Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Yrjar Gedde
Fredrik Bjørnland

Exploring Procedural Content Generation and Personalized Learning for Virtual Driving Education

Master's thesis in Computer Science
Supervisor: Odd Erik Gundersen
Co-supervisor: Johannes Rehm
June 2023

NTNU
Norwegian University of
Science and Technology

Abstract

As the world increasingly embraces digitalization, the use of virtual simulators for educational purposes has gained widespread recognition and adoption. The traffic school WAY has been using virtual driving simulators in their education to reduce risk, save money, and train students on situations that rarely happen in real life. To exploit the capabilities of virtual simulators and optimize their effectiveness, WAY wants to personalize lessons with tailor-made content based on each student's individual skill level. This thesis explores the application of Procedural Content Generation (PCG) and personalized learning in the context of virtual driving simulators, with the aim of creating varying and personalized traffic situations for WAY's driving education. By using the skill levels provided by WAY's existing skill model, the aim is to enhance learning through optimal difficulty and traffic situations tailored to best suit the student. The selected methodology utilized when exploring the feasibility of this research encompasses two phases. First, a literature review providing an overview of the state of the art in procedural content generation and personalized learning is conducted. Subsequently, a Design Science Research project is employed to evaluate the suitability. The identified literature reveals a taxonomy of PCG variations and a framework for experience-driven content generation, both forming the foundation for the proposed model. The contributions of the Design Science Research project include a semantic representation for depicting traffic situations and driving lessons, an algorithm for generating personalized content, an analysis of the generated content, and proposals for future research directions. The analysis was conducted by creating scenarios of skill levels with special characteristics before an analysis of the correlation between skill levels and trained skills and difficulty levels was done. The results indicate that the model succeeds in personalizing content through promising skill coverage and difficulty adjustments while also highlighting the challenges and limitations.

Sammendrag

I takt med den økte digitaliseringen i verden, har bruken av virtuelle simulatorer til pedagogiske formål blitt anerkjent og tatt i bruk i stadig økende grad. Trafikkskolen WAY har benyttet virtuelle kjøresimulatorer i opplæringen sin for å redusere risiko, spare kostnader og trene elevene på situasjoner som sjelden oppstår i virkeligheten. For å utnytte potensialet til virtuelle simulatorer og optimalisere effektiviteten, ønsker WAY å tilpasse undervisningen med skreddersydd innhold basert på hver enkelt elevs individuelle ferdighetsnivå. Denne avhandlingen utforsker anvendelsen av *Procedural Content Generation* (PCG) og *Personalized Learning* i konteksten til virtuelle kjøresimulatorer. Målet med dette er å skape varierte og personlig tilpassede trafikksituasjoner for WAYs kjøreopplæring. Ved å benytte ferdighetsnivåene som tilbys av WAYs eksisterende ferdighetsmodell, er målet å forbedre læringen gjennom optimal vanskelighetsgrad og trafikksituasjoner skreddersydd for den enkelte elev. Den valgte metodologien som benyttes for å utforske gjennomførbarheten av denne forskningen omfatter to faser. Først gjennomføres en litteraturgjennomgang som gir en oversikt over den siste utviklingen innen *Procedural Content Generation* og *Personalized Learning*. Deretter benyttes et forskningsprosjekt basert på *Design Science Research* for å evaluere annvendelsesomeråde. Litteraturgjennomgangen avdekker en klassifisering for PCG og en rammeverk for erfaringsbasert innholdsgenerering. Disse legger grunnlaget for den foreslåtte modellen. Bidragene fra forskningsprosjektet inkluderer en semantisk representasjon for å skildre trafikksituasjoner og kjøreopplæring, en algoritme for generering av personlig tilpasset innhold, en analyse av det genererte innholdet og tilslutt, forslag til fremtidig arbeid. Analysen ble gjennomført ved å skape scenarier med spesielle egenskaper for ulike ferdighetsnivåer, før en analyse av sammenhengen mellom ferdighetsnivåer, trente ferdigheter og vanskelighetsnivåer ble utført. Resultatene indikerer at modellen lykkes med å personalisere innholdet gjennom lovende dekning av ferdigheter og justeringer av vanskelighetsgrad, samtidig som utfordringer og begrensninger blir fremhevet.

Preface

This thesis is written in relation to the course TDT4900 - Computer Science, Master's thesis at the Norwegian University of Science and Technology. The research was conducted during the spring of 2023 and builds upon the work carried out in the project thesis from the fall of 2022. We want to express our gratitude towards our supervisor Odd Erik Gundersen, and co-supervisor Johannes Rehm for providing valuable feedback and challenging us along the way. Additionally, we would like to express special thanks to WAY developer Øyvind Eriksen for assisting us in ensuring compatibility with the existing Unity solution and Irina Reshodko for introducing the key components of the skill model. Lastly, we are thankful for the opportunity to work in cooperation with WAY as a company. They have been nothing but supportive, and we have learned a lot about research in a commercial environment.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter serves as an introductory section for the thesis, providing an overview of the motivation, problem description, research methodology, our contributions, and the structure of the thesis. As an emerging area of research, Procedural Content Generation (PCG) offers promising advances in developing highly interactive and dynamic virtual worlds [36]. This Master's thesis centers around the implementation of (PCG) to generate diverse traffic scenarios for the driving school WAY, utilizing their virtual driving simulator, built using the Unity platform. The generated traffic situations will be tailored to the WAY students, assisting them in their preparation for obtaining their driving license. Using PCG, we can generate various traffic scenarios that help students prepare for real-world driving situations. Furthermore, this thesis explores the concept of personalized learning within this context, where the simulator adapts to each student's progress and unique learning curve. This approach ensures that students are consistently challenged according to their skill level, thus enhancing the learning process [12]. Integrating PCG and personalized learning into driving simulators offers a potentially transformative approach to driving education, creating a more engaging, immersive, and effective learning experience for students as they work towards obtaining their driving license.

1.1 Motivation

Acquiring a driver's license involves gaining knowledge and proficiency in a diverse range of concepts and skills across different environments and scenarios. To ensure the most effective skill development, it is important to acknowledge the diverse backgrounds of students, their varying skill sets, and their unique cognitive abilities. Consequently, individually tailored training becomes crucial to accommodate these differences and ensure effective learning outcomes [29]. By offering personalized experiences, virtual simulators can potentially improve learning outcomes and increase driver motivation [30], thus facilitating a faster and safer journey toward acquiring their driver's license.

Utilizing a virtual simulator with personalized content to teach driving and traffic skills has several notable advantages. Firstly, the use of a virtual simulator eliminates the inherent risks associated with real-life driving, providing a safer learning environment. This enables the exposure of students to more advanced and challenging scenarios at an earlier stage, potentially leading to accelerated learning progress. Another significant benefit is the ability to control and customize the traffic situation a student encounters during the lesson, thus allowing for the design of optimal challenges for enhanced learning. Similarly, a personalized environment can ensure that the difficulty level is appropriately adjusted to each student's abilities and progress. Employing a virtual simulator with automatically generated personalized content can, in addition, reduce the monitoring demands from driving instructors. Unlike traditional driving lessons that require a one-to-one instructor-student ratio, a virtual simulator facilitates a many-to-one relationship, where instructors can guide and support multiple students simultaneously.

