to the goal position. In addition, the exercise waypoints, exercise borders and exercise models are activated. As a consequence, traffic will start in the desired road section, the boundaries in which the user can freely navigate will be defined, and additional objects such as parking cars as view obstacles will be loaded. The user's task is now to navigate to the goal position safely. At the same time, the BT agent will be activated to assess the user's actions and to give instructions and feedback. Since the BT is designed in a generic way to handle all types of crossings, no

configuration is required as the BT agent will adapt automatically to the task at hand based on the user's and the environment's state (see Chapters 3 and 4).

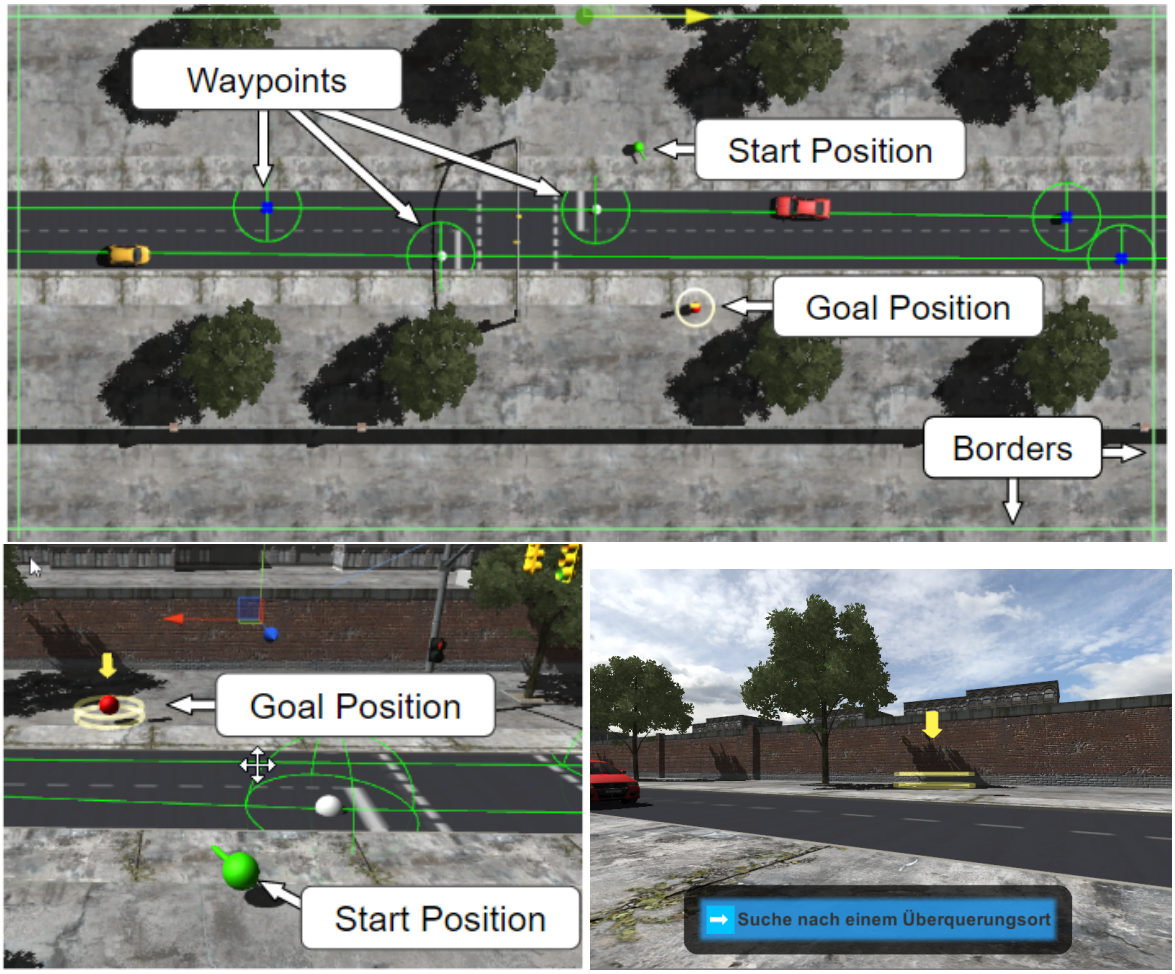


Figure 6.15: SafeChild exercise anatomy (top: isometric top view, bottom left: close view of start and goal position, bottom right: user view at runtime)

6.5 Chapter Summary

This chapter described the implementation of the SafeChild system. It shows how the BT based Domain and Student Modeling approach described in Chapters 3 and 4, as well as the interface described in Chapter 5, are realized based on the Unity game engine. In particular, the chapter presents the system's architecture, design decisions, frameworks and tools, as well as the setup of exercises within SafeChild. The current implementation of SafeChild is easily extendable through the Unity editor and thanks to the Behavior Tree Domain and Student Modeling method, additional exercises can be added with little effort. The modular setup of the system's architecture also allows the exchange of modules such as the player module. Thus, additional input and display options could be integrated without affecting the rest of the system. In terms of future work, the implementation would need to be polished before SafeChild could be distributed as a product. This includes, for instance, the addition of menus that allow non-programmers to access all administrative functionalities, long-term testing for robustness and optimization to the code to improve performance, especially on weaker computers. Further, the graphical quality of the system could

be upgraded through better models or textures. Most importantly, additional educational features (e.g. vocal instructions and feedback) should be added based on future research and studies. VR hardware and VR applications are currently (as of 2018) becoming increasingly popular, and thus, tools and frameworks to support the implementation of such software are quickly evolving as well. This allows small teams with limited resources (as in the SafeChild project) to rapidly develop feature-rich VR applications. Thus, the challenge and focus of developing high-quality VR applications shifts more towards conception and design instead of technical implementation.

Chapter 7

Evaluation

This chapter is dedicated to the evaluation of the SafeChild system with regard its potential as a self-training tool for child pedestrians. For this purpose, two studies were conducted. A pilot study to get initial feedback about the system from the target user group and a more elaborate main study in a classroom setting to evaluate skill improvements through training with the system. In the latter study, different instruction and feedback timing strategies are compared, and in particular, their effectiveness for advanced skills (see Section 4.2) are studied in detail.

7.1 Pilot Study

The pilot study was conducted as the first evaluation of the SafeChild system with the target user group of elementary school children. For this study, the main goal was to get the first impression of general user acceptance of the system and the feasibility of the VR training environment to evoke the same behavior from child pedestrians as in the real world. Thus, the two following hypotheses were formulated for this study.

PH1: The SafeChild system is appealing and easy to operate for children.

PH2: Children perform better on basic skills than advanced skills in SafeChild.

7.1.1 Experiment Design

The experiment was designed for German-speaking children from 6 to 9 years old to match the critical age group (Percer, 2009). 5-year-olds have been excluded, because (1) we believed that the controls would be too hard for them to use; (2) they are rarely exposed to traffic situations when alone since they do not go to school yet. The participants were supposed to conduct the experiment at home, under the supervision of their parents. SafeChild was provided to the volunteers as a Web-application or a download. The application itself contained detailed information and instructions for parents and children. Parents were involved for several reasons. As they hold the main responsibility for traffic safety education of their child, by supervising the study, they could detect her or his individual weaknesses and misconceptions and help their children work on them after the study. Parents could also help children with the controls and the questionnaires during the study (but were asked to avoid helping with the completion of the exercises), and provide us with valuable feedback on the SafeChild approach and implementation. Last but not least,

they could choose the most convenient time to conduct the study. The application for the pilot study was a customized version of the SafeChild system. Once started, it guided the participant through the three steps of the experiment: (1) A pre-questionnaire, (2) a set of road crossing tasks within the SafeChild city environment and (3) a post-questionnaire. The application logged experiment data to a dedicated web server. This data included questionnaire answers as well as replayable log files for each road-crossing task.

7.1.2 Pre- and Post-Questionnaires

The pre-questionnaire consisted of seven questions; it focuses on children's demographic data (age, gender) and prior experience with traffic safety education and video gaming. The post-questionnaire also contained seven questions asking about the just-completed road-crossing tasks and the overall SafeChild experience. This included children's general attitude toward the approach, their opinions on whether they could learn something in such an environment, and whether the environment was realistic. Further children were asked if they got tired during the experiment, and whether or not the controls and the tasks were too hard or too easy. The last two questions asked parents whether they helped their child and how. They were also given the opportunity to provide comments on the general approach. Using the example of the pre-questionnaire, Figure 7.1 shows the presentation of the questionnaires within the application (see Appendix B.2 and B.3 for the complete version and English translation).

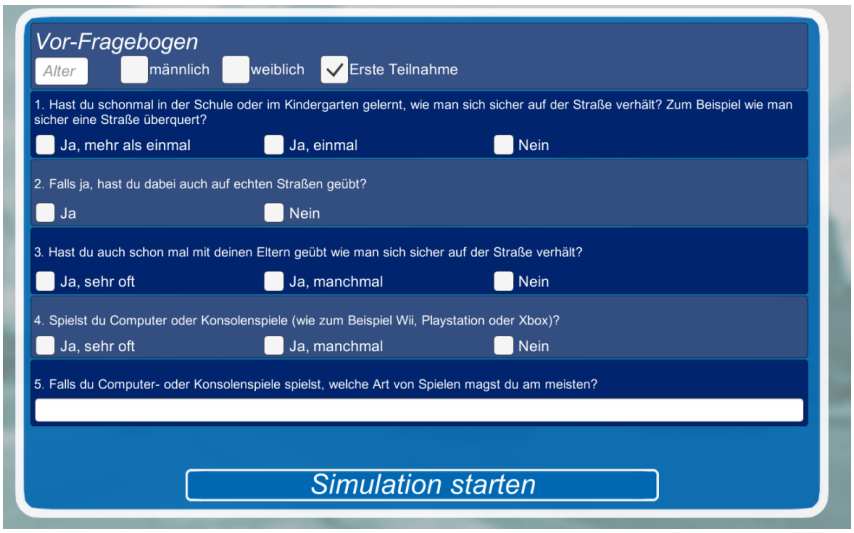


Figure 7.1: Pilot study pre questionnaire

7.1.3 Road Crossing Tasks

A total of ten exercises and a familiarization task in the beginning had to be completed. These consist of three tasks related to using traffic light, three tasks related to using zebra crossing, three tasks related to unregulated crossings, and one combination exercise. For traffic light and zebra crossing, the learner starts once directly in front of the designated crossing area; once, within a small distance, but with the traffic light / zebra crossing still in sight; and once, further away, where a turn in the virtual city environment is required to find the designated place to cross. In all cases, the final goal is directly visible, and the learner is supposed to utilize the available regulated crossing to reach it. For the unregulated crossings tasks,

the user starts once on a sidewalk next to a straight road without obstacles; once, with a parking truck as an obstacle; and once, close to a road curve obstructing the field of vision. The task is to recognize shortcomings of the crossing place if they are present and cross the road safely. In the final exercise the learner can plan the route to the goal and is given the opportunity to either cross the road once with an unregulated crossing or to cross twice using traffic light and zebra crossing; the expected behavior is to prefer the combination of traffic light and zebra crossings over the unregulated crossing (see Appendix B.1 for screenshots of all exercises).

The simulation parameters used for the road-crossing task were chosen to match the real-world conditions: car speed was around 50 km/h (the upper speed limit in German cities), and the walking speed of the child was set to 1.2 m/s (a common speed for 6-year-olds (David & Sullivan, 2005). For the "unregulated crossing" tasks, the frequency of incoming cars was high in the beginning and decreased over time to create larger gaps. Since the study was designed to be conducted by parents with their children in their own home, the standard interface configuration consisting of one monitor and keyboard was used. The controls were based on 6 keys for forward and backward movement, turning left or right and turning the head left or right. Head-turning was included as a separate action to allow the participant to look to both sides while crossing the road. The ITS functionalities were not activated for the study as our goal was to obtain baseline performance data and to validate the overall approach. Figure 7.2 shows a screenshot of the control introduction screen of the pilot study with the respective key bindings.

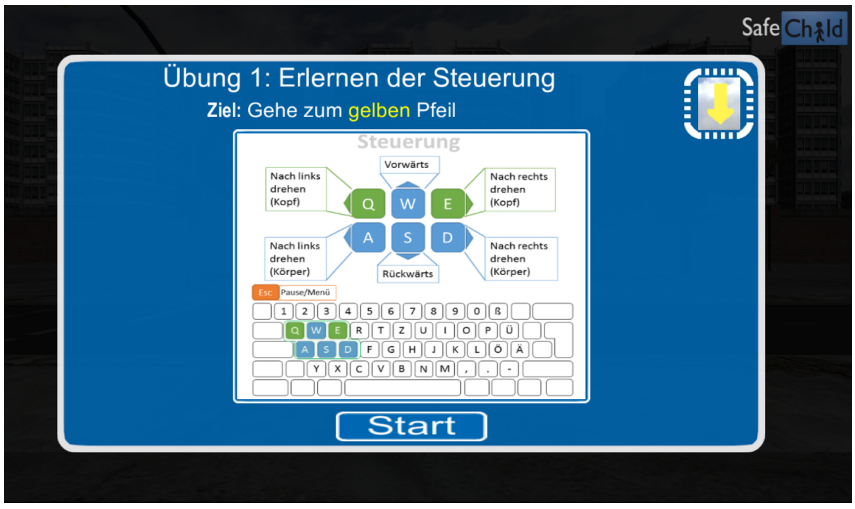


Figure 7.2: Pilot study keybindings

7.1.4 Discussion of the Results

Questionnaire Analysis

A total of 10 participants took part in the study. The results of the pre-questionnaire including demographics are presented in Table 7.1 (part A), while the results of the post-questionnaire containing feedback on the system are shown in 7.1 (part B).

The feedback about the pilot study version of the SafeChild indicates, that most children liked the overall experience, and, even more important, most of them believed that they could improve their pedestrian skills through such training. Children also mostly considered the traffic simulation realistic. However, controls via keyboard have been a problem for many participants. Thus, either a better

A: Pre-Questionnaire		B: Post Questionnaire	
total number of participants: 10			
Question	Number	Question	Number
Male	6	Liked the system in general	8
Female	4	Thought the system was educative	9
Age 6	4	Thought the graphics were real	9
Age 7	3	Thought controls were easy	4
Age 8	0	Did get tired	5
Age 9	3	Found exercises too easy	5
Traffic ed. school/kindergarten	7	Parents did help	7
Traffic ed parents	9		
Video game experience	7		

Table 7.1: Questionnaire summary

introduction of controls or other interaction methods should be considered. Half of the children considered the exercises to be too easy, although none of them was able to excel at all required skills. An explanation could be that when the children were able to complete a task , they assumed that they performed correctly even when exhibiting unsafe pedestrian behavior (see the next section for a detailed explanation). Also half of the children said that they got tired while participating. This result matches with the observation of the tasks performance logs showing that children sometimes needed more than six minutes to finish an exercise. As a comparison, adults are able to finish the entire sequence of exercises a little under five minutes. The problems with the controls also may be a factor here. Finally, seven parents reported that they helped their child a little, but limited the help to the controls and not completing the tasks. In summary, it can be concluded for the first hypothesis (PH1: The SafeChild system is appealing and easy to operate for children) that the presentation of the training environment is well accepted, but an improvement of controls is much-needed. Further, misjudgment about own progress and difficulty of exercises and need for adult support in many cases indicate that the adaptive support provided by an ITS could improve user experience and learning.