WAY's current approach involves a set of predefined generic lessons in their driving simulator, through which students can undergo training. The selection of which lesson should be applied is made by the instructors, who manually find a fitting lesson for a given student and guide them to desired traffic situations to train the student on a particular skill. By doing this, they utilize some of the benefits described earlier. Still, the personalization of content relies heavily on the instructors' subjective meaning and guidance in the virtual environment. In addition to the unnecessary time spent finding optimal traffic situations both in-game and in preparation for a lesson, this approach only facilitates one-to-one guidance. The manual design of personalized training lessons for individual students is also too resource intensive, both in terms of expenses and time requirements, rendering it impractical for a larger group of students. However, WAY has already developed a skill model for automatically grading the student's performance, elaborated further in section 2.3. This skill model has a profile for each student with their respective performance on skills needed to master driving, thereby enabling the implementation of personalized generated content.

1.2 Problem Description

Based on the motivation presented above, we have formulated the following objective.

Objective *Investigate the feasibility of leveraging WAY's Unity-based simulator to automatically generate customized virtual driving lessons tailored to individual skill levels*

This objective aims to explore the potential of utilizing technology to generate personalized learning experiences in WAY's simulator. This research seeks to determine whether such an approach is viable and effective by examining the ability to create lessons based on specific skill levels. The investigation will evaluate the feasibility of automated customization, considering factors such as individualization, traffic situation representation, lesson content adaptation, and the overall impact on learning outcomes. To aid in achieving this, we have divided the objective further down into the following research questions:

1.2.1 Research Questions

RQ1 *How can we facilitate the creation of personalized driving lessons to optimize learning?*

WAY's existing skill model provides a solid foundation for tailoring content to individuals. Achieving a quality result necessitates a good translation of skill levels into traffic scenarios.

RQ2 *How can we effectively represent all traffic scenarios and combine them into a driving lesson?*

To enable the automatic generation of content, it is essential to establish an adaptable and robust representation of traffic scenarios. This representation requires the flexibility to effectively support the creation of every desirable situation and to combine the scenarios into a lesson.

RQ3 *How can we ensure an appropriate level of difficulty for each specific traffic situation?*

This research question explores methods to ensure that each traffic situation has an appropriate difficulty level. The aim is to investigate how to calibrate the difficulty level of different scenarios to match the skill level and learning demands of individual drivers.

1.3 Research Method

To address the research questions, the selected research methodology comprises two essential phases: a literature review and a Design Science Research (DSR). A Design Science Research (DSR) project aims to create novel and ground-breaking artifacts, which can be constructs, models, methods, and instantiations [4]. In this case, the artifact will be a model for procedural content generation of traffic situations in WAY's virtual driving simulator. The execution of a DSR project involves several stages, including problem identification, design and development, evaluation, and reflection. Through talks with WAY employees and a thorough literature review, the problem will be identified and specified to the research questions introduced in the section above. A system will then be designed and developed in the WAY Unity simulator, as described in chapter 5. Subsequently, the effectiveness and usefulness of the system will be evaluated through experiments made on simulated scenarios, introduced in chapter 6. This involves designing scenarios, collecting and analyzing the data that can be extracted from the generated lessons, and analyzing the gathered data to determine the system's feasibility.

1.4 Contributions

The thesis provides a set of contributions, further discussed in section 8.2. These contributions are as follows:

- A literature review providing an overview of procedural content generation and personalized learning in the context of virtual simulators.
- A semantic representation for depicting traffic situations and driving lessons.
- A model and algorithm for automatic generation of personalized content compatible with WAY's unity-based simulator.
- An analysis of the model and the generated content in light of the research questions listed earlier.
- Proposals for directions to extend the research in future work.

1.5 Thesis Structure

This thesis will investigate the research objective and research questions introduced in this chapter and is structured as follows. Chapter 2 provides an introduction to WAY, including its current utilization of the simulator, and presents the Bayesian Network skill model employed for progress monitoring. In chapter 3, we explore the core theoretical concepts that form the basis of the model design. Chapter 4 examines related work in the field of study and situates our research within this context, focusing on procedural content generation and personalized learning. In chapter 5, we establish the system requirements and present our architecture and model. Chapter 6 showcases a set of scenarios and the corresponding results. In chapter 7, we critically analyze the strengths and weaknesses of our solution, as well as potential improvements and possibilities. Finally, in chapter 8, the reader is presented with a conclusion on the research conducted and proposals for future work.

Chapter 2

WAY

WAY is a driving school company founded in June 2015 that actively seeks to digitize traffic training through automated feedback and virtual environments [1]. They intend to create a more efficient, environmentally friendly, cheaper, and safer path to acquire a class B driver's license. This chapter aims to provide information about WAY's virtual driving simulators, how they conduct and create their lessons, and their Bayesian Network that infers the skill level of their students.

2.1 Driving Simulator

Currently, WAY offers both traditional driving lessons and virtual lessons through the use of a 3D simulator. Virtual lessons are conducted in a full-scale 360-degree motion-based driving simulator with personal supervision from a licensed driving instructor. A real car is installed on a platform that can simulate movements and vibrations. The platform responds to the driver's actions and simulates acceleration, braking, and turning motions. This adds a physical dimension to the simulation, enhancing the realism and immersion. The driver interacts with the simulator using the car's controls, such as the steering wheel, pedals, and gearshift. The simulator software interprets the driver's inputs and adjusts the virtual environment accordingly. Real-time feedback is provided to the driver through visual cues, sounds, and haptic feedback from the motion platform. Constructive feedback is also provided by the driving instructor monitoring the session. After the lesson, the student's skill levels are inferred by a Bayesian Network, which will be introduced in section 2.3.