Performance on Traffic Tasks

Any traffic task implemented in SafeChild can have only two outcomes: if a child crosses the street and reaches the goal, the exercise is completed; if a child is hit by a virtual car, the exercise aborts. Out of 96 completed tasks (4 recordings were excluded because of off-task behavior), 82 constituted completed and 14 aborted attempts. However, even when the goal is reached, it does not mean that the child has exhibited correct (safe) behavior and successfully demonstrated all the pedestrian skills involved in the task. In fact, only 15 times were tasks solved without a single failed skill. It is important to understand that due to the fact the SafeChild problem-solving environment is highly dynamic with random-generated elements, children can get lucky and complete a task without applying some necessary skills. However, failing skills on a regular basis will eventually result in a car hit, which is rather similar to real-life traffic situations. In order to investigate the second research hypothesis (PH2: Children perform better on basic skills than advanced skills in SafeChild.), we have computed the average basic skill performance and average advanced skill performance for each learner. Then we could compare their means using the paired t-test.

The test shows that on average a learner performs significantly better on basic skills ($M = .93; SD = .04$) than on advanced skills ($M = .53; SD = .17$),$t(9) = 8.11$, $p << .001$, $d = 2.56$. Both the p-value and the Cohen's d indicate a very strong difference between the two variables. This, essentially, confirms that difficulties in advanced pedestrian skills reported in traffic education literature (Oron-Gilad et al., 2011) are reflected SafeChild.

Despite the very small sample size, a first analysis with regard to learning in SafeChild was conducted. Basically, the only instructional support that the system gave was the immediate knowledge-of-result feedback at the end of task completion. A failure to apply a particular skill, which leads to a car hit, triggered an error message. This allowed a child to self-reflect on what s/he had done wrong and, next time, exhibit a safer behavior. Having this in mind, we analyzed the logs of 14 incorrect task attempts and identified the skills responsible for each failure. In some cases a combination of skills was causing an incorrect attempt (e.g. "find an appropriate gap to cross" and "keep observing traffic while crossing"); and in some cases, it was hard to decide, which skill is responsible. In such situations, all candidates were treated as a probable cause. After each such case, we looked if there was another opportunity for a child to reflect on her/his error and apply the skill correctly. We found seven such occurrences: three when the causing skills were basic and four, when they were advanced. Remarkably, in all three "basic" cases, children have been able to correctly apply the failure-causing skills on the next iteration and never failed it again, including the new context of the combined traffic tasks. On the other side, all four cases of failed advanced skills were followed by more occurrences of the same erroneous behavior. This indicates that, even with minimum instructional support, children were able to improve their incorrect traffic behavior, when they needed to acquire a basic skill and even transfer it to a novel context. However, without additional support, they were not able to master the advanced skills and kept exhibiting the same dangerous behavior even after failing several traffic-tasks and receiving error flag messages.

7.1.5 Limitations of the Study

Despite the promising results obtained during the study, there were a number of limitations that needed to be considered. First, the experiments were conducted in a home environment, and the level of support provided by parents was not observable. Second, there was no sound in the simulation that could have warned children of incoming cars, and third, the interface and especially the keyboard based controls turned out to be difficult for many participants. Furthermore, the potential higher level of fatigue in later tasks could influence children's performance. Another important drawback of the experiment was a small number and high variability of subjects. It is important to underline that 6-years-olds and 9-years-olds are very different categories of traffic safety learners. Not only do they have accumulated very unequal amounts of real worlds traffic experience, but they also significantly differ in terms of the development of cognitive and perceptual-motor abilities important for mastering and executing pedestrian safety skills, controlling themselves in a VR and cognitive transfer. We were able to observe it firsthand, as the data generated by 9-years old children and 6-years old children varied substantially. For examples, according to Piaget (Ginsburg & Opper, 1988), before the age of 7, children remain egocentric in their thinking, having problems perceiving the world from the viewpoint of others. Nonetheless, the study has been useful overall and allowed us to pilot the approach, elicit the general attitude of the target population toward the system, evaluate important hypotheses and collect valuable information for designing the main study.

7.2 Main Study

Based on the results of the pilot study and the validation of general user acceptance as well as the feasibility of the VR training environment to reproduce the challenges of traffic safety as in the real world, the main study was designed to further improve the user experience of the system and to evaluate the ITS component of the system.

7.2.1 Experiment Design

In contrast to the pilot study, where the experiments were carried out at the participant's homes, the main study was conducted in more controlled settings in a classroom. Instead of parents, the observer role was taken by investigators of the study and teaching personnel. In addition, the primary input device was changed from a keyboard to a gamepad to alleviate some of the interaction difficulties observed during the pilot study. Finally, the amount and setup of exercises were adjusted, and different automatic instruction and feedback models were introduced using the ITS component of the system. The details of the experiment design are provided below.

Participants

The participants of the study were children of the second and third grades (average age = 7,5) recruited from an urban elementary school in Germany. First-graders were excluded due to the two reasons: insufficient reading ability and lack of any formal theoretical traffic safety training. Children needed to have some reading skills since instructions and feedback were given in the text form). They were also required at least minimal understanding of traffic safety rules, as the SafeChild system focuses on training practical pedestrian skills. In this school, children learn these rules through a traditional theoretical traffic safety program during the first grade. Parents and teachers were provided with detailed information about SafeChild prior to the experiment. Parental consent forms were collected for all children involved in the study. Overall, 58 subjects participated in the experiment (males = 31, females = 27; 2nd-graders = 29, 3rd-graders = 29) balanced across six classes (9-10 pupils per class).

Instruments

Pen and paper questionnaires were used to gather demographic information, feedback about the system and to assess theoretical knowledge pre- and post-treatment. The requested demographics include age, gender, whether the participants cross roads alone on the way to school, whether they feel safe while doing so and whether they play computer games. Regarding feedback, the questions include whether the overall experience was enjoyable, whether it was educational, whether the graphics were realistic, whether the controls were easy and whether working with the system was tiring.

The theoretical questions focused only on advanced skills (as explained in Chapter 2). Three questions were asked pre-treatment and, in a slightly altered version, post-treatment. They involve finding a safe spot to cross, estimating danger for different crossing types and recognizing traffic that could pose a threat in their crossing task. The specific questions in German and English can be found in the Appendix C.3 and C.4.

System Variations Three versions of the SafeChild system with different instructional models have been developed for this experiment. They are described in detail in Section 5.4, but a brief summary is provided here as well as a reminder. The first model – "N" (no intervention) – shows learners no instructions or feedback at all. It has served as the baseline and helped us to examine whether children could improve through unassisted practice with SafeChild.

The second model – "I" (immediate) – has provided learners with instructions about what to do next and with the direct feedback in the form of popping-up color-coded and icon-annotated messages. The left part of Figure 8.6 presents an example. Before crossing a road at the traffic light area, as the learner has stopped at the curb, an instructional message with a blue background is displayed: "Wait for green". Once the learner has successfully waited until the green light turns on, the background of the message switches to green, a happy smiley icon appears next to it, the message blinks and dissolves. If the learner starts crossing without waiting, the blue background of the instruction message switches to red and a sad smiley icon annotates it. If SafeChild needs to present to the learner another instruction, it is displayed above the old message.

The third model – "D" (Delayed) – has presented the same set of instructions but have not presented any feedback until the completion of an exercise. Once the goal has been reached, or an exercise has been interrupted due to contact with a moving car, a complete list of the skills involved is displayed to the learner; same colors and icons are used to indicate the correctness of a skill application (right part of Figure 8.6). The delay of feedback has been introduced to reduce the additional cognitive effort associated with processing feedback and instructions while performing the exercise.

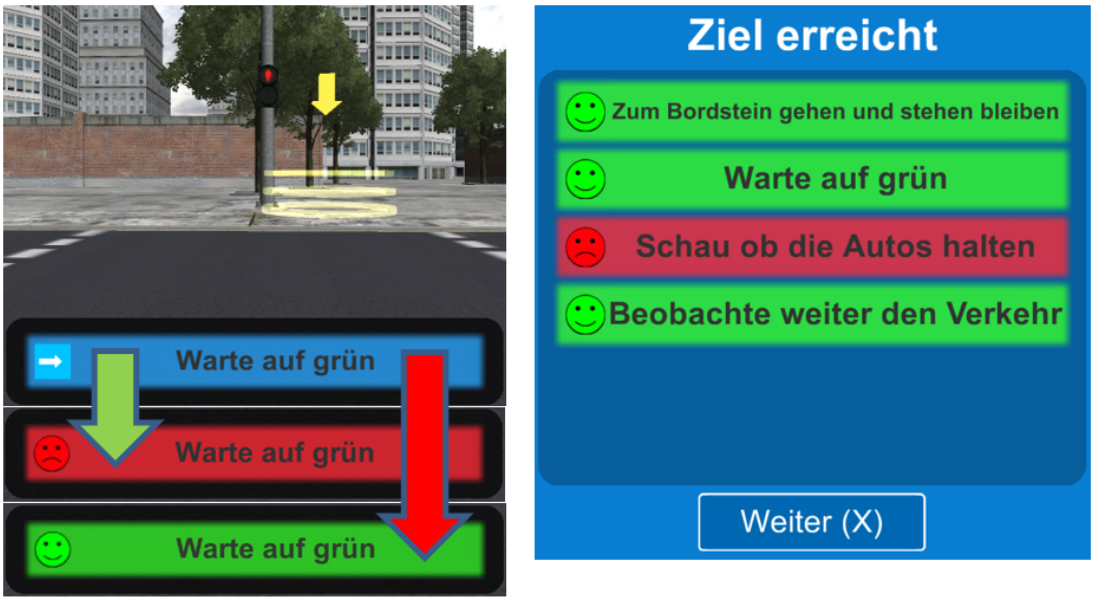


Figure 7.3: Two feedback strategies. Left: immediate feedback. The blue text box shows instructional text ("Wait for green"). The box changes immediately to green if the skill is applied correctly, or to red otherwise. Right: delayed feedback. Feedback is shown after the completion/interruption of an exercise. The green messages correspond to correctly applied skills ("Stop at a curb", "Wait for green", "Observe cars while crossing") and the red message indicates an error ("Wait for cars to stop").

Hardware For the experiment, The SafeChild system was installed and executed on ten 15" notebooks, each equipped with a gamepad (see Figure 7.4). The key mapping for the gamepad is shown in Figure 7.5.

This corresponds to the Desktop VR setup described in Section 5.3. The notebooks were installed in a classroom with two table rows. Sight protections were put between the notebooks, and the seats were arranged back-to-back in order to prevent students from looking at the screens of others.

Figure 7.4: Experiment interface: notebook and gamepad

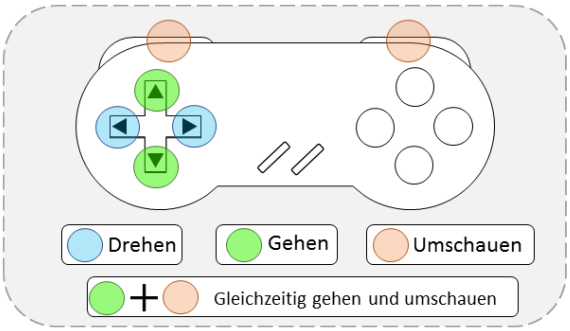


Figure 7.5: Gamepad keybindings (translation: **Drehen** - turn, **Gehen** - walk, **Umschauen** - look around (turn head), **Gleichzeitig gehen und umschauen** - walk and look around at the same time)

Procedure The duration of each experiment session was 35 minutes and organized as shown in Figure7.6 (a regular class takes 45 minutes in Germany, so 10 minutes were used as a buffer. For example for children to get seated and additional questions). A total of six sessions were completed, the participants of each session were from the same class and balanced across different instructional models.

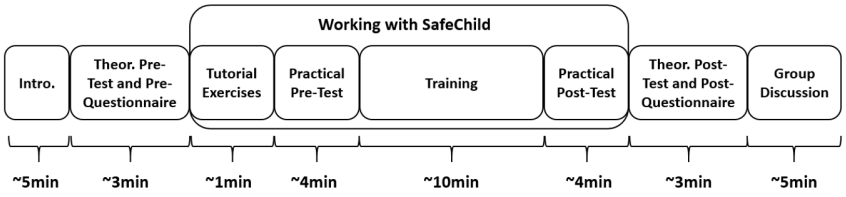


Figure 7.6: Experiment timeline

At the beginning of each session, the experimenter introduced the basic idea of the SafeChild project and the study to the participating children. Then, every participant was assigned to a computer. Sight

barriers were put to prevent children from watching on each other's screens. After the children have taken seat, they were asked to fill-in the pre-questionnaire and the theoretical pre-test. Before working with the SafeChild system, the experimenter demonstrated the controls. In SafeChild, a total of 14 exercices had to be completed. These consisted of one familiarization exercise, three practical pre-test exercises (1 traffic light, 1 zebra crossing, 1 unregulated crossing), seven training exercise (2 traffic light, 2 zebra crossing, 3 unregulated crossing) and 3 post-test exercises (1 traffic light, 1 zebra crossing, 1 unregulated crossing). Screenshots for all exercises can be found in Appendix C.2. The pre- and post-test exercises were nearly identical with slight changes to the environment such as the placement of buildings, trees or cars (see Appendix C.1). For traffic light and zebra crossing, the test started right in front of the regulated crossing, and the goal was directly on the opposite side of the road. The task was to get to the other side safely with the help of the regulated crossing. For the unregulated crossing, the start was between a parking car and a curve with the goal on the other side. The task is to get away from both, curve and parking car, to gain enough vision to both sides and then cross safely. For the zebra crossing and traffic light training exercises, the first one started, as in the pre- and post-test, directly at the designated place to cross, while in the second exercise starting position and goal position was away from the designated crossing place. Thus, it was necessary to find and go to the crossing place first before crossing. The walking speed was set to 1.55 meters per second (m/s) which is a typical value for brisk walking in this age group (David & Sullivan, 2005). This is an increase as compared to the 1.2 m/s in the pilot study since 6-year-olds were excluded from the study. On the other hand, car speed was slightly reduced to around 30 km/h. This reflects the typical speed limit for roads around schools that children are likely to cross. In fact, the school from which the participants were recruited is located at such a road without regulated crossings nearby. As in the pilot study, the density of cars for unregulated crossings would decrease over time until there are no cars at all. Interestingly, interviews with teachers revealed that there is no consensus about how children in the target age group should train the crossing of unregulated roads. One opinion is that, due to underdeveloped perceptual-motor skills, children should only cross roads when they are completely empty. Another opinion however is that children should be encouraged to find and choose appropriate gaps in traffic. By decreasing density over time, SafeChild gave children the opportunity to exhibit both behaviors. During the practical part of the study, the skills as described in Section 4.1.2 needed to be exhibited by the user. After completing the exercises within SafeChild, the participants filled out the post-questionnaire and were gathered for a concluding group discussion. The participants were given a chance to ask questions and were instructed again on how to behave safely on roads, to practice often with their parents and that the virtual world depicted in SafeChild is different from the real world.