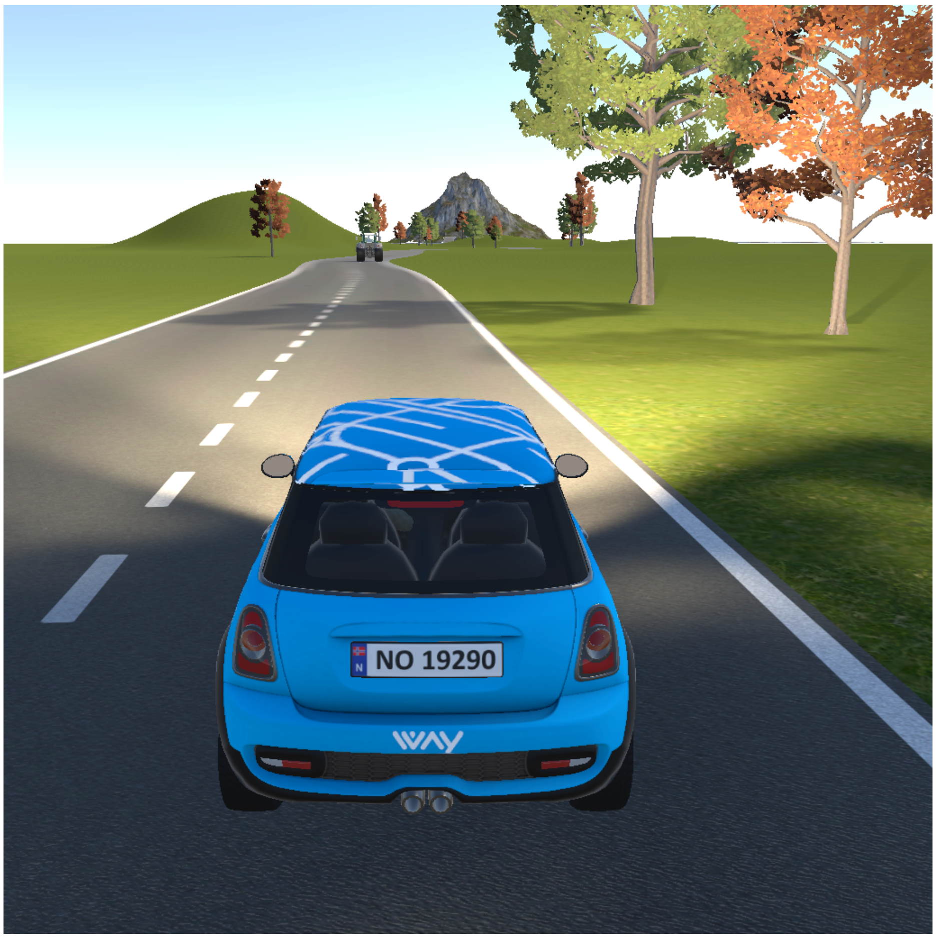


Figure 2.1: Screenshot from the Virtual Simulator

2.2 Lessons

The lessons the students participate in are comprehensively designed lessons that simulate real-life environments. Lessons covering all scenarios needed to obtain your driver's license have been designed and created in Unity, a cross-platform game development engine used to make video games, simulations, and other 3D applications. A curriculum of 24 lessons has been developed, with the objective of gradually increasing the complexity of each lesson. The initial lessons focus on fundamental driving skills, such as turning and braking, while the subsequent lessons introduce more intricate scenarios, including yielding and navigating traffic lights. The curriculum follows a progression that allows drivers to build upon their foundational skills and gradually develop their proficiency. Each lesson is designed to challenge drivers with new concepts and situations, ensuring a well-rounded understanding of traffic rules and practical driving techniques. Even though the curriculum, as seen in table 2.1, initially was planned as a complete and chronological guide towards acquiring your driver's license, driving instructors at WAY have stated that driving instructors must assess the student's skill level and driving history before choosing an appropriate lesson to maximize learning outcomes.

Lessons
1 Grunnkurs Dag 1
2 Grunnkurs Dag 2
3 Grunnkurs Dag 3
4 Grunnkurs Voksen
5 Boligmiljo
5 Boligmljo Uten Trafikk
6 Landevei - Kort Stans Ny Start
6 Landevei - Kort Stans Ny Start Advanced
7 Landevei - Stans Veikant
8 Landevei - Observasjon
9 Landevei
10 Landevei - Vinter
11 Bymiljo
12 By og motortrafikkvei
13 Landevei - Forbikjoring
14 Morkekjoring 1
15 Morkekjoring 2
16 Utrykning
17 Kartlegging
18 Tettsted Rundkjoring
OUS Kapittel 1 - Lavkontrast
OUS Kapittel 2 - Landevei
OUS Kapittel 3 - Morkekjoring
OUS Kapittel 4 - Bymiljo

Table 2.1: List of Current Lessons

Content in Unity is created and controlled through C# scripts and native Unity game objects. WAY's lessons consist of a large tile resembling a realistic scenario with multiple game objects. To create the roads on which the student drives, they utilize a Unity tool called EasyRoads3D. EasyRoads3D is a popular Unity plugin that allows developers to create realistic and dynamic road networks in their 3D game environments. It provides a user-friendly interface and a range of tools to design and generate road systems with curves, intersections, and bridges, among others. The game world is instantiated with multiple cars and pedestrians. To make the cars and pedestrians interact consistently with the game world, WAY has developed its own road network software. This software ensures that pedestrians and cars respect the road boundaries, adhere to the speed limits, and comply with traffic rules.

2.3 Skill Model

To review the student's performance and monitor their progress, WAY uses a Bayesian network as a skill model. The skill network model employed at WAY adopts a Bayesian network framework to capture causal relationships between distinct driving skills. They use evidence obtained from the driver's performance on specific maneuvers during the driving session to update the network.

A Bayesian network, as a probabilistic graphical model, represents variables and their conditional dependencies utilizing a directed acyclic graph (DAG). The graph nodes correspond to the variables in the model, whereas the edges between them depict the relationships among these variables. Bayesian networks take advantage of Bayes' theorem as seen in 2.1 to model the probability of an event based on prior knowledge of conditions that might be associated with the event. By explicitly representing the conditional dependencies between variables, Bayesian networks enable probabilistic inference, which allows one to predict the likelihood of an outcome based on observed evidence.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2.1}$$

The applications of Bayesian networks span diverse fields such as artificial intelligence, machine learning, decision-making, and risk analysis [18]. They prove particularly valuable in scenarios characterized by uncertainty and incomplete information, as they can effectively model intricate relationships between variables and make probabilistic predictions based on incomplete data.

In this specific context, WAY works with two types of variables: directly measurable evidence, called performance variables, and latent skill mastery variables. The goal is to predict the probability distribution of the latent variables based on the observed evidence through the model's interdependencies.

Performance Variables

In the representation of evidence within the skill network model, they introduce two additional nodes: the measured performance node and the actual performance node. These nodes are interconnected through a conditional probability distribution (CPD), which is derived based on the accuracy of the assessment system employed. The probabilistic relationship can be seen in equation 2.2. When a driver executes a maneuver that directly corresponds to a skill in the skill network, a pair of measured and actual performance nodes is generated and labeled with the relevant timestamp. Both the measured and actual performance nodes encompass performance grade values ranging from A to F.

$$P(Grade_{measured}|Grade_{actual}) \tag{2.2}$$

The actual performance node serves as a direct descendant of the corresponding skill mastery node as described in equation 2.3. It reflects the actual level of performance achieved by the driver concerning the specific skill being assessed. Importantly, the CPD associated with each skill may exhibit variations due to factors such as the level of difficulty in mastering the skill and the tolerance for errors. Consequently, it becomes necessary to learn and infer this relationship from the available data.

$$P(Grade|Mastery) \tag{2.3}$$

Therefore, when evidence in the form of observed performance is acquired, its influence propagates upward in the skill network, subsequently affecting the prediction of the parent skill mastery node. This propagation of influence aids in refining the estimation of the driver's skill level based on the observed evidence.

Mastery Variables

Within the skill network, the skill mastery nodes are characterized by three potential states: mastered, learning, and struggling. These nodes have the flexibility to have an arbitrary number of parents and child skill mastery nodes, as well as associated evidence. Typically, skill mastery nodes remain unobserved, allowing probabilistic inference. However, in cases where a driving instructor manually overrides the skill mastery value, such as transitioning from the learning state to the mastered state, the respective skill mastery node becomes observed and no longer relies on the evidence for its determination. The CPD for parental-child skill mastery defines the interdependencies among the mastery nodes within the network described in equation 2.4. This CPD can be established through heuristic approaches or acquired through the analysis of a dataset generated by knowledgeable driving instructors.

$$\begin{aligned}&P(Mastery|Parent_1, Parent_2, ..., Parent_m),\\&where\ Parent_i \in \{Mastered, Learning, Struggeling\}\end{aligned} \tag{2.4}$$

Furthermore, skill progression, a quantitative measure representing the network's belief about the skill level, is calculated by equation 2.5. It is derived as a weighted average of the inferred beliefs associated with the skill mastery nodes. The skill progression considers the probabilities assigned to the different states of the skill mastery nodes and combines them accordingly.

$$\begin{aligned}&Progression =\\&0 \times P_{posterior}(Mastery = Struggeling) +\\&0.5 \times P_{posterior}(Mastery = Learning) +\\&1 \times P_{posterior}(Mastery = Mastered)\end{aligned} \tag{2.5}$$

Using skill mastery nodes and their associated CPDs, the skill network model provides a structured representation of skill states and their relationships. This framework facilitates the integration of expert knowledge and data-driven insights, enabling the estimation of skill levels and their progression over time.

Knowledge Transfer

In the absence of specific prior information, each driver is presumed to represent an average driver randomly selected from the broader population of drivers. The skill mastery nodes, being integral components of the skill network, possess prior values based on the Parent-Children CPDs.

However, following the initial driving session, the system begins to acquire valuable information regarding the driver's skill levels in the form of posterior probabilities. These posterior probabilities serve as transferred knowledge within the skill network model and are incorporated accordingly during the inference process after subsequent driving sessions.

By utilizing these posterior probabilities as a form of acquired knowledge, the skill network model effectively integrates the new information obtained from each driving session, enhancing the accuracy of subsequent skill level predictions and providing a more refined understanding of the driver's proficiency.

Chapter 3

Background Theory

This chapter builds upon the project thesis, and it aims to provide the necessary background knowledge for utilizing procedural content generation to create personalized traffic situations based on a student's skill level. Section 3.1 introduces *Serious games* and the design principles that should be adhered to when developing serious. Section 3.2 explores the concept of player modeling, which involves capturing and analyzing player behavior and characteristics to personalize the game experience. Section 3.3 introduces the term *Procedural Content Generation.* Section 3.4 presents a taxonomy for procedural content generation. This taxonomy serves as a foundation for understanding the subsequent discussions on PCG. Section 3.5 introduces the core components of experience-driven procedural content generation. This approach emphasizes generating content that considers the player's experience and skill level, aligning with the objective of creating personalized traffic situations based on student capabilities. This knowledge forms the basis for the following discussions and implementation of personalized traffic situations in the context of student skill levels.

3.1 Learning in Serious Games

Serious Games are an increasingly popular way to learn new material and are used in various areas such as healthcare, education, business, and marketing [20]. Michael and Chen describes Serious Games as games where the primary objective is something other than providing entertainment and enjoyment to the players. With the combination of learning strategies and game elements, serious games aim to be more engaging and motivating than regular learning techniques. When creating serious games where the main purpose is learning, the creators want to optimize the learning rate for every user. Greitzer et al. propose five design guideline principles to design e-Learning and training applications:

1. *Stimulate semantic knowledge.* Facilitate learning by relating material to the learner's experiences and existing semantic knowledge structures.

2. *Manage the learner's cognitive load.* To avoid overloading the learner's cognitive load, organize the material into smaller chunks. Make the chunks cover material that is gradually more complex.

3. *Immerse the learner in problem-centered activities.* The learner should immediately be provided with opportunities to work on meaningful and realistic tasks.

4. *Emphasize interactive experiences.* Activities should be developed that require the manipulation of objects to be solved. This encourages the active construction and processing of training material, which, in turn, helps build lasting memories.

5. *Engage the learner.* Design learning scenarios that keep learners within a narrow range of difficulty, where the material is challenging but not overwhelming.

Constructing coherent knowledge structures is essential for optimizing the storage, retrieval, and representation of information [22]. It is crucial to organize and categorize the information based on existing knowledge. By connecting new information to individuals' own experiences and preexisting semantic knowledge structures, the retrieval of such information can be enhanced.

Managing the learner's cognitive load is a key concept within learning. Human memory is divided into working memory and long-term memory, and while long-term memory is effectively unlimited in size, working memory is very limited [28]. Therefore, it is important that learners are given time to digest and reflect on newly learned information to transform knowledge from working to long-term memory. Learning should be centered around learning through interactive experiences and not through techniques such as rote learning.

When learners are engaged in problem-solving activities instead of passively digesting course content, learners are compelled to think about, organize, and use the information in ways that facilitate the active construction of meaning and help build lasting memories. This form of learning also increases motivation.

Baecker and Buxton presents an interesting paradox about learning where it is explained that to learn, users must have meaningful interaction with the system but are required a certain level of knowledge and experience before they are able to interact with the system. Research aimed at addressing this paradox proposes that a viable solution to deal with this paradox is to encourage learners to immediately work on realistic and meaningful tasks. Unlike other passive activities, the learner should immediately apply prior knowledge to problems [2].

According to Greitzer et al., the fifth principle suggests the design of learning scenarios that keep learners in a "narrow zone" of performance difficulty. The goal is to create training situations that challenge students, ensuring their engagement and motivation while avoiding scenarios that are too easy or excessively difficult. Maintaining this balance is crucial for optimizing learning outcomes and preventing frustration or disinterest among learners. Figure 3.1 visually represents this concept, emphasizing the importance of finding the sweet spot where learners are appropriately challenged without being overwhelmed or bored. By aligning training scenarios with the learner's capabilities and providing an optimal level of difficulty, educators can promote effective learning experiences that enhance student performance and motivation.

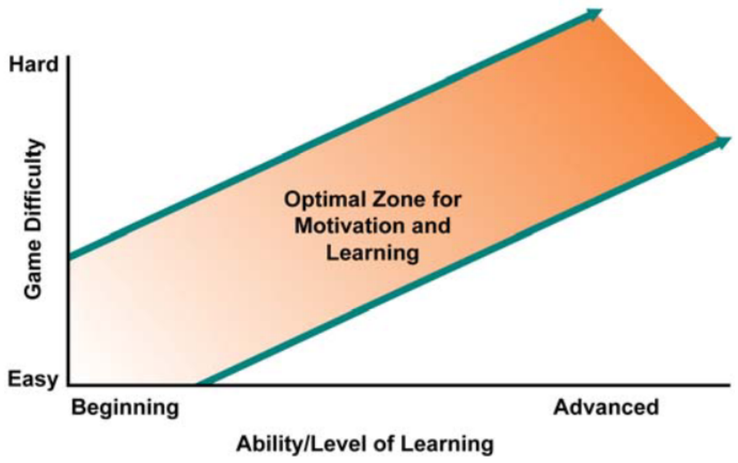

Hard
Game Difficulty
Easy
Optimal Zone for Motivation and Learning
Beginning
Advanced
Ability/Level of Learning

Figure 3.1: Difficulty related to Level of Learning
[12]

3.2 Player Modeling

In section 3.1, the significance of balancing game difficulty is emphasized, aiming for a challenging experience without crossing the threshold of being excessively difficult. To address this, games often allow players to set the difficulty level themselves. By letting the player profile themselves, game developers hope that the player knows themselves enough to set the correct difficulty and that the pre-made discrete-level alternatives correspond to all player types. This can be a problem for players who miss the insight to rank themselves correctly or who fall between the discrete alternatives. This problem is more prevalent in serious games [24]. Players with varying backgrounds, skill levels, and cognitive abilities experience different learning outputs from different inputs, and may therefore want to use alternative approaches to learn a skill. When employing the conventional approach of developing serious games with static content, it often becomes impossible to implement personalized approaches. To address these shortcomings, serious games must become more dynamic and player-centric [13].