7.2.2 Research Hypothesis

In the pilot study, it was established that the SafeChild system is appealing to children and that challenges of traffic safety to them is well reflected in the virtual environment. This approves the general usability of the SafeChild as a tool for traffic safety education.

However, the study also has shown that with only basic and binary feedback (hit by car or reached the goal), the educational effect of the system, especially with regard to advanced skills, was rather limited. The main focus of the main study therefore is to investigate whether the addition of an ITS component could improve substantially on this aspect. The need for an ITS was derived from the large solution space of exercises in SafeChild, which is a result of the highly dynamic nature of traffic safety and the ill-defined

pedestrian tasks (as discussed in Chapter 3). In order to guide a learner in a step-wise fashion through these exercises, the system must interpret and understand user actions to give appropriate instructions and feedback based on the specific training situation.

Since the combination of ITS, VR and pedestrian safety is a novel approach, the hypotheses are formulated in a way to examine incrementally more subtle effects starting from the overall performance, to the performance on advanced skills, to the impact of the delay feedback strategy on the performance on advanced skills. Feedback and especially timing has been subject to discussion within education research for a long time (Hattie & Timperley, 2007), and there is no clear winner between delayed and immediate feedback. Although feedback after an exercise was proposed for real-time learning environments (H Chad Lane & Johnson, 2008), this approach needs to be confirmed and re-evaluated to the change of domain and target user group in this research. One specific concern against delayed feedback is the lack of motivation for children to process information after an exercise before engaging into the next interactive experience.

MH0 Children are able to improve in theoretical knowledge about the dangers of traffic through training with SafeChild (especially non-obvious ones)

MH1 Instructions and feedback generated by SafeChild have a significant positive effect on improving learners' practical performance on traffic safety skills. (By reducing missing or incorrectly performed behavior)

MH2 Instructions and feedback generated by SafeChild have a significant positive effect on improving learners' practical performance on advanced skills.

MH3 Instructions with delayed feedback generated by SafeChild have a significant positive effect on improving learners' practical performance on advanced skills.

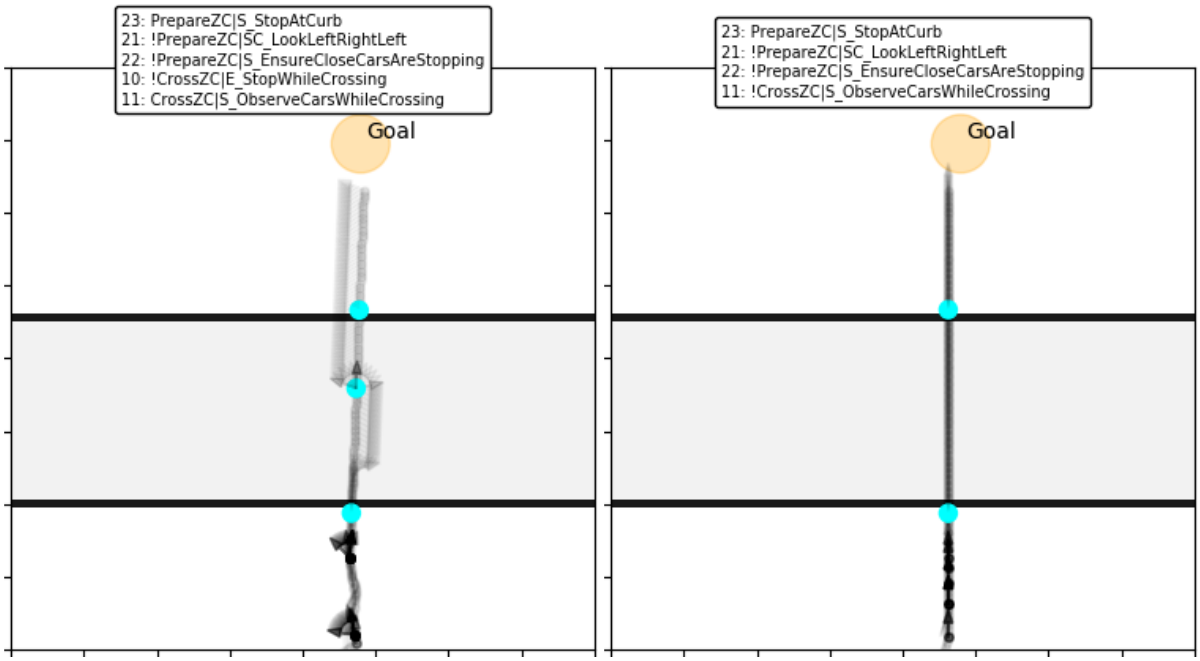
7.2.3 Experiment Results

The data gathered during the experiment includes the results of the questionnaires and logged information for each participant and exercise. The logs contain raw movement information about learners' locomotion and head-turning as well as skill performance generated by the student modeling process. The technical correctness of the Student Modeling method needed to be confirmed. For this purpose, the logged data was used to generate charts that depict the course of each exercise. They show the position (as dots) and look direction (as arrows) of the user together with skill detection events as circles. A chronological list of correct and failed skills is listed in a separate box. Examples of these charts are shown in Figure 7.7.

These charts enabled efficient manual validation of the Student Modeling Performance, as anomalies such as false positives would visually stand out. For instance, if there are no arrows indicating left and right looks but the skill of looking left right left would be present in the chart. The result of manual inspection indicated that the Behavior Tree based Student Modeling Framework has performed as designed and correctly from a technical point of view in all instances.

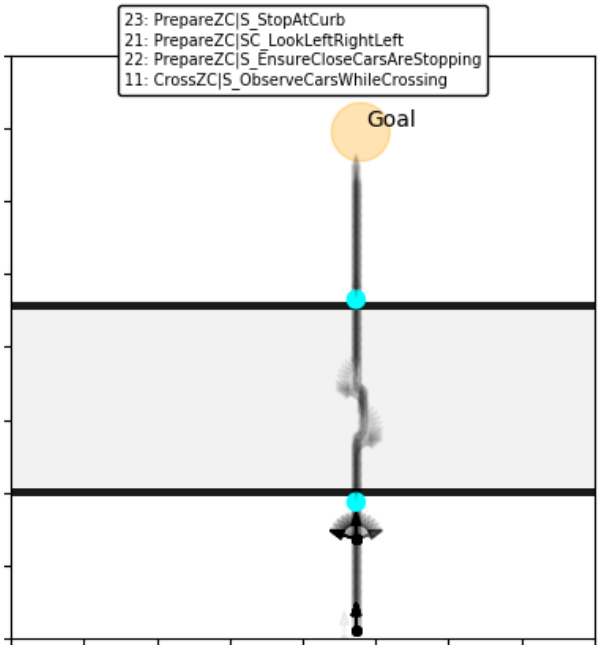
Demographics and User Experience

The results of the questionnaire are summarized in Table 7.2 below. It shows for the pre-questionnaire that the majority of participants do cross roads on their way to school and that most of those feel safe doing so. Since the other results indicate that many participants actually struggle with behaving safely in



(a) Incorrect execution

(b) Incorrect execution



(c) Correct execution

Figure 7.7: Task Performance Charts. All charts depict performances of zebra crossings. Sub-figure (a) shows an erroneous execution. As the arrows indicate, the participant had looked only left before crossing and also stopped during crossing. S/he has however stopped before crossing and also observed cars while crossing as indicated in the skill box and the darker parts of the plots, which indicate a longer stay of the user at that place. Sub-figure (b) shows another incorrect execution. The participant did walk straight from the start to the goal without stopping or looking anywhere besides towards the goal. Finally, sub-figure (c) shows a correct execution. The participant had stopped before crossing, looked left-right-left and ensured that incoming cars are stopping. During crossing s/he has looked both sides, thus observed cars while crossing.

	Question	Yes	No	% of Yes
Pre Quest.	Do you cross roads alone on your way to school?	41	17	71.4
	Do you feel safe while doing so?	35	6	85.3
	Do you play video games?	45	13	77.6
Post Quest.	Did you like the training?	55	1	98.2
	Do you think you learned something?	54	2	96.4
	Where the controls easy to use?	40	15	72.7
	Did the cars and buildings look realistic?	40	16	71.4
	Did you get tired during training?	4	52	7.1

Table 7.2: Main study questionnaire results

all situations, it is apparent that additional practical training is indeed important and necessary. A high percentage of the participants are familiar with digital gaming and thus the style of interaction provided by SafeChild. For feedback related questions of the post-questionnaire, the results show that almost every participant of the study enjoyed the training provided by SafeChild and only a few children perceived the training as too tiring. The high ratio of positive answers regarding perceived realism and ease of controls show that the interface is mostly appropriate for the children in this age group. Compared to the pilot study, both the strain of exercises and difficulties with controls were reduced. However, it is also apparent that there is still room for improvement in these areas. With the recent rise of VR on the consumer market, devices that allow more natural interaction and are appropriate for children could soon be available. Initial experiments have already been conducted based on the SafeChild system as described in Chapter 5, which show how a fully immersive head-mounted display in combination with a gesture tracking device can use natural head rotation and arm swing for looking around and walking (Section 5.3 presents more details and why this setup is currently not yet feasible for child pedestrian safety training in regular schools). Last but not least, all participants answered that they learned something. Although this is a positive result in general, the subsequent analysis shows that the observable improvements were not as strong as this answer might suggest.

Theoretical Questions

A comparison of pre- and post-treatment scores showed no significant difference across all questions for all instructional models (Table 7.3). Therefore, the hypotheses MH0 has to be rejected. There are several possible reasons for this result. First of all, SafeChild mainly trains practical skills, not theoretical understanding. The gain in theoretical knowledge would have been a desirable side effect. The questions were also deliberately designed to differ from SafeChild exercises, hence, increasing the difficulty even more, especially because all questions target advanced skills that children struggle with in general. Second, the experiment was rather short due to hard time constraints and thus potentially too short to have an impact on theoretical knowledge gain. Finally, the practical part of the experiment was more attractive to the participants. As a consequence, children could have been distracted during the pre-test and might have been exhausted after the training. The questions' design goal was to check for theoretical knowledge gain in a very short time but in retrospect might have been too optimistic in terms of the educational effect of SafeChild and the cognitive abilities of young children.

Exercise	% Correct Pre-Test	% Correct Post-Test
1 (Recognize dangerous crossing places)	24	19
2 (Understand to stay vigilant at all types of crossings)	12	9
3 (Recognize incoming cars)	47	43

Table 7.3: Theoretical test performance

	Pre-Test	Post-Test	p-value	t-Statistics	df	Cohen's d
	M(SD)	M(SD)				
N	9.50(1.79)	10.40(1.57)	**.007**	2.71	19	0.61
D	9.89(1.59)	11.00(1.76)	**.005**	2.84	18	0.65
I	10.61(1.97)	11.28(1.71)	.099	1.34	17	0.32
Total	9.98(1.82)	10.88(1.69)	<.001	3.86	56	0.51

Table 7.4: Comparison of pre- and post-test scores across instructional models

Overall Practical Performance

To analyze hypothesis MH1, we relied on the diagnostic information produced by the Student Model (correctly/incorrectly applied skills). Two records were excluded. One participant took the experiment in an extra (shortened) session during a break, and another participant had tremendous problems with the controls and thus was not able to finish the exercises. The pre- and post-test scores were computed based on the number of skills a learner applied correctly. A total of 17 skills[1] were registered in each test; one point was awarded for each skill. A paired t-test on the entire group of learners indicates an overall significant improvement from pre- to post-test (see Table 7.4). However, when we look at individual groups, the results are less conclusive. The post-test scores of the groups receiving delayed (D) and immediate (I) feedback are above the control group (N); however, only for Groups "D" and "N" the increase in scores from pre- to post-test was significant. Group "I" itself did not improve significantly. There is also no significant difference in knowledge gain (either absolute or normalized) between the groups. Hence, such a binary skill performance comparison does not yield sufficient evidence to uniformly support MH1. When we treat performance on each skill as either zero or one, we lose a lot of valuable information. A more detailed analysis is required focusing on the substantial pedestrian activity that helps to assess partial progress on individual skills with better accuracy.

Practical Performance on Advanced Skills

As mentioned in Chapter 4, four advanced skills were identified amongst the skills considered by SafeChild: "Make sure that cars have stopped", "Observe cars while crossing", "Find a good unregulated area to cross" and "Find a good time gap to cross". These skills were analyzed in details and more fine-grained features characterizing learners' performance of these skills were selected and computed. Significant effects on learning with SafeChild have been be observed on all advanced skills, except for the skill

[1]This is fewer than the full list of skills presented in Chapter 4 as several skills were omitted from the analysis. "Find a regulated area to cross" was applied correctly by all children. "Find a good un-regulated area to cross" was never applied correctly. Since this skill is a pre-requisite for the subsequent skills such as "finding a good time gap to cross", the system was never set to a state to track those skills. As described later, the raw log information was pro-cessed to analyze partial success in those skills.

Scenario	Condition	Pre-Test M (SD)	Post-Test M (SD)	p-value	t-Statistics	df	Cohen's d
Traffic light	N	0.55 (0.69)	0.45 (0.76)	.71	-0.57	19	-0.13
	D	0.47 (0.84)	0.74 (1.37)	.25	0.69	18	0.16
	I	0.44 (0.78)	0.33 (1.19)	.69	-0.49	17	-0.12
Zebra crossing	N	0.20 (0.70)	0.25 (0.55)	.41	0.24	19	0.05
	D	0.11 (0.32)	0.74 (1.10)	**.009**	2.59	18	0.59
	I	0.56 (1.15)	0.78 (2.13)	.31	0.51	17	0.12
Unregulated crossing	N	0.55 (0.83)	0.60 (0.94)	.42	0.20	19	0.04
	D	0.21 (0.54)	0.74 (0.99)	**.02**	2.14	18	0.49
	I	0.33 (0.84)	0.94 (1.55)	.06	1.64	17	0.39

Table 7.5: Comparison of pre- and post-test scores for the measure "number of looks left and right while crossing" (skill "observe cars while crossing") for three different scenarios (traffic light, "zebra" crossing, unregulated crossing) and across feedback models

"Make sure that cars have stopped". A potential explanation could be that children incorrectly perceive regulated crossings (the context of applying this skill) as completely safe with no need to check whether the incoming cars are stopping indeed.

Observe cars while crossing The correct application of the skill "Observe cars while crossing" requires a child to look left and right while s/he walks across the road. Some children did not look right and left at all; others did look, but only in one direction, which was interpreted by the student model as an incorrect application of a skill. We have decided to compute the number of right and left looks during road crossings to give learners partial credit for trying to apply this skill. The paired t-test on this data indicates that children in the "D" group improved significantly on "observing cars while crossing" both, zebra crossings (from ($M = 0.11$, $SD = 0.32$) to ($M = 0.74$, $SD = 1.10$), $t(18) = 2.59$, $p = .009$) and unregulated crossings (from ($M = 0.21$, $SD = 0.54$) to ($M = 0.74$, $SD = 1.00$), $t(18) = 2.14$, $p = .02$). The group improved in the traffic light scenario as well, but not significantly (from ($M = 0.47$, $SD = 0.69$) to ($M = 0.74$, $SD = 0.86$), $t(18) = 0.69$, $p = .25$). From observing the logs, it looked like children mistakenly assumed crossings regulated by traffic light to be completely safe and paid no attention to cars as long as the pedestrian light is green. The other groups ("N" and "I") did not improve significantly in any crossing scenario. Table 7.5 summarized the results of pairwise comparisons of pre- and the post-test values for all crossing scenarios and feedback conditions.

Find a good unregulated area to cross To successfully apply the skill "Find a good unregulated area to cross", a pedestrian needs to walk to a place providing sufficient vision in both directions. In the test scenario, a curve and a parking car were present as sight obstacles. Both were placed on the left-hand side of the starting position. Therefore, the task of a learner was to walk a short distance to the right until s/he can see far enough left to make an informed decision about the possibility of crossing. The "safe distance" was computed based on a car speed of 40 km/h and a safety margin of 1 second (i.e., after successful crossing, it would take at least a second for the car to reach the crossing area). Based on these parameters, no participants reached a good unregulated place to cross. This could be both a deficiency in children's distance perception or a problem of fine-tuning our computational parameters. Although no participant

Condition	Pre-Test M(SD)	Post-Test M(SD)	p-value	t-Statistics	df	Cohen's d
N	0.62 (1.20)	2.67 (2.97)	**.004**	2.99	19	0.67
D	0.18 (0.40)	5.50 (5.73)	**<.001**	3.97	18	0.91
I	0.11 (0.39)	4.63 (5.11)	**<.001**	3.69	17	0.87

Table 7.6: Comparison of pre- and post-test scores for the measure "distance walked towards safe place" (skill "find a good unregulated area to cross") across feedback models

Condition	Pre-Test M(SD)	Post-Test M(SD)	p-value	t-Statistics	df	Cohen's d
N	0.25 (0.72)	0.30 (0.66)	.42	0.21	19	0.05
D	0.18 (040)	0.74 (1.10)	**.009**	2.58	18	0.59
I	0.61 (1.14)	0.83 (1.69)	.28	0.61	17	0.14

Table 7.7: Comparison of pre- and post-test scores for the measure "number of looks left and right before crossing" (skill "find a good time gap to cross") across feedback models

walked far enough, a significant increase in the effort to avoid the obstacles (measured in meters that learners walked toward the good crossing area) has been observed for all groups (see Table 7.6).

It is also worth mentioning that learners receiving support from SafeChild improved significantly more. When comparing the differences in the improvement (i.e. gains in distance walked toward the safe crossing area between the pre- and post-test) across the groups, we have seen that the "N" group (M= 2.29, SD= 2.73) is significantly outperformed by both, the "I" group ($M = 4.60$, $SD = 5.12$), $t(36) = 1.76$, $p = .043$, and the "D" group ($M = 5.38$, $SD = 5.79$), $t(37) = 2.14$, $p = .019$. Between the "I" and "D" groups, there was no significant difference $t(35) = 0.43$, $p = .34$.

Find a good time gap to cross The correct behavior of the skill "find a good time gap to cross" consists of looking to both directions while having sufficient vision and starting to cross when there are no cars around or cars are sufficiently far away. The last right and left looks need to be within a few seconds from the moment of starting to cross. This is one of the most challenging but also one of the most important skills; there are training systems that focus on this skill alone (J. A. Thomson et al., 2005). Unfortunately, since no participants successfully applied the previous skill "Find a good unregulated area to cross", they had no chance to correctly implement this skill due to limited vision. Yet, we could estimate the partial improvement children have made on this skill. For this, we counted the number of left and right looks before starting to cross as a measure of attention. A paired t-tests on all three groups indicates that the "D" group had a significant increase from pre- ($M = 0.18$, $SD = 0.40$) to post-test ($M = 0.74$, $SD = 1.10$), $t(18) = 2.58$, $p = .009$, while the other two groups improved, but not significantly ("I": from ($M = 0.61$, $SD = 1.14$) to ($M = 0.83$, $SD = 1.69$), $t(17) = 0.61$, $p = .28$; "N": from ($M = 0.25$, $SD = 0.72$) to ($M = 0.30$, $SD = 0.66$), $t(19) = 0.21$, $p = .42$). See Table 7.7 for more data.

Additional Findings During Training

In addition to the results of the pre- and post-tests, also the performance during training was analyzed. The most notable finding by using the one-tailed Fisher's exact test is that children who received instructions

performed significantly better in finding a better place to cross. For the second traffic light exercise the test yields $p < 0.05$ and for the second zebra crossing exercise $p < 0.005$. As described in Section 7.2.1, children started away from the regulated crossing and were required to find and walk to the regulated crossing. For the second unregulated exercise, with the parking car that needs to be walked away from, the test yielded as well a significant result with $p < 0.02$. However, instructions did not help with the third unregulated exercise where children needed to walk away from a road curvature. This could be due to the same reasons as in the third post practical post-test exercise as discussed above. These results shows that instructions indeed affect children's behavior during training.

Practical Performance Summary

The immediate feedback helped learners only with one of the advanced skills ("Find a good unregulated area to cross"); however, the "N" group could demonstrate significant improvement on this skill as well. Hence, we do not have enough evidence to confirm hypothesis MH2 – not any adaptive support can help learners improve their performance on advanced skills.

The group receiving delayed feedback has not significantly improved on one out of four advanced skills ("Make sure that cars have stopped"). We believe that the reason for it is the persistent misconception ("Cars always stop at regulated areas") that children acquire through observation of others, insufficient details during theoretical traffic safety education and/or the fact that it is hard for them to stay alert with regard to the behavior of others. The current design of SafeChild did not target this misconception specifically. This group also has not improved on applying the skill "Observe cars while crossing" in the most "regulated" situation – when crossing at the traffic light zones. The potential explanation here is very much similar – children believe that they are absolutely save, while crossing at a traffic light. Such behavior follows the simplified instruction they tend to receive from their parents, that can be summarized as "green means go". For the rest of the skills, though, the "D" group was the only one to improve significantly. Hence, we can accept the MH3 hypothesis – delayed feedback does help learners to improve their performance on advanced skills.

Interestingly, the only situation where "D" and "I" performed similarly, was the "Find a good unregulated area to cross" skill, which is applied in less dynamic settings as it neither involves decision making nor the combination of multiple actions under time pressure. The two skills, where the "D" group has improved and "I" did not, require children to do exactly that. "Observe cars while crossing" is applied as a part of a sequence of skills in a dynamic manner. "Find a good time gap to cross" expects children to make a decision while constantly monitoring the current state of traffic.

7.2.4 Limitations

Despite the overall promising outcomes, we have observed some limitations of the study and identified several aspects of the system and the experiment that require improvement. Most notably, the text-based presentation of information was a problem for many participants since reading abilities are not reliable for this age group. Second, the sample size and the duration of the study were rather small to draw conclusive results. Longer or multiple sessions could yield better, more sustained outcomes. Further, although children showed significant progress on most skills, the exhibited behavior was often far from safe. SafeChild does allow children to reach the goal of an exercise even if the crossing has not been

100% safe. The incorrect applications of skills are registered, and the feedback is given; yet, it is up to a learner to act upon it.

7.2.5 Chapter Summary

The two studies presented in this chapter showed that the SafeChild approach to child pedestrian safety training is generally very well accepted by learners. Almost all children across genders and age groups that participated in the training reported that they enjoyed working with the system. This matches the observation during the classroom experiment where children showed strong excitement and asked to repeat the practical exercises. Further, the majority of children that worked with the system perceived the VR training environment as realistic, found the gamepad based control easy to use and quickly understood the concept of the exercises. The fact that the parents allowed their children to participate in the study further indicates that they are content with this type of training as well.

The necessity of practical child pedestrian safety training was also apparent in the results. The indicators include poor performance of many children, especially on advanced skills. Also, underestimation of the difficulty for crossing roads, as seen in the pilot study, is alarming. This combined with the fact that a large number of children do indeed walk to school alone shows the need to improve practical child pedestrian safety skills.

With regard to the ITS component of the system and the novel Behavior Tree based Student Modeling framework, the studies have shown promising results as well. One positive finding is the ability of the Student Modeling framework to effectively and efficiently detect errors and correct skill applications. This was confirmed by manually inspecting charts generated from logged Student Modeling data. In principle, this shows the feasibility of this method for intelligent problem-solving support in the form of timely and situational instructions and feedback. The form of such intervention however needs to be improved to be effective. The textual form utilized in the main classroom study has shown to be sometimes difficult to understand and process quickly for young children. Options for future investigations include adding audio or symbolic elements to the messages or pausing the simulation at certain times. Finding a definite answer to this complex and interdisciplinary question is outside the scope of this thesis, but a critical prerequisite before autonomous VR training can be applied to this domain.

Chapter 8

Conclusion

The goal of this thesis was to design, implement and evaluate an Intelligent Tutoring System (ITS) with a Virtual Reality (VR) interface for pedestrian safety education and to investigate the potential and challenges of this combination. For this purpose, the prototype of such a training system using a novel Domain and Student Modeling method was created. In this final chapter, the most important results of the previous chapters will be summarized before revisiting the research questions formulated in the first chapter. A discussion about limitations as well as directions for potential future work will then wrap this thesis up.

8.1 Summary

In the first chapter of this thesis, the idea and motivation to combine VR with ITS for the purpose of improving child pedestrian safety training by enabling children to train safely, realistically and independently was introduced. The rest of the dissertation was then dedicated to realizing this idea by designing, implementing and finally evaluating the VR-ITS SafeChild. For this purpose, the state of the art in related research fields, with a focus on studies using VR for pedestrian safety education and VR with ITSs, was established first. Based on the outcome, the need for novel Domain and Student Modeling methods for the pedestrian safety domain with its highly dynamic nature and ill-defined tasks was identified. This problem was then tackled by designing a novel Domain and Student Modeling framework based on the concept of Behavior Trees (BT). This framework was then used to encode pedestrian safety skills and to determine correct and incorrect actions of the user during exercises. For the interface part of SafeChild, a virtual urban environment with traffic was created and several experiments with different VR setups, ranging from desktop to fully immersive VR, were conducted. Due to practical reasons, the desktop VR setup was deemed the most feasible for today's usage in ordinary schools. The results of the novel Domain and Student Modeling Framework and the VR interface were then merged to implement the actual system, which was used for two studies with the target user group. While the first experiment was designed to evaluate the general feasibility and user acceptance of SafeChild, the goal of the second study was to investigate the educational potential of the system with different feedback timing strategies. In total, the thesis contributes a VR-ITS with a novel Domain and Student Modeling framework for the domain of pedestrian safety training (which was untouched by ITS research) with an initial evaluation of its effectiveness. Thus, the gap towards highly available, effective and self-taught child pedestrian education was significantly narrowed.

8.2 Research Questions Revisited

In this section, the research questions stated in Chapter 1 will be revisited to discuss how this dissertation has helped to answer or work towards a solution for them. Some research questions were only partly answered by this thesis, and further investigation in future work will be required to fully resolve them.

RQ1: What are the skills in the domain of pedestrian safety education that are best to be trained in a VR-ITS? Mainly through literature research and studying previous work it was identified that there is a large number of factors and skills which influence pedestrian safety. Out of these skills, there are a few that are especially challenging to train for children due to their incomplete cognitive development. Since these (advanced) skills rely on practice in real-world environments, they are in particular suited to be trained in a VR environment. In total, the thesis has considered twelve skills under different crossing conditions of which four are advanced skills. See Section 4.1.2 for the complete list.

RQ2: How should the skills and their correct implementation be encoded in a VR-ITS and how should the system evaluate user actions to detect errors? For this purpose, the existing methods of Domain and Student Modeling within VR-ITS have been examined with the conclusion that they are not ideal for a domain such as pedestrian safety education, which is highly dynamic and contains ill-defined tasks. To resolve this problem, this thesis introduced a novel Domain and Student Modeling framework based on the concept of BTs. BTs are common in Artificial Intelligence (AI) programming in computer games and offer a number of standard constructs to create concise graphical representations of complex behavior. By using these components to realize the Domain Modeling, the typical traversal and update mechanisms of BTs can be used for Student Modeling. The concept is described in detail in Chapter 3 while the pedestrian safety education specific implementation is presented in Chapter 4.

RQ3: What VR interface is suited for widespread pedestrian safety training? For the SafeChild system developed In this dissertation, three different forms of VR interfaces were designed and studied for their feasibility with regard to child pedestrian safety education. This includes a fully immersive setup with a Head-Mounted-Display (HMD) and other wearable trackers, a semi-immersive setup with multiple stereoscopic monitors and a motion tracking camera and last but not least a traditional desktop setup using a single monitor and keyboard or gamepad input. Due to practical reasons, including IT equipment and funding situation in typical German elementary schools, the low cost, and ease of deployment of the desktop VR setup was selected as the current best choice. The software architecture of SafeChild is designed in a way that additional interaction models can be quickly integrated. The majority of interface related content can be found in Chapter 5 of this thesis.