If Serious Games incorporates the dynamic adjustment of game elements based on individual player performance, it can enhance the sense of personalization and uniqueness in the game, ultimately fostering an enhanced learning experience. To accomplish this, serious games require a mechanism to construct player models that accurately reflect their skills and experiences. When creating player models, the objective is either to classify players into predefined profiles or assign them a vector representation that represents their skills and experiences. For player classification tasks, unsupervised learning approaches such as clustering can be employed to group players [6]. The concept of Player Experience Modeling, introduced by Julian [19], presents a framework to develop such models, which will be further explored in section 3.4.

3.3 Procedural content generation

Procedural content generation (PCG) is the algorithmic creation of game content with limited or indirect user input [34]. In this context, game content encompasses all elements of a game that directly influence gameplay, excluding non-player character behavior and the game engine itself [19]. Using PCG, game developers can provide content that is adjusted to individual players based on a variety of parameters. PCG has been employed in games since the late 1970s. Noteworthy early titles like Elite and Rogue displayed some of the first applications. In the case of Rogue, it showcased one of the initial instances of using PCG to generate levels randomly. Following these ideas, *rogue-like* became an established genre with household names such as Diablo and Angband basing their PCG on this genre [42]. Elite also implemented an early-stage PCG, but contrary to the modern interpretation of the concept, Elite's PCG was fully deterministic and used as a form of data compression [36]. PCG has, however, advanced considerably since its introduction into games. Since the aforementioned games implemented simple algorithms to improve the game quality of 2D environments, many different artificial theories have been applied to the field. Ranging from machine learning to evolutionary algorithms, most have proved their worth on different occasions. Chapter 4 will elaborate further on several of these applications.

3.4 Search-Based Procedural Content Generation

Julian Togelius, Georgios N. Yannakakis, Kenneth O. Stanley, and Cameron Browne introduce in *Search-Based Procedural Content Generation: A Taxonomy and Survey* [39] a taxonomy of approaches and variations of PCG. These distinctions are not always binary but can exist on a continuous spectrum, placing PCG instances closer to one extreme or the other. The five distinctions identified are as follows:

1. *Online Versus Offline*

Online content generation refers to the process of creating game content dynamically during gameplay, responding to player actions, real-time events, or procedural algorithms. It enables dynamic and adaptive experiences. Offline generation, in contrast, creates predetermined game content before the game is initiated, remaining static throughout the session.

2. *Necessary Versus Optional Content*

Necessary content is game content essential for progression and the player is therefore obligated to interact with the content, whereas optional content can be avoided. The correctness of necessary content is crucial, while the generation of optional content comes with milder demands.

3. *Random Seeds Versus Parameter Vectors*

This distinction refers to the extent of control that developers have during content generation. PCG instances exist on a continuum ranging from random seeds to parameter vectors. On the one end, content can be generated completely randomly with random seeds. In contrast, it can be finely controlled through various parameter vectors.

4. *Stochastic Versus Deterministic Generation*

Stochastic generation introduces variation and randomness to the outcome, even with identical input parameters. Deterministic generation, on the other hand, ensures consistent and predictable results with no variation. The choice between the two depends on the desired level of variability and predictability in the generated content.

5. *Constructive Versus Generate-and-Test*

A constructive generative algorithm guarantees the correctness of the generated content by generating it based on specific rules and hard constraints. In contrast, a generate-and-test algorithm creates content first and then tests its validity. If the content fails the test, new content is generated and tested again until a satisfactory result is obtained.

In addition to this taxonomy, *Search-Based Procedural Content Generation: A Taxonomy and Survey* [39] also introduces the term Search-Based Procedural Content Generation (SBPCG). SBPCG is a special case of the generate-and-test approach to PCG, where the algorithm explores a population of candidate content. SBPCG has two key qualifications that differ from traditional generate-and-test methods. Instead of simply rejecting or accepting candidate content, the search-based test function grades the content with a quantifiable and comparable value. This test is often referred to as the fitness function or the utility function, and the assigned value is often referred to as the fitness of the content. The second key qualification is that creation of new candidate content is dependent on the fitness of previously evaluated content instances. The objective is to generate new content that surpasses the fitness value of the previous instances, thereby achieving continuous improvement. Figure 3.2 presents an overview of the approaches of the three different methods.

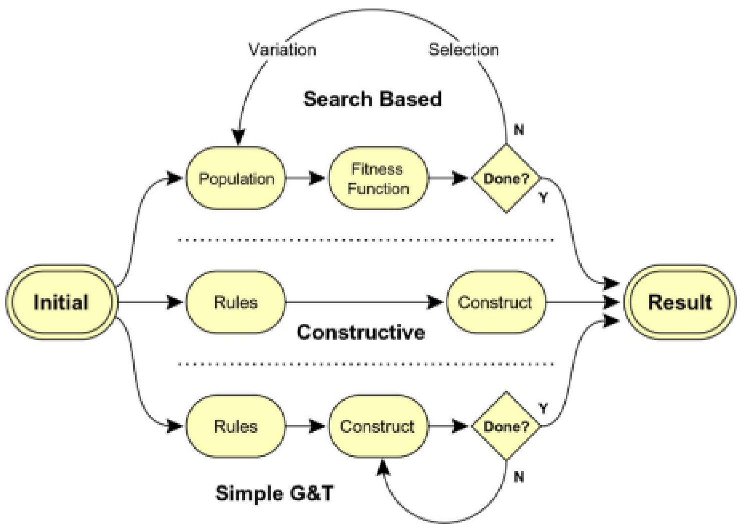


Figure 3.2: Procedural Content Generation Methods

3.5 Experience Driven Content Generation

To facilitate the optimization of engagement and cognitive learning in serious games Julian introduces the concept of *Experience-Driven Procedural Content Generation* (EDPCG), consisting of four critical components.

Player experience modeling

The first component is *Player experience modeling* (PEM). This relates to how the player experience should be modeled as a function of game content and player info. A player is characterized by his playing style and his responses to events in the game. PEM is broken further down into three sub-types, subjective PEM, objective PEM, and gameplay-based PEM. Subjective PEM refers to data explicitly expressed by players. Most of the time, this information is retrieved using something that resembles a questionnaire. Objective PEM gathers information about the players indirectly from the player's expressed emotions and infers their respective states of mind. Lastly, gameplay-based PEM relies on the interaction between the player and the game for its data retrieval.