RQ4: How is the general user acceptance of the main target group of young children for a VR-ITS to teach them pedestrian safety? The studies conducted in this thesis, pilot and main, both included questionnaires to ask about feedback with regard to the system (presented in Chapter 7). The vast majority of users have a very positive stance towards the SafeChild system. They liked the overall approach of having a game-like training application, attributed educational potential to the system and thought of the virtual environment as realistic. This can be further confirmed by witnessing the children during the experiment since joyful comments and strong demand for repeating the training could be observed. With

regard to the controls, the pilot study delivered mixed results which lead to the change from keyboard to gamepad controls for the main study. In fact, the gamepad controls were better received by the participating children. With these results, it can be said that this form of training is highly motivating and holds great potential for effective self-training for children.

RQ5: Can the resulting system developed in this thesis actually help children improve in pedestrian safety skills and do different feedback strategies cause a difference? For the last research questions, the main study (described in Chapter 7) delivered promising results but also made clear that there is great room for improvement. Also, the long-term effects, as well as the transfer of knowledge and skills to real-world behavior, are yet to be determined. However, the study showed that children were able to improve within the system in a statistically significant manner. Also, a more in-depth analysis of performance in advanced skills revealed that system generated instructions and feedback do make a difference and are useful to improve performance. Both delayed and immediate feedback prompted participants of the study to walk further away from an obstacle while delayed feedback also helped with the skills of "Observe cars while crossing" and "Find a good time gap to cross" as indicated by a greater number of head-turns during and right before crossing. In conclusion, SafeChild has the potential to help children to improve, and different feedback strategies seem to make a difference.

8.3 Limitations

The ambitious goal of changing nowadays child pedestrian safety education by introducing a VR-ITS for a wide range of schools could not be achieved by a single PhD thesis. Despite the technical and conceptual contributions made through this dissertation, there are a number of limiting factors that prevented a further advancement on this topic. One of the biggest limiting factors is the lack of dedicated and pedestrian safety specialized psychological and pedagogical research to accompany the more technology-oriented work presented here. Therefore, the exercise design, as well as the pedagogical instructions and feedback, are more of a proof of concept instead of scientifically mature solutions. This is further amplified by the fact that pedestrian safety is a new domain for ITS research and thus there is no closely related previous work to rely on. This leads to the second limiting factor: lack of close collaboration with practitioners of child pedestrian education. Although individual voluntaries, as well as an elementary school, agreed to participate in the pilot and main study, there was no formal cooperation or support from traffic safety educators. Due to the complexity of technology, it was therefore not possible to get specific design suggestions during short interview sessions. However, the feedback that was received from traffic safety experts were positive towards the general approach. Last but not least, time and resources were also a limiting factor. Major parts of this thesis was conducted as part of the SafeChild project which is a government-funded project as part of the software campus program. Thus, the research needed to be conducted within the hard constraints of the project. In summary, this thesis dedicated most of the resources into technological advancement to enable the creation of a VR-ITS for pedestrian safety, but further research in bigger scale focused on the pedagogical and psychological aspect for such a system are still lacking.

8.4 Future Work

As for the last section of this thesis, the following paragraphs provide an outlook about different directions for future work based on the foundation created by this dissertation and the SafeChild system.

8.4.1 Interface

In terms of VR interfaces, the recent years have brought a number of novel devices into the market with others being announced or in prototype stage. These devices were not in the scope of this thesis but could help in making more immersive VR interfaces feasible for a wider range of potential users. Just to name a few, Google Daydream[1], and Samsung GearVR[2] both work towards using smartphones as standalone VR devices (Figure 8.1). Both computation and display are handled by a smartphone which is put inside an HMD. Future work could investigate into using these type of devices as a highly portable and increasingly affordable solution for immersive VR pedestrian safety training.

(a) “Google Daydream [Online image]” (2018) (b) “Samsung GearVR [Online image]” (n.d.)

Figure 8.1: Smartphone-based VR HMDs

In addition to smartphone-based VR devices, the use of Augmented Reality (AR) could also be an interesting option for future work where virtual objects (such as cars) are augmented onto real-life roads. A promising but still costly device would be the Microsoft Hololens[3] shown in Figure 8.2 which is also compatible with the frameworks described in this thesis. As already mentioned in Chapter 5, special devices for locomotion in VR could also become interesting at some point for child pedestrian safety education in VR.

[1] https://vr.google.com/daydream/, accessed 04-09-2018
[2] https://www.samsung.com/samsung/Gear_VR/, accessed 04-09-2018
[3] https://www.microsoft.com/en-us/hololens, accessed 04-09-2018

Figure 8.2: “Microsoft Hololens [Online image]” (n.d.)

8.4.2 Intelligent Tutoring

As mentioned in the limitations, the main focus of this thesis was to develop methods for Domain and (within exercise) Student Modeling for pedestrian safety education that is compatible with a VR interface. One key component of an ITS, the pedagogical model, was only implemented in a rudimentary way. Future work here needs to focus on how to use the information provided by the Student Model to deliver effective automatic pedagogical interventions to create a better learning experience. This includes, for instance, using voice in addition or as a replacement for textual instructions and feedback, aggregating exercise performance information across multiple exercises to infer more accurate information about a user’s skill level and to design more adaptation functionalities. These additions need to be then studied in large-scale user experiments with delayed and roadside post-tests to determine long-term effectiveness as well as skill transfer from VR to the real-world. Once a solid foundation is approved by these studies, additional traffic safety elements such as other pedestrians, pedestrian bridges or traffic regulating police officers could be introduced to widen the spectrum of training scenarios.

In addition to the child pedestrian safety domain, the methods for Domain and Student Modeling developed in this thesis could also be applied to other domains with similar characteristics. This includes for instance bicycle training and driving training. In terms of domains that are not related to traffic, different disciplines of sport also share the highly dynamic nature and ill-defined tasks. Teamplay and quick reaction are crucial here, and although there is a set of strategies, a complete plan for a whole game can usually not be predetermined.

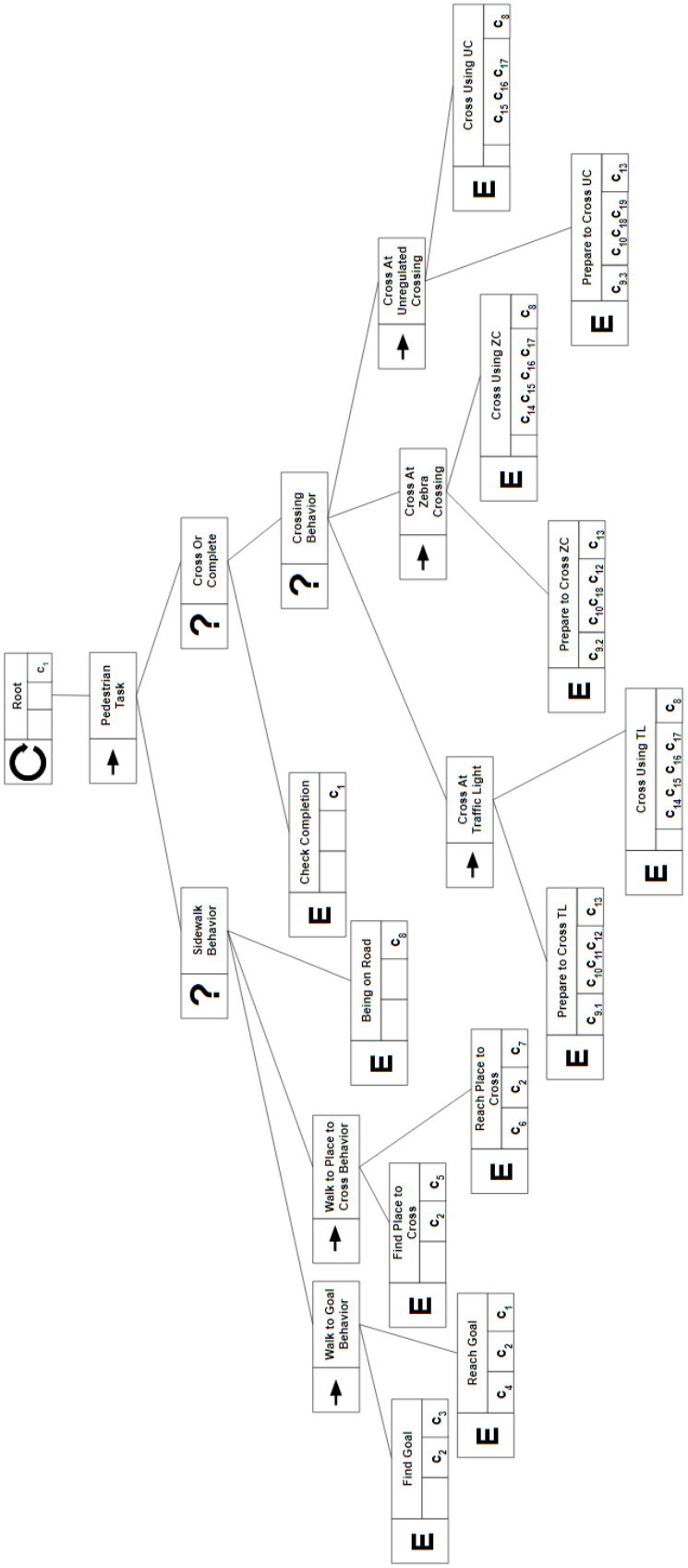
Appendix A

Pedestrian Safety Task Domain Model

A.1 Constraints

c_1	Pedestrian at goal
c_2	Keep distance to road
c_3	Goal found
c_4	Goal unreachable on sidewalk
c_5	Place to cross found
c_6	Entered road at wrong position
c_7	Place to cross reached
c_8	Reach Sidewalk
$c_{9.1}$	Leave traffic light area
$c_{9.2}$	Leave zebra crossing area
$c_{9.3}$	Leave unregulated crossing area
c_{10}	Stop at curb
c_{11}	Wait for green
c_{12}	Make sure that cars have stopped
c_{13}	Enter road
c_{14}	Cross at designated area
c_{15}	Observe cars while crossing
c_{16}	Cross straight
c_{17}	Cross without stopping
c_{18}	Look Left-Right-Left
c_{19}	Find a good time gap to cross

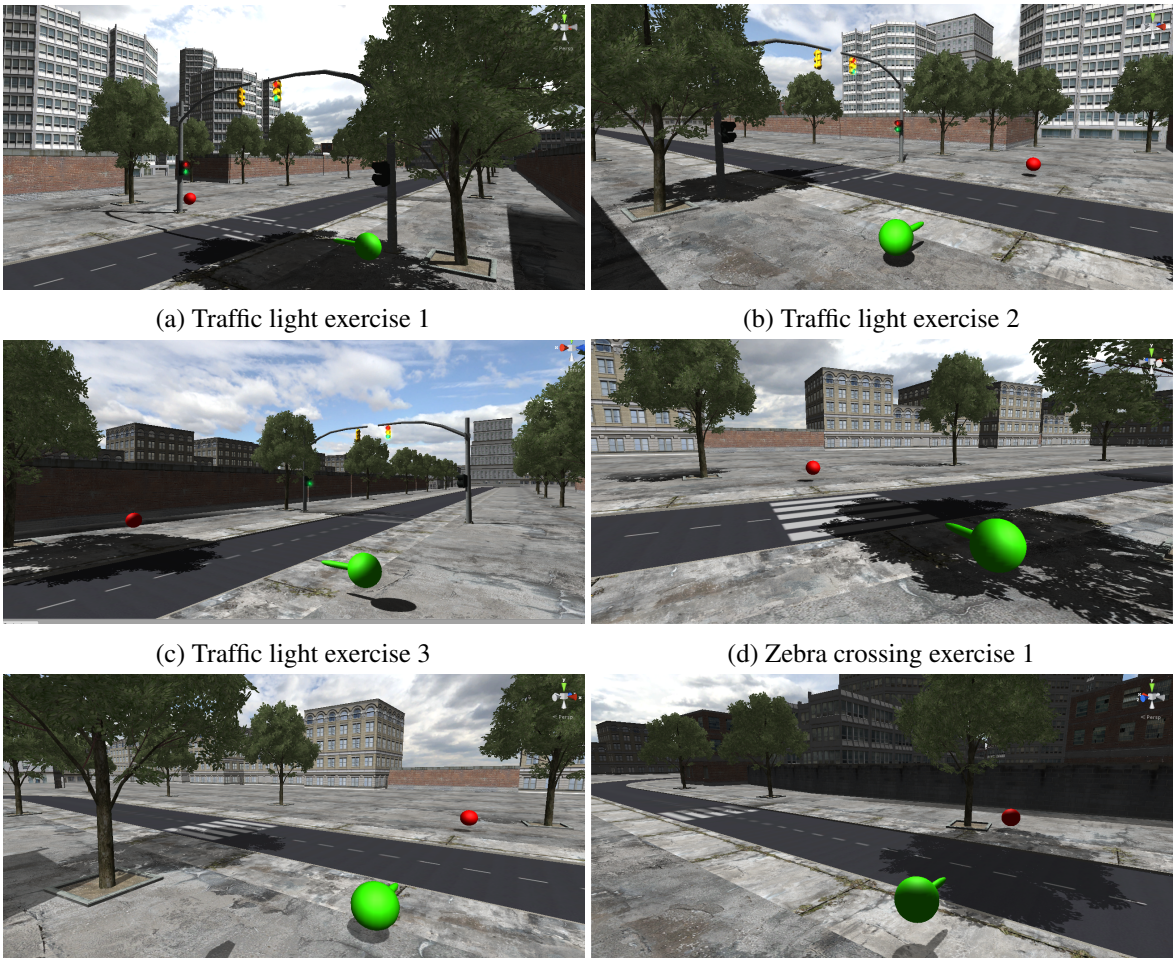
A.2 Expert Behavior Tree



Appendix B

Pilot Study Material

B.1 Exercises



(a) Traffic light exercise 1

(b) Traffic light exercise 2

(c) Traffic light exercise 3

(d) Zebra crossing exercise 1

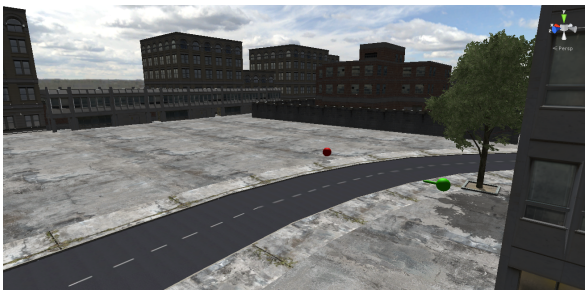
(e) Zebra crossing exercise 2

(f) Zebra crossing exercise 3

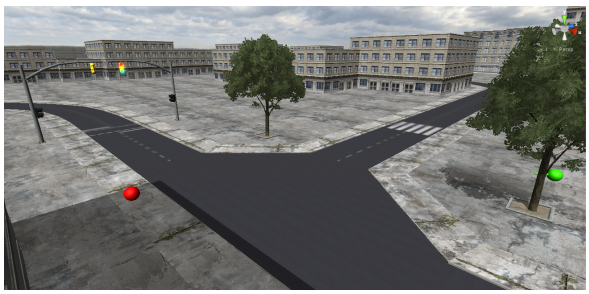
Figure B.1: Exercises 1 to 6 of the Pilot Study. The green ball indicates the starting point with the nose showing the initial view direction. The red ball shows the goal position.

(a) Unregulated crossing exercise 1

(b) Unregulated crossing exercise 2



(c) Unregulated crossing exercise 3

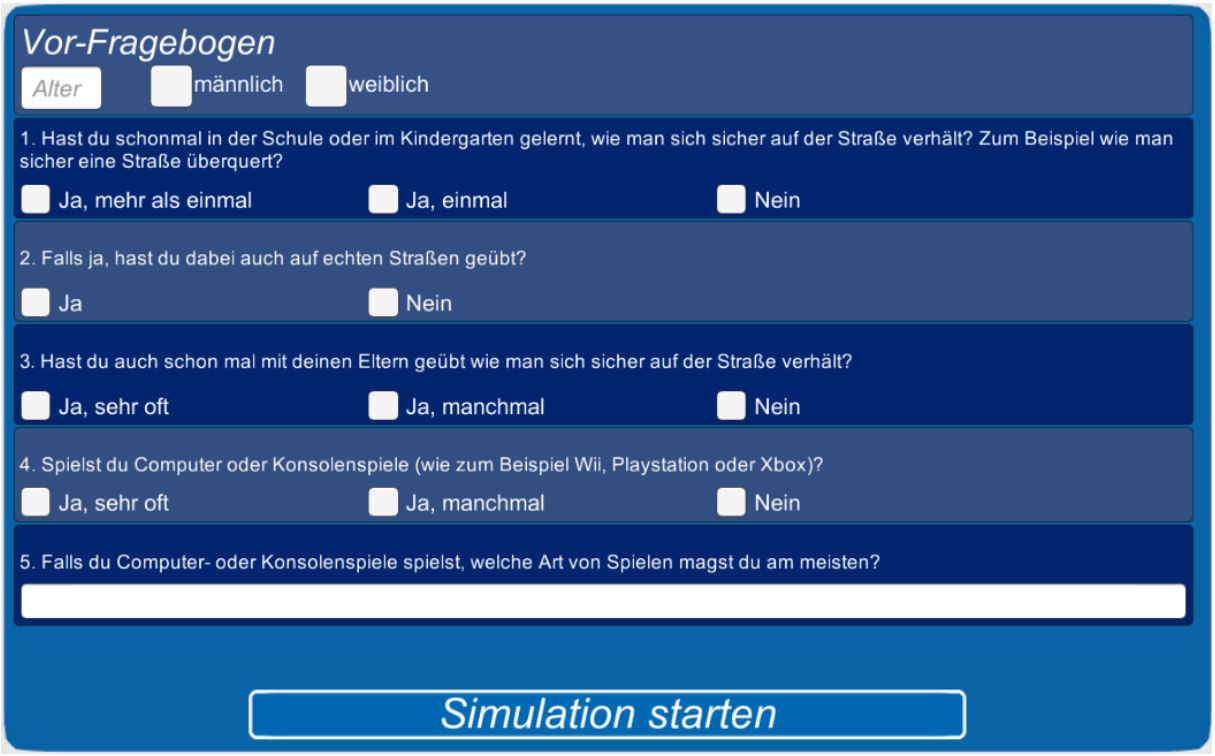


(d) Combined crossing exercise

Figure B.2: Exercises 7 to 10 of the Pilot Study. The green ball indicates the starting point with the nose showing the initial view direction. The red ball shows the goal position.

B.2 Pre-Questionnaire

B.2.1 German Original



B.2.2 English Translation

Header: Pre-Questionnaire (Input for age and selection of male or female)

Question 1: Did you learn safe behavior on roads at school or kindergarten?
(Answer options: "yes, more than once", "yes, once", "no")

Question 2: If yes, did you train on real roads?
(Answer options: "yes", "no")

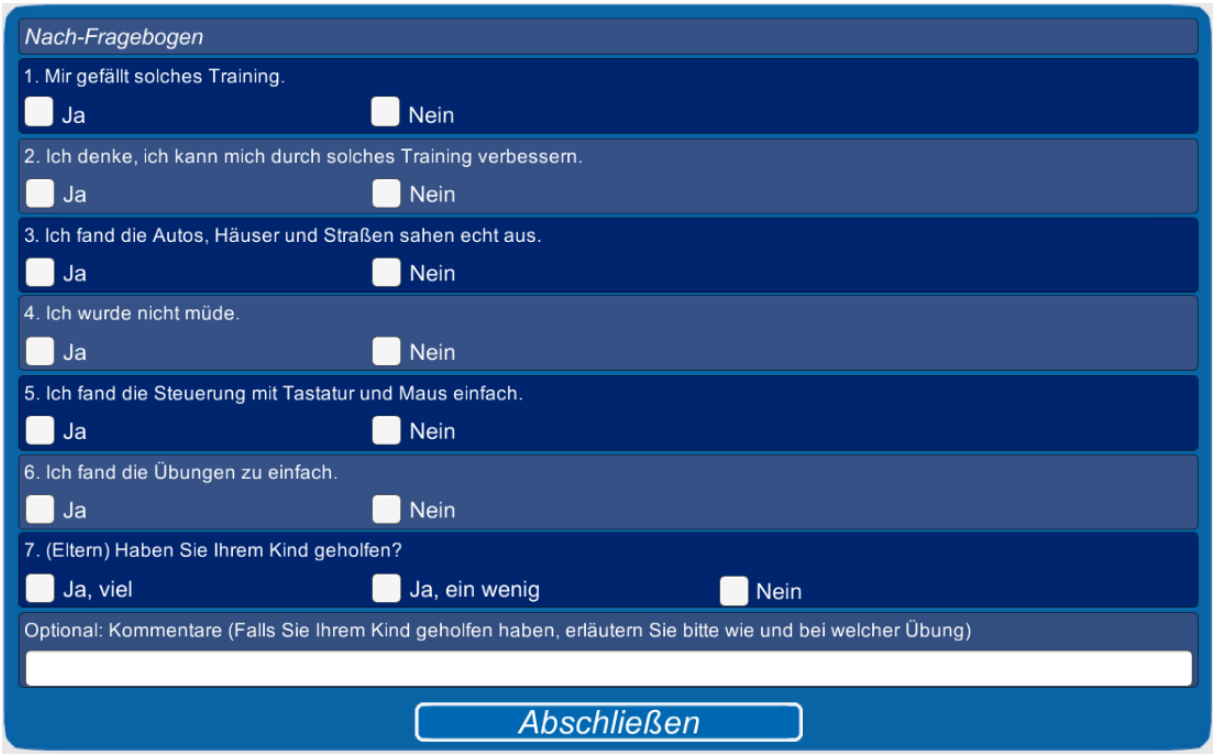
Question 3: Did you train safe roadside behavior with your parents?
(Answer options: "yes, very often", "yes, sometimes", "no")

Question 4: Do you play computer or console games? (such as Wii, Playstation or Xbox)
(Answer options: "yes, very often", "yes, sometimes", "no")

Question 5: If you play computer or console games, which kind of games do you like most?
(Answer options: free text)

B.3 Post-Questionnaire

B.3.1 German Original



B.3.2 English Translation

Question 1: I liked such training. (Answer options: "yes", "no")

Question 2: I think that I can improve through such training.
(Answer options: "yes", "no")

Question 3: I thought that cars, buildings and roads looked realistic.
(Answer options: "yes", "no")

Question 4: I did not get tired. (such as Wii, Playstation or Xbox)
(Answer options: "yes", "no")

Question 5: I found the controls with keyboard and mouse easy to use.
(Answer options: "yes", "no")

Question 6: I found the exercises too easy. (Answer options: "yes", "no")

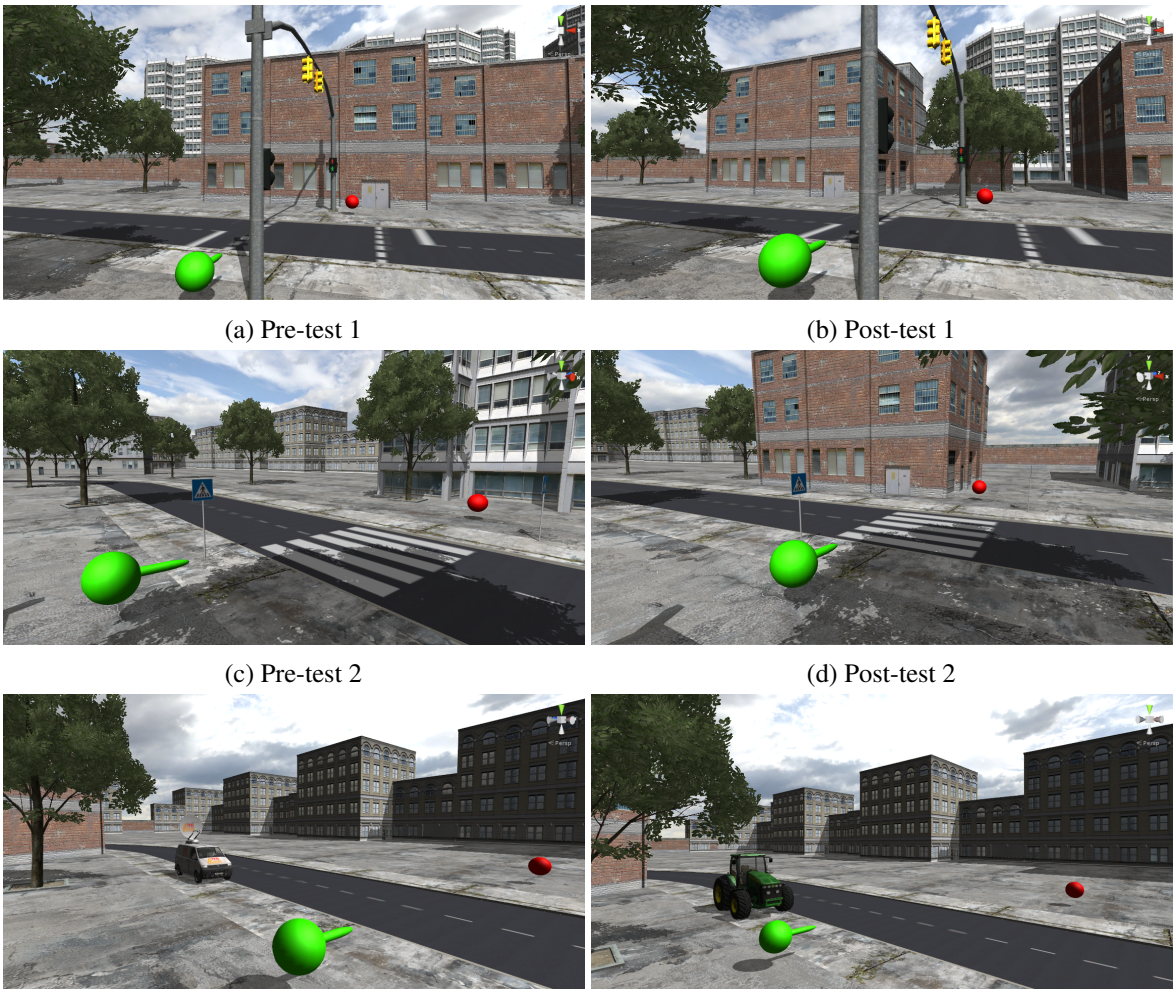
Question 7: (Parents) Did you help your child?
(Answer options: "yes, a lot", "yes, a little", "no")

Optional Comments: If you helped your child, please describe how and during which exercise.
(Answer options: free text)

Appendix C

Main Study Material

C.1 Pre- and Post-Tests



(a) Pre-test 1 (b) Post-test 1

(c) Pre-test 2 (d) Post-test 2

(e) Pre-test 3 (f) Post-test 3

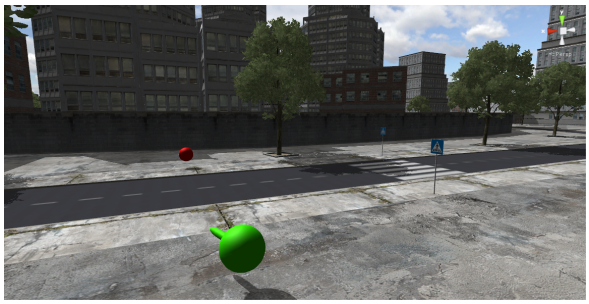
Figure C.1: Pre- and post-test exercises. The green ball indicates the starting point with the nose showing the initial view direction. The red ball shows the goal position.

C.2 Main Study Exercises

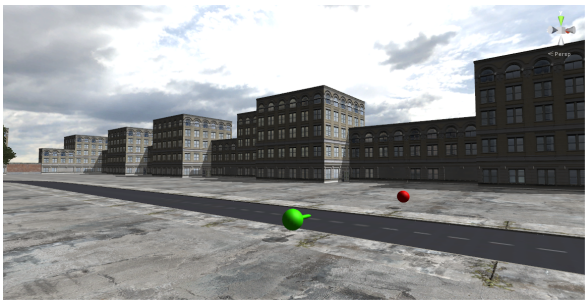
(a) Traffic light exercise 1

(b) Traffic light exercise 2

(c) Zebra crossing exercise 1

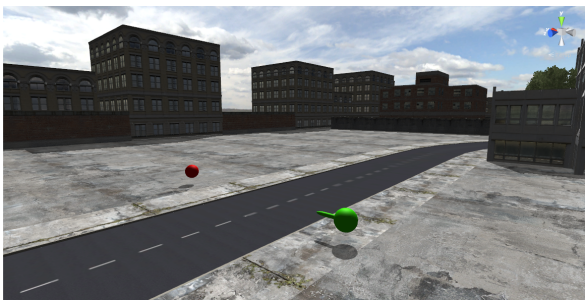


(d) Zebra crossing exercise 2



(e) Unregulated crossing exercise 1

(f) Unregulated crossing exercise 2



(g) Unregulated crossing exercise 3

Figure C.2: Main study exercises. The green ball indicates the starting point with the nose showing the initial view direction. The red ball shows the goal position.