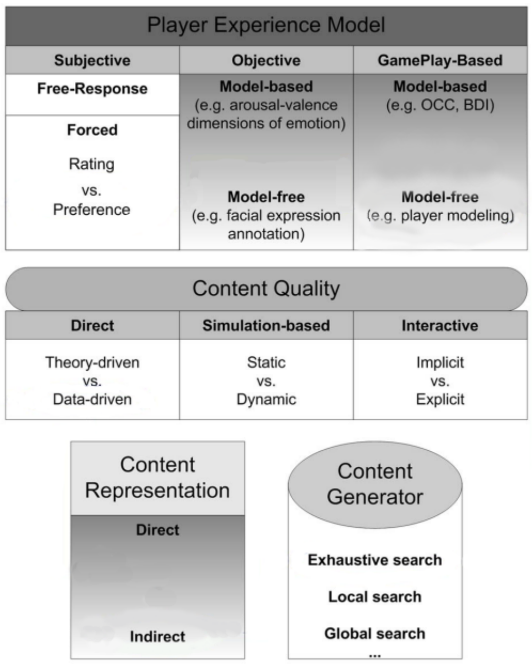


Figure 3.3: Overview of the PCG framework
[19]

Content quality

The next component introduced is *Content quality.* To be able to generate high-quality game content, it is crucial that the system can evaluate different pieces of game content. It is common to use an evaluation function that evaluates an item of game content and assigns it a scalar that reflects its suitability. To evaluate the game content, the system designers first decide what properties they want to optimize. A designer may want to design a game that is fun, engaging, and replayable. The goal of the designer is then to create an evaluation function that reflects how much a piece of game content can contribute to those goals. Togelius et al. [39] introduces three classes of evaluation functions: *direct, simulation-based,* and *interactive.*

The direct evaluation function extracts a given set of features from the game content and directly maps this to a quality value of the content. The mapping can be both linear and non-linear and does not involve large amounts of computation. Direct evaluation functions can be further broken down into *theory-driven* and *data-driven* functions. The theory-driven functions are based on the creator's intuition or theory about the player experience. Then a mapping is created between the game content and the experience model. An example of an approach that uses a theory-driven evaluation function can be seen in [9]. Data-driven evaluation functions are based on collecting data from the results that come from different types of content. Data is typically extracted through a questionnaire or physiological measurements. The data is then used to automatically create mappings between player experience and game content. An example of an approach that uses a data-driven evaluation function can be found in [26].

In a simulation-based evaluation function, an artificial agent plays through the generated content. Features such as whether or not the agent won and how fast it won are extracted and evaluated. This can be further broken down into static and dynamic simulation-based functions, where the agent in a dynamic simulation changes during the game. In contrast, the agent in the static simulation remains static. In an interactive evaluation function, the evaluation function evaluates the actual gameplay of a player.

Content representation

Furthermore, there is the concept of *Content representation*, which addresses how the world should be represented to facilitate the optimization of the content generator. Content representation can often be placed on a spectrum between direct and indirect encodings. This corresponds to the concepts of *Random Seeds* versus *Parameter Vectors* introduced in section 3.5 where the random seed is an indirect encoding and parameter vectors are a direct encoding. A direct encoding can be a 2D grid where the content of every cell is represented. This causes a direct one-to-one mapping between the representation and the actual game content. An indirect encoding can be a specification of certain desired properties, such as the number of enemies, the number of gaps, or the distance between platforms. These properties dictate how the game world is generated. A principle within content representation, and especially content representation related to evolutionary computation, is *locality*. High locality means that a small change in the content representation results in little change in the utility value and is more easily achieved with a more direct approach to content representation.

Content generation

The final key component in the PCG framework is *Content generation*. Utilizing the player model, content representation, and an evaluation function to assess the content, the actual content can now be generated. When working with Search-based generation [39], the content generator uses an optimization function to search through the content space and approximate the optimal solution. Different optimization functions can be applied based on the size of the search space. With a small number of dimensions, an exhaustive search may be sufficient to provide a robust solution to online PCG. With larger search spaces, techniques such as simple heuristic and gradient-search algorithms and stochastic global optimization techniques such as evolutionary algorithms and particle swarm optimization may be used. When working with constructive and generate-and-test approaches to content generation, searching through content space is not necessary. With these techniques, the generators follow a set of constraints that make sure that the generated content is optimal.

Chapter 4

Related Work

This chapter provides an overview of the related work in the field while exploring relevant methods and techniques that can be applied to generate personalized driving lessons procedurally. The literature presented in this chapter is a combination of papers discovered in the project thesis and additional papers explored during the spring semester. The chapter presents the methodology employed for the literature review in this study, followed by an exploration of various applications of Procedural Content Generation in serious games and other environments. Additionally, the chapter discusses relevant literature concerning personalized learning. To conclude, a summary of key findings is presented before our contribution is positioned within the broader context of the field of study. Table 4.2 presents a comprehensive comparison of the reviewed literature on procedural content generation, while table 4.3 offers a similar comparison for the literature related to personalized learning.

4.1 Method

The challenge of developing personalized lessons in a virtual simulator is closely related to both procedural content generation, as discussed in section 3.3, and personalized learning. These connections served as the basis for the literature search conducted in this study. The search strategy consisted of a database search with the key terms seen in table 4.1, followed by subsequent snowball and citation search from relevant articles [40]. The search was performed using Scopus and Google Scholar as the primary search engines. The identified articles were then managed and organized using the Mendeley reference management tool, allowing for efficient storage and labeling of the retrieved literature.

Search Terms
Procedural Content Generation
Virtual simulators
Personalized learning
Traffic scenarios
Serious games
Interactive virtual worlds
Skill enhancement
Intelligent Tutoring Systems

Table 4.1: Search Terms

4.2 Literature

This section comprises a review of various applications of PCG in diverse domains and a set of papers relating to personalized learning.

4.2.1 Procedural Content Generation

The papers concerning PCG are discussed below and compared in table 4.2, where the distinctions relate to the key terms introduced in sections 3.1, 3.5, and 3.4.

Answer Set Programming for Declarative Content Specification: A Scalable Partitioning-Based Approach

Calimeri et al. [5] investigate partition-based generation techniques with Answer Set Programming (ASP). They propose a multi-step generation algorithm to generate 2-D caves in unity. Utilizing the declarative nature of ASP, they achieve results that are better scaleable to larger maps than search-based approaches. The algorithm recursively generates area partitions that obey the ASP rules, resulting in a satisfying map. Although their work demonstrated the potential of procedurally generating a functional game world based on semantics and simple rules, it did not address the crucial aspects of personalization and difficulty adaptation central to our research objective.

Towards Automatic Personalised content creation for racing games

Togelius et al. [38] review the potential of procedurally generating both player models and personalized race tracks in *Towards Automatic Personalised content creation for racing games* [38]. They employ the cascading elitism algorithm, a multi-objective artificial evolution technique, for both player modeling and track generation. The player modeling was optimized on three fitness objectives, and the results showed promise in resembling human behavior. They then proceeded to generate personalized race tracks for the player models. To tailor the tracks to each individual, they break down the concept of "fun" during gameplay into discrete elements. This approach led to the definition of three distinct fitness objectives, which they then optimized for using the cascading elitism algorithm. The results showed promise in generating personalized tracks and, thus, transferability to our domain. However, it has fewer requirements and a much simpler 2-D environment.