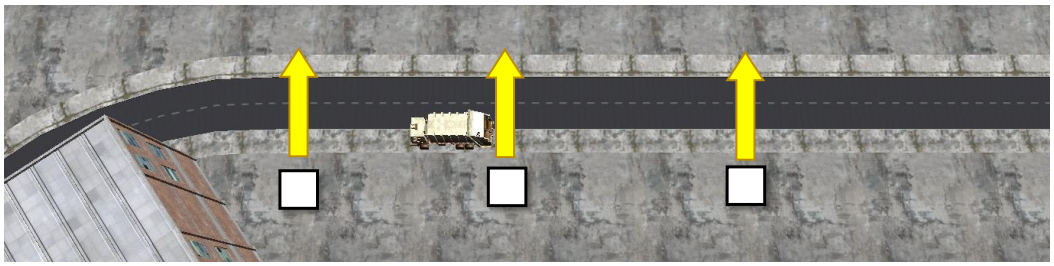
C.3 Pre-Questionnaire German

Vorher Fragebogen **Gruppe: ____ Rechner: ____**

Wie alt bist du? _____

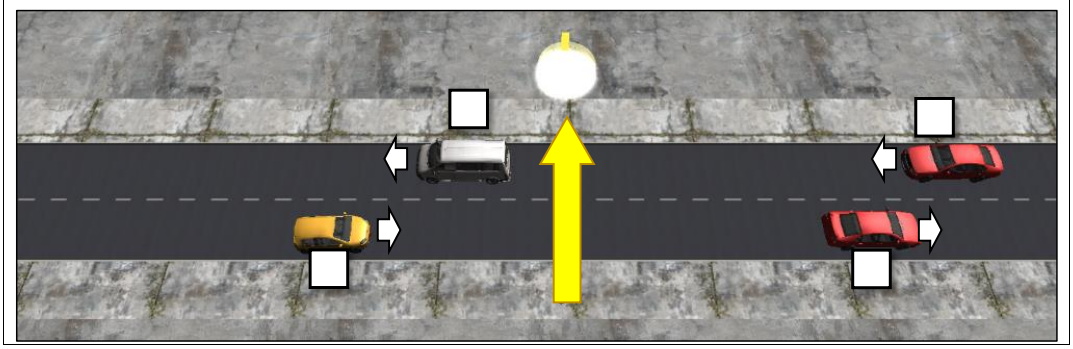
Bist du ein Junge oder ein Mädchen?	☐ Junge	☐ Mädchen
Überquerst du alleine Straßen auf dem Schulweg?	☐ Ja	☐ Nein
Fühlst du dich dabei sicher?	☐ Ja	☐ Nein
Spielst du Videospiele?	☐ Ja	☐ Nein

Frage 1: Kreuze die Stellen an, wo es unsicher ist, über die Straße zu gehen.



Frage 2: Kreuze an, an welchen Übergängen du auf den Verkehr achten musst.

Frage 3: Du willst beim gelben Pfeil die Straße überqueren. Kreuze an, welche Autos du noch durchlassen musst.



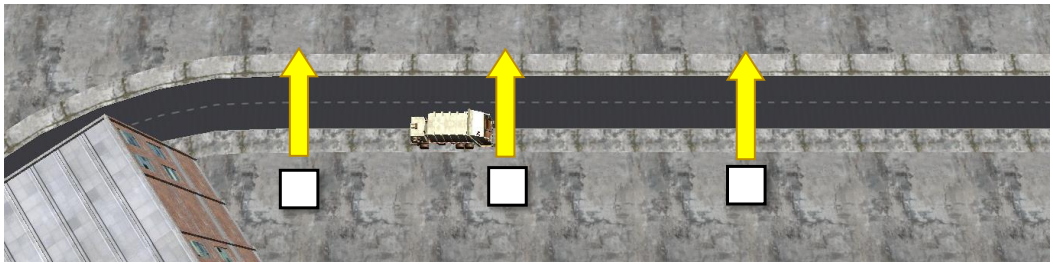
C.4 Pre-Questionnaire English

Pre Questionnaire **Group: ____ PC: ____**

How old are you? _____

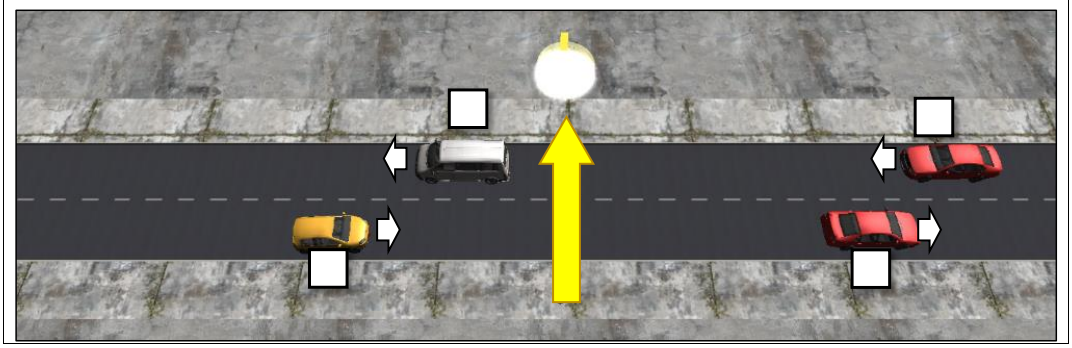
Are you a boy or a girl?	☐ Boy	☐ Girl
Do you cross roads alone on your way to school?	☐ Yes	☐ No
Do you feel safe while doing so?	☐ Yes	☐ No
Do you play video game?	☐ Yes	☐ No

Question 1: Mark those places where it is unsafe to cross the road.



Question 2: Mark those crossings where you need to pay attention to traffic.

Question 3: You want to cross at the yellow arrow. Mark which cars you need to let pass.

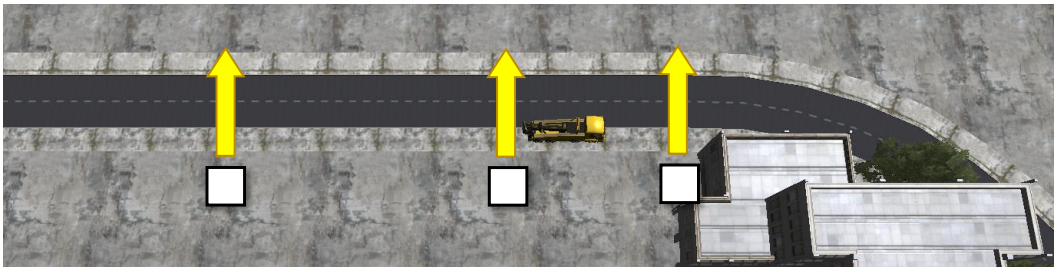


C.5 Post-Questionnaire German

Naher Fragebogen **Gruppe: ____ Rechner: ____**

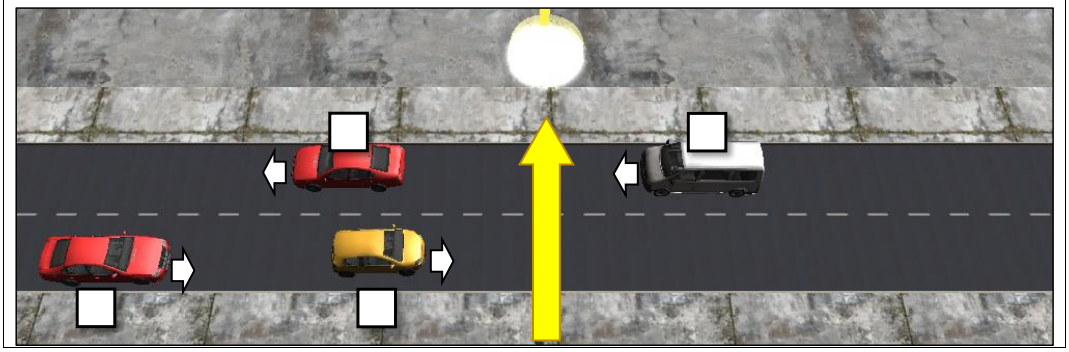
Hat dir das Training gefallen?	☐ Ja	☐ Nein
Hast du etwas gelernt?	☐ Ja	☐ Nein
War die Steuerung einfach?	☐ Ja	☐ Nein
Sahen die Autos, Straßen und Häuser echt aus?	☐ Ja	☐ Nein
Wurdest du müde?	☐ Ja	☐ Nein

Frage 1: Kreuze die Stellen an, wo es unsicher ist, über die Straße zu gehen.



Frage 2: Kreuze an, an welchen Übergängen du auf den Verkehr achten musst.

Frage 3: Du willst beim gelben Pfeil die Straße überqueren. Kreuze an, welche Autos du noch durchlassen musst.

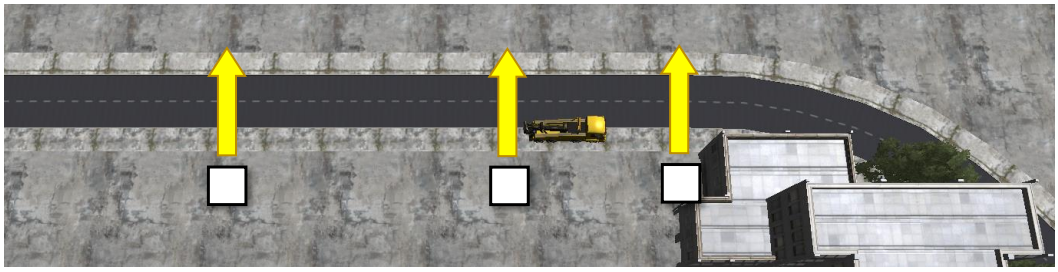


C.6 Post-Questionnaire English

Post Questionnaire **Group: ____ PC: ____**

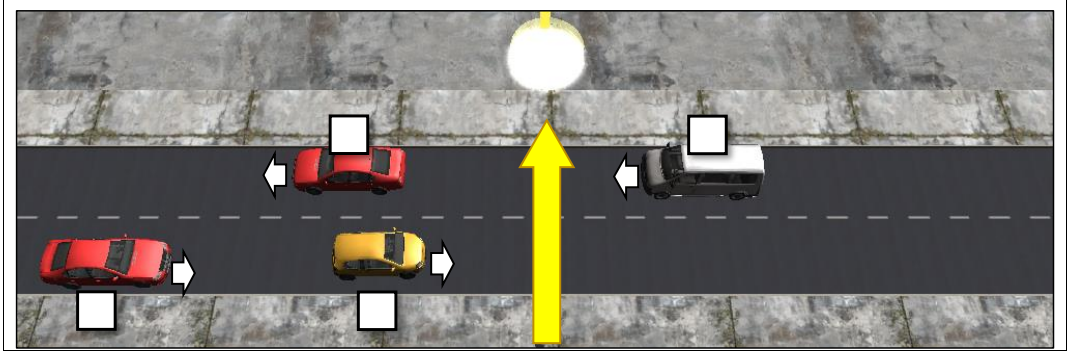
Did you like the training?	☐ Yes	☐ No
Did you learn something?	☐ Yes	☐ No
Were the controls easy to use?	☐ Yes	☐ No
Did the cars, roads and buildings look real?	☐ Yes	☐ No
Did you get tired?	☐ Yes	☐ No

Question 1: Mark those places where it is unsafe to cross the road.



Question 2: Mark those crossings where you need to pay attention to traffic.

Question 3: You want to cross at the yellow arrow. Mark which cars you need to let pass.



Bibliography

Ampofo-Boateng, K., Thomson, J., Grieve, R., Pitcainr, T., Lee, D., & Demetre, J. (1993). A developmental and training study of children's ability to find safe routes to cross the road. *British journal of developmental psychology*, *11*(1), 31–45.

Ampofo-Boateng, K. [Kwame] & Thomson, J. A. (1991). Children's perception of safety and danger on the road. *British journal of psychology*, *82*(4), 487–505.

Anderson, J. R. (1996). Act: A simple theory of complex cognition. *American Psychologist*, *51*(4), 355.

Anderson, J. R., Corbett, A. T., Koedinger, K. R., & Pelletier, R. (1995). Cognitive tutors: Lessons learned. *The journal of the learning sciences*, *4*(2), 167–207.

Bart, O., Katz, N., Weiss, P. L., & Josman, N. (2008). Street crossing by typically developed children in real and virtual environments. *OTJR: Occupation, Participation and Health*, *28*(2), 89–96.

Bloom, B. S. (1984). The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational researcher*, *13*(6), 4–16.

Brusilovsky, P. & Peylo, C. (2003). Adaptive and intelligent web-based educational systems. *International Journal of Artificial Intelligence in Education (IJAIED)*, *13*, 159–172.

Buche, C., Bossard, C., Querrec, R., & Chevaillier, P. (2010). Pegase: A generic and adaptable intelligent system for virtual reality learning environments. *International Journal of Virtual Reality*, *9*(2), 73–85.

Burton, R. R. & Brown, J. S. (1976). A tutoring and student modelling paradigm for gaming environments. *ACM SIGCUE Outlook*, *10*(SI), 236–246.

Carbonell, J. R. (1970). Ai in cai: An artificial-intelligence approach to computer-assisted instruction. *IEEE transactions on man-machine systems*, *11*(4), 190–202.

CAVE Automatic Virtual Environment [Online image]. (n.d.). Retrieved June 25, 2018 from https://i0.wp.com/www.technobyte.org/wp-content/uploads/2016/03/CAVE-Cave-Automatic-Virtual-Environment.jpg.

Chittaro, L., Buttussi, F., & Zangrando, N. (2014). Desktop virtual reality for emergency preparedness: User evaluation of an aircraft ditching experience under different fear arousal conditions. In *Proceedings of the 20th acm symposium on virtual reality software and technology* (pp. 141–150). ACM.

Coates, G. (1992). Program from invisible site-a virtual sho, a multimedia performance work presented by george coates performance works. *San Francisco, CA*.

Congiu, M., Whelan, M., Oxley, J., Charlton, J., D'Elia, A., & Muir, C. (2008). Child pedestrians: Factors associated with ability to cross roads safely and development of a training package. *Victoria: Monash University Accident Research Centre (MUARC)*.

Corbett, A. T. & Anderson, J. R. (1994). Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction*, *4*(4), 253–278.

Corbett, A. T., Koedinger, K. R., & Anderson, J. R. (1997). Intelligent tutoring systems. In *Handbook of human-computer interaction (second edition)* (pp. 849–874). Elsevier.

Cruz-Neira, C., Sandin, D. J., & DeFanti, T. A. (1993). Surround-screen projection-based virtual reality: The design and implementation of the cave. In *Proceedings of the 20th annual conference on computer graphics and interactive techniques* (pp. 135–142). ACM.

David, K. S. & Sullivan, M. (2005). Expectations for walking speeds: Standards for students in elementary schools. *Pediatric Physical Therapy*, *17*(2), 120–127.

Dromey, R. G. (2003). From requirements to design: Formalizing the key steps. In *Software engineering and formal methods, 2003. proceedings. first international conference on* (pp. 2–11). IEEE.

Duperrex, O., Bunn, F., & Roberts, I. (2002). Safety education of pedestrians for injury prevention: A systematic review of randomised controlled trials. *Bmj*, *324*(7346), 1129.

Full Motion Flight Simulator [Online image]. (n.d.). Retrieved June 25, 2018 from https://abellaerospace.com/wp-content/uploads/2018/02/full-flight-33.jpg.

Furness III, T. A. (1986). The super cockpit and its human factors challenges. In *Proceedings of the human factors society annual meeting* (Vol. 30, *1*, pp. 48–52). SAGE Publications Sage CA: Los Angeles, CA.

Gigante, M. A. (1993). Virtual reality: Definitions, history and applications. In *Virtual reality systems* (pp. 3–14). Elsevier.

Ginsburg, H. P. & Opper, S. (1988). *Piaget's theory of intellectual development*. Prentice-Hall, Inc.

Goetgeluk, J. (2013). Virtuix omni [online image]. https://commons.wikimedia.org/wiki/File:Virtuix_Omni_Skyrim,_with_logo.JPG. Retrieved June 25, 2018.

Google Daydream [Online image]. (2018). Retrieved September 08, 2018 from https://vr.google.com/daydream/static/images/pages/home/daydream-view-fog.png.

Grantcharov, T. P., Kristiansen, V., Bendix, J., Bardram, L., Rosenberg, J., & Funch-Jensen, P. (2004). Randomized clinical trial of virtual reality simulation for laparoscopic skills training. *British journal of surgery*, *91*(2), 146–150.

Hart, S. G. & Staveland, L. E. (1988). Development of nasa-tlx (task load index): Results of empirical and theoretical research. In *Advances in psychology* (Vol. 52, pp. 139–183). Elsevier.

Hattie, J. & Timperley, H. (2007). The power of feedback. *Review of educational research*, *77*(1), 81–112.

Hill, R., Lewis, V., & Dunbar, G. (2000). Young children's concepts of danger. *British Journal of Developmental Psychology*, *18*(1), 103–119.

HTC Vive Controller [Online image]. (n.d.). Retrieved June 25, 2018 from https://www.vive.com/media/filer_public/ac/85/ac8560e4-8d7f-42b6-9394-8fa6d5064b4e/controller_01.jpg.

Isla, D. (2005). Managing complexity in the halo 2 ai system. In *Proceedings of the game developers conference* (Vol. 63).

Johnson, W. L. & Rickel, J. (1997). Steve: An animated pedagogical agent for procedural training in virtual environments. *ACM SIGART Bulletin*, *8*(1-4), 16–21.

Kodaganallur, V., Weitz, R. R., & Rosenthal, D. (2006). An assessment of constraint-based tutors: A response to mitrovic and ohlsson's critique of "a comparison of model-tracing and constraint-based intelligent tutoring paradigms". *International Journal of Artificial Intelligence in Education*, *16*(3), 291–321.

Kulik, J. A. & Fletcher, J. (2016). Effectiveness of intelligent tutoring systems: A meta-analytic review. *Review of Educational Research*, *86*(1), 42–78.

Lane, H. C. [H Chad] & Johnson, W. L. (2008). Intelligent tutoring and pedagogical experience manipulation in virtual learning environments. *The PSI handbook of virtual environments for training and education*, *3*.

Lane, H. C. [H. Chad], Core, M., Gomboc, D., Karnavat, A., & Rosenberg, M. (2007). Intelligent Tutoring for Interpersonal and Intercultural Skills. In *Interservice/Industry Training, Simulation and Education Conference (I/ITSEC)*.

LapSim System [Online image]. (n.d.). Retrieved June 25, 2018 from http://www.surgicalproductsmag.com/sites/surgicalproductsmag.com/files/legacyimages/1005/lapsim.jpg.

Leap Motion [Online image]. (n.d.). Retrieved June 25, 2018 from https://en.wikipedia.org/wiki/File:Atomic_force_microscope_block_diagram.svg.

Livak, T., Heffernan, N., & Moyer, D. (2004). Using cognitive models for computer generated forces and human tutoring. In *13th annual conference on (brims) behavior representation in modeling and simulation. simulation interoperability standards organization. arlington, va. summer 2004*.

Lynch, C., Ashley, K., Aleven, V., & Pinkwart, N. (2006). Defining ill-defined domains; a literature survey. In *Proceedings of the workshop on intelligent tutoring systems for ill-defined domains at the 8th international conference on intelligent tutoring systems* (pp. 1–10).

Marion, N., Septseault, C., Boudinot, A., & Querrec, R. (2007). Gaspar: Aviation management on an aircraft carrier using virtual reality. In *Cyberworlds, 2007. cw'07. international conference on* (pp. 15–22). IEEE.

Marzinotto, A., Colledanchise, M., Smith, C., & Ögren, P. (2014). Towards a unified behavior trees framework for robot control. In *Robotics and automation (icra), 2014 ieee international conference on* (pp. 5420–5427). IEEE.

Mayo, M., Mitrovic, A., & McKenzie, J. (2000). Capit: An intelligent tutoring system for capitalisation and punctuation. In *Advanced learning technologies, 2000. iwalt 2000. proceedings. international workshop on* (pp. 151–154). IEEE.

McComas, J., MacKay, M., & Pivik, J. (2002). Effectiveness of virtual reality for teaching pedestrian safety. *CyberPsychology & Behavior*, *5*(3), 185–190.

Microsoft Hololens [Online image]. (n.d.). Retrieved September 08, 2018 from https://img-prod-cms-rt-microsoft-com.akamaized.net/cms/api/am/imageFileData/RE1GJzk.

Microsoft Kinect [Online image]. (n.d.). Retrieved June 25, 2018 from https://i-msdn.sec.s-msft.com/dynimg/IC534687.png.

Mike Koozmin. (2014). Indoor pedestrian training [Online image]. http://s79f01z693v3ecoes3yyjsg1.wpengine.netdna-cdn.com/wp-content/uploads/2016/01/streetsmart1a2.jpg. Retrieved June 25, 2018.

Mitrovic, A. (2002). Normit: A web-enabled tutor for database normalization. In *Computers in education, 2002. proceedings. international conference on* (pp. 1276–1280). IEEE.

Mitrovic, A. (2003). An intelligent sql tutor on the web. *International Journal of Artificial Intelligence in Education*, *13*(2-4), 173–197.

Mitrovic, A., Koedinger, K. R., & Martin, B. (2003). A comparative analysis of cognitive tutoring and constraint-based modeling. In *International conference on user modeling* (pp. 313–322). Springer.

Mitrovic, A., Mathews, M., Ohlsson, S., Holland, J., McKinlay, A., Ogden, S., … Dopping-Hepenstal, S. (2014). A virtual reality environment for prospective memory training.

Mitrovic, A., Suraweera, P., Martin, B., & Weerasinghe, A. (2004). Db-suite: Experiences with three intelligent, web-based database tutors. *Journal of Interactive Learning Research*, *15*(4), 409.

Mott, B. W. & Lester, J. C. (2006). U-director: A decision-theoretic narrative planning architecture for storytelling environments. In *Proceedings of the fifth international joint conference on autonomous agents and multiagent systems* (pp. 977–984). ACM.

Mujber, T. S., Szecsi, T., & Hashmi, M. S. (2004). Virtual reality applications in manufacturing process simulation. *Journal of materials processing technology*, *155*, 1834–1838.

Nvidia 3D Vision Surround [Online image]. (n.d.). Retrieved June 25, 2018 from https://hothardware.com/articleimages/Item1526/3ds_surround.jpg.

Nwana, H. S. (1990). Intelligent tutoring systems: An overview. *Artificial Intelligence Review*, *4*(4), 251–277.

Oculus Rift [Online image]. (n.d.). Retrieved June 25, 2018 from https://images-na.ssl-images-amazon.com/images/I/61NybFdk4NL._SL1300_.jpg.

Ogren, P. (2012). Increasing modularity of uav control systems using computer game behavior trees. In *Aiaa guidance, navigation, and control conference* (p. 4458).

Ohlsson, S. (1994). Constraint-based student modeling. In *Student modelling: The key to individualized knowledge-based instruction* (pp. 167–189). Springer.

Ohlsson, S. & Rees, E. (1991). The function of conceptual understanding in the learning of arithmetic procedures. *Cognition and Instruction*, *8*(2), 103–179.

Orlosky, J., Weber, M., Gu, Y., Sonntag, D., & Sosnovsky, S. (2015). An interactive pedestrian environment simulator for cognitive monitoring and evaluation. In *Proceedings of the 20th international conference on intelligent user interfaces companion* (pp. 57–60). ACM.

Oron-Gilad, T., Meir, A., Tapiro, H., & Borowsky, A. (2011). Towards understanding child-pedestrian's deficits in perceiving hazards when crossing the road, final report. Negev: Ben-Gurion University, Human Factors Engineering Laboratory.

Percer, J. (2009). *Child pedestrian safety education: Applying learning and developmental theories to develop safe street-crossing behaviors*.

Robertson, G. G., Card, S. K., & Mackinlay, J. D. (1993). Three views of virtual reality: Nonimmersive virtual reality. *Computer*, *26*(2), 81.

Rumbaugh, J., Jacobson, I., & Booch, G. (2010). *Unified modeling language reference manual* (2nd). Addison-Wesley Professional.

Rump, P. (2014). *Knowledge tracing: Modeling the acquisition of procedural knowledge* (Master's thesis). Saarland University, Saarland, Germany.

Samsung GearVR [Online image]. (n.d.). Retrieved September 08, 2018 from https://s7d2.scene7.com/is/image/SamsungUS/Gear_VR_Gallery_081917?\protect\T1\textdollarproduct-details-jpg\protect\T1\textdollar.

Schwebel, D. C., Barton, B. K., Shen, J., Wells, H. L., Bogar, A., Heath, G., & McCullough, D. (2014). Systematic review and meta-analysis of behavioral interventions to improve child pedestrian safety. *Journal of pediatric psychology*, *39*(8), 826–845.

Schwebel, D. C., Gaines, J., & Severson, J. (2008). Validation of virtual reality as a tool to understand and prevent child pedestrian injury. *Accident Analysis & Prevention*, *40*(4), 1394–1400.

Schwebel, D. C. & McClure, L. A. (2010). Using virtual reality to train children in safe street-crossing skills. *Injury prevention*, *16*(1), e1–e1.

Schwebel, D. C., McClure, L. A., & Severson, J. (2014). Usability and feasibility of an internet-based virtual pedestrian environment to teach children to cross streets safely. *Virtual reality*, *18*(1), 5–11.

SDERA. (n.d.). Izzy's road safety games [online image]. http://izzygames.sdera.wa.edu.au/. Retrieved June 25, 2018.

Shabani, K., Khatib, M., & Ebadi, S. (2010). Vygotsky's zone of proximal development: Instructional implications and teachers' professional development. *English language teaching*, *3*(4), 237–248.

Sitzmann, T. (2011). A meta-analytic examination of the instructional effectiveness of computer-based simulation games. *Personnel psychology*, *64*(2), 489–528.

Steuer, J. (1992). Defining virtual reality: Dimensions determining telepresence. *Journal of communication*, *42*(4), 73–93.

Suraweera, P. & Mitrovic, A. (2002). Kermit: A constraint-based tutor for database modeling. In *International conference on intelligent tutoring systems* (pp. 377–387). Springer.

Sutherland, I. E. (1965). The ultimate display. *Multimedia: From Wagner to virtual reality*, 506–508.

Sutherland, I. E. (1968). A head-mounted three dimensional display. In *Proceedings of the december 9-11, 1968, fall joint computer conference, part i* (pp. 757–764). ACM.

Thalmic Labs Myo [Online image]. (n.d.). Retrieved June 25, 2018 from https://rainemagazine.com/wp-content/uploads/2014/11/black_myo_top1.jpg.

Thomson, J., Tolmie, A. K., Foot, H., & McLaren, B. (1996). Child development and the aims of road safety education. HMSO. Retrieved from https://strathprints.strath.ac.uk/18694/

Thomson, J. A., Tolmie, A. K., Foot, H. C., Whelan, K. M., Sarvary, P., & Morrison, S. (2005). Influence of virtual reality training on the roadside crossing judgments of child pedestrians. *Journal of Experimental Psychology: Applied*, *11*(3), 175.

Toroyan, T., Peden, M. et al. (2007). Youth and road safety. In *Youth and road safety*. OMS.

Underwood, J., Dillon, G., Farnsworth, B., & Twiner, A. (2007). Reading the road: The influence of age and sex on child pedestrians' perceptions of road risk. *British journal of psychology*, *98*(1), 93–110.

Van der Molen, H. H., Rothengatter, J., & Vinje, M. P. (1981). Blueprint of an analysis of the pedestrian's task—i: Method of analysis. *Accident Analysis & Prevention*, *13*(3), 175–191.

Vanlehn, K. (2006). The behavior of tutoring systems. *International journal of artificial intelligence in education*, *16*(3), 227–265.

Virtual Aviation [Online image]. (n.d.). Retrieved June 25, 2018 from https://media-cdn.tripadvisor.com/media/photo-s/01/c2/a7/ca/boeing-737-800-flight.jpg.

Warwitz, S. (2009). *Verkehrserziehung vom kinde aus* (6th ed.). Schneider.

WikiHow. (n.d.). Walking safely with a child [online image]. Retrieved June 25, 2018 from https://www.wikihow.com/Teach-Children-Basic-Street-Safety-when-Walking.

Winn, W. (1993). A conceptual basis for educational applications of virtual reality. *Technical Publication R-93-9, Human Interface Technology Laboratory of the Washington Technology Center, Seattle: University of Washington*. Retrieved October 05, 2018 from http://www.hitl.washington.edu/research/education/winn/winn-paper.html~.

Youngblut, C. (1998). *Educational uses of virtual reality technology.* Institute for defense analysis Alexandria VA.