SceGene: Bio-Inspired Traffic Scenario Generation for Autonomous Driving Testing

Li et al. [21] argues in *SceGene: Bio-Inspired Traffic Scenario Generation for Autonomous Driving Testing* that comprehensive tests are necessary to discover potential vulnerabilities in autonomous driving systems. Such extensive testing in the real world requires several years, whereas the efficiency could be improved dramatically in a simulation-based environment. To generate enough scenarios for training, Li et al. introduce SceGene, a biologically inspired search-based algorithm for traffic scenario generation [21]. The algorithm encodes traffic scenarios as genotypes and optimizes them through genetic operations on a set of requirements. They achieve promising results, successfully generating diverse scenarios suitable for training autonomous systems. The method shows similarities to our case in generating simulator-based traffic scenarios. Still, it differs in focusing on generating a diverse range of scenarios suited for training compared to the personalization demands that come with our objective.

Online level generation in Super Mario Bros via learning constructive primitives

Shi and Chen [35] propose to generate content procedurally utilizing a hybrid between generate-and-test and constructive methods to create levels in Super Mario Bros. They introduce the notion of constructive primitives (CP's), which in essence, are predefined quality building blocks for the game environment. To learn these CP's through active learning, they train a binary classifier in the form of a weighted random forest (WRF). This WRF is then used to produce the CP's through a generate-and-test method. Once the CP's quality is satisfactory, the complete procedural levels can be constructively generated online by sequences of CP's with a simple algorithm. The constructive algorithm works by receiving some specified values for controllable parameters, which in turn determine the values of important content features. These content features impose some restrictions to which sequences of CP's are valid and ensure that the resulting game levels are satisfactory. Shi and Chen successfully managed to showcase the potential of combining predefined quality building blocks into an acceptable game world. When compared to our objective, the method has some notable shortcomings, but the core idea of sequencing building blocks might be applicable.

Integrated Approach to Personalized Procedural Map Generation Using Evolutionary Algorithms

Raffe et al. [32] investigates experience-driven procedural content generation of maps in a unity-based 3D action-shooter game with evolutionary algorithms. The paper decomposes the optimization process of maps into two integrated searches. The two processes are geometry optimization and content density optimization. The goal of the game is to get from one end of a given map to the other without succumbing to enemies scattered around the map. When generating the geometry of a map, they represent it as an n-ary tree structure, where each node represents a room and each edge a corridor between rooms. In the geometry optimization process, they utilize a genetic algorithm. Once the geometry scores sufficiently well, they proceed to optimize content density. The possible content is divided into sub-types, and each room has discretized properties for each sub-type that can be set to either none, low, medium, or high. A compositional pattern-producing network then calculates the content. Furthermore, they frame player modeling as a content-based recommender system, where a classifier is trained on evaluations given by the player previously and subsequently used to present the player with the most suited game levels. The method described shows similarities to our domain in terms of creating levels tailored to individuals but differs in the requirements for personalization. They also determine the density of content through discretized properties, which could be relevant to our objective.

Toward supporting stories with procedurally generated game worlds

Hartsook et al. [14] propose a technique for automatically generating fully playable computer role-playing games. To achieve the desired result, they introduce the concepts of *islands* and *bridges*. Islands are areas in which meaningful actions take place, whereas bridges are essentially transport stages where more or less random incidents occur. They utilize an offline search-based approach and employ a genetic algorithm to build environments consisting of a set of islands and bridges. The fitness function is an evaluation criterion that reviews both the players' preferences through subjective PEM, as well as the realism decided by a transition graph. The transition graph states how likely it is that two subtypes of islands and bridges are adjacent. The approach showcases how search-based models can optimize for personal preferences and presents the notion of islands and bridges. These concepts might have relevance to our objective, particularly addressing research question one and research question two.

Using gameplay semantics to procedurally generate player-matching game worlds

Lopes et al. [14] presents a semantic generation framework for creating player-matching game worlds in adaptive games [25]. The framework integrates behavior and experience modeling, content correlation, and procedural content generation. The chosen game for implementation is *Stunt Playground*, a sandbox game where players can perform stunts in an arena. The behavior modeling captures the player's style using heuristics such as distance in the air, time in the air, average speed, and number of flips. The experience modeling focuses on maximizing the fun factor, considering heuristics like initial fun value, difference between behavior scales, respawns, and time stopped. Content correlation tracks the relationship between gameplay experience and observed content. The gameplay semantics, defined in the semantic library, allow designers to specify the gameplay value of different entities. The retrieval process matches player behavior and experience with semantic gameplay descriptions to generate player-matching content. The semantic layout solver is employed for post-retrieval content generation, using placement rules and object features to determine valid locations for entities in the game world. The integration of the semantic generation framework with *Stunt Playground* demonstrates the practical application of adaptive gameplay and provides insights into the potential of adapting game worlds based on player behavior and experience. Lopes et al. introduces a novel approach to experience-driven content generation but differs from the objective stated earlier in this thesis by focusing on fun instead of learning when generating the game world.

A semantic generation framework for enabling adaptive game worlds

In *A semantic generation framework for enabling adaptive game worlds*, Lopes and Bidarra introduces a framework for adaptive game worlds based on Player and Experience Models. Lopes and Bidarra defined adaptive games as games that possess the ability to perceive and understand how players interact with them, enabling them to dynamically adjust and adapt to cater to the individual in-game requirements and objectives of these players, thus enhancing the overall gameplay experience. The framework introduced can be seen in figure 4.1. It consists of 5 components, Player and Experience Models, Game Observer, Content Utility Model, Generator, and a Semantic library [23].

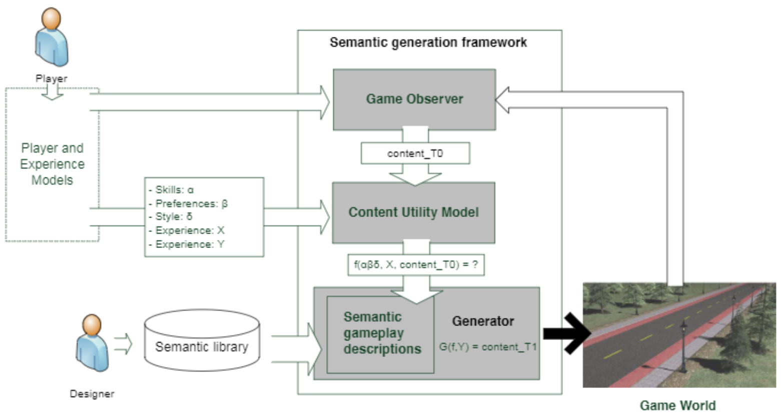


Figure 4.1: Generation framework
[23]

The paper describes two simulated scenarios: a First Person Shooter game and a driving simulator. The scenario with the driving simulator represents a serious game where a student drives around in a game world and executes instructions from a driving instructor, where instructions are chosen based on the student's skill level. The framework's application in this scenario touches upon the two research questions, **RQ1** and **RQ3**, but does not touch upon research question **RQ2**.

A semantic approach to patch-based procedural generation of urban road networks

Teng and Bidarra [37] propose a procedural method for the generation of urban road networks using patch-based techniques and geometric graphs. Their model segregates road networks into *main* and *local streets*, each fulfilling different roles. A parametric graph-growing algorithm is employed for main-street generation based on various parameters. The local-street generation phase involves connecting initial patches to main streets and expanding the network iteratively. A queue of potential vertices is maintained, and feasible patches are selected and appended based on constraints. The selection process considers user-defined parameters and the propagation direction. The method is highly dependent on the semantics of patches described in the paper. Semantics is used to represent the features needed for controllability of the road generation, and are classified into vertex, edge, and patch categories. Patch examples are shown in figure 4.2. This paper may provide helpful insight to answer research question R1 introduced in section 1.2 but does not introduce player modeling or personalization needed to satisfy research questions R2 and R3.

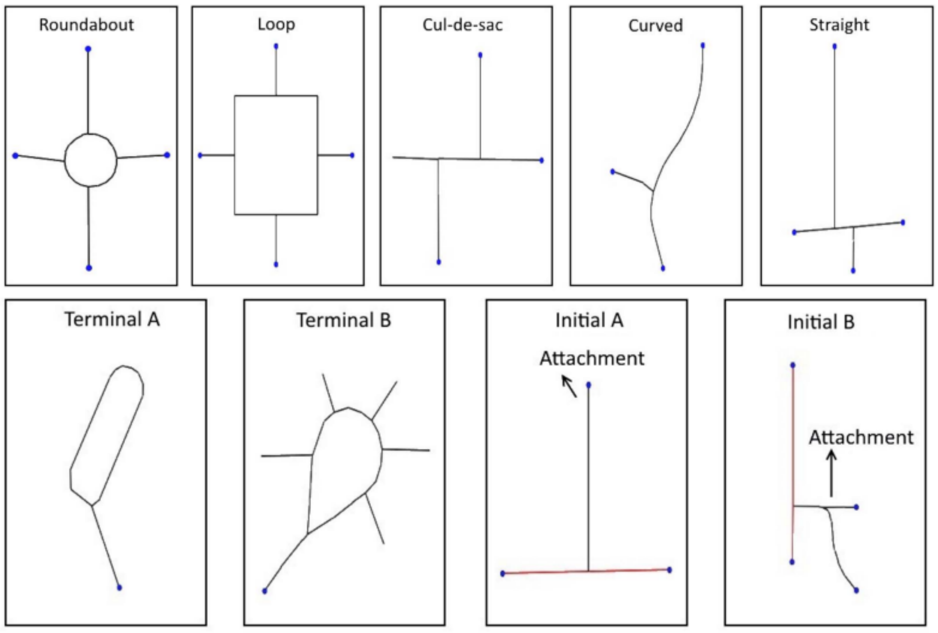


Figure 4.2: Example of patches
[37]

Challenge Balancing for Personalized Game Spaces

In *Challenge Balancing for Personalized Game Spaces* Bakkes et al., explore the use of AI to personalize game environments for increased player participation and enjoyment [3]. It focuses on personalizing the game space for individual players with regard to experienced challenges, which aligns with the concept of experience-driven procedural content generation (EDPCG). Three aspects of this personalization include challenge balancing, player modeling, and space adaptation. Challenge balancing refers to adapting the game's challenge level to a player's skills, the term player modeling refers to establishing models of player behavior, and space adaptation refers to the modification of the game space in response to the user's experience. The researchers apply these concepts to an enhanced version of the game *Infinite Mario Bros*, focusing on procedural content generation to optimize the player's individual experience in real-time gameplay. This research proposes a method to personalize the difficulty of video games based on implicit feedback from players without disrupting the gameplay. The process is divided into three phases: (1) learning a global safe policy offline, where the model labels gameplay sessions with human participants to establish an estimate of the player's preferred challenge level, (2) learning a feedback model offline, where a random forest decision tree classifier is used to map gameplay observations to estimates of the player experience, and (3) online personalization during gameplay, which uses the previously learned feedback model to adjust the game's difficulty based on the player's interactions. A user abandonment model is also integrated to enhance state-space exploration, enabling the system to rapidly adapt to individual players' preferences. Bakkes et al. introduces an approach to challenge balancing that is relevant to research question RQ3. However, it does not explicitly discuss learning in serious games or the semantic representation of content to answer research questions RQ1 or RQ2.

Scenario generation for emergency rescue training games

Hullett and Mateas presented a generator for constructing emergency rescue scenarios with the use of PCG [16]. They envisioned a serious game where rescue workers could train on realistic scenarios to evaluate hazards, determine the correct approach to the situation, and identify potential areas victims could be trapped. To increase the replayability of their scenarios, they looked at methods to dynamically generate scenarios with a high degree of internal consistency that provided training to satisfy given pedagogical goals. Paramount for the generation was that the PCG needed to produce content that seemed realistic in terms of appearing like a believable rescue scenario while still maintaining some degree of randomness. They proposed a solution that was built upon a hierarchical task network and qualitative physical reasoning. The system was provided a representation of an uncollapsed structure before applying the qualitative physical reasoning to end up with a fully collapsed rescue scenario. This paper presents an application of procedural content generation in a serious game but lacks the individualization required for our subjective.

Towards player adaptivity in a serious game for conflict resolution

Grappiolo et al. [11] explain how they created a serious mini-game for conflict resolution using EDPCG. The goal of the game is to avoid and resolve conflicts between NPCs to aid children in acquiring the soft skill of resolving conflicts. This is done by distributing resources between the various groups of NPC's. The mini-game relies on EDPCG and employs a search-based approach to content generation. More specifically, they use a genetic algorithm to generate subsequent levels with different difficulties. Whenever a level is completed, the model assesses the skill level of the student and optimizes the new level with a personalized fitness function. The fitness function is based on an Artificial Neural Network that attempts to approximate the unknown function between game-level elements and player cooperation. There exist several similarities between this method and the requirements of our objective. They employ PCG in a serious game with individualization and difficulty adjustment. However, due to the lack of training data, using an ANN would not be a feasible solution to research our objective.

Particle Swarm Optimization for procedural content generation in an endless platform game

Grappiolo et al., Hartsook et al., Raffe et al. utilize different variants of (Genetic Algorithms) GA's for PCG, but Pontes et al. argue in *Particle Swarm Optimization for procedural content generation in an endless platform game* [31] that another search-based algorithm, namely Particle Swarm Optimization (PSO) can outperform GA's in terms of both computational demands and performance in certain domains. In this case, the domain is a simple endless 2D platform game. The goal of the game is to avoid death by falling into holes or landing on spikes situated on the map, as well as gathering clock items to keep the clock from running down. The fitness for each particle was calculated by adding a penalty for each part of the course that was impossible with the game mechanics, as well as measuring how close it was to the desired difficulty level. They also showed that the algorithm ran fast enough to generate new content online, making the game never-ending. The attribute of adapting content to the desired difficulty and the possibility of generating content efficiently enough to make the algorithm run online are both interesting discoveries presented by this paper. The environment is, however, significantly less complex compared to ours, possibly limiting the transferability of the online attribute.

Table 4.2: Comparison of PCG applications

Title	**Serious Game**	**Player Model**	**Generation**	**Algorithm**	**Online/Offline**
Shi and Chen [35]	No	No	Constructive	Weighted Random Forests	Online
Hartsook et al. [14]	No	Subjective	Search-based	Genetic Algorithm	Offline
Raffe et al. [32]	No	Subjective, Recommender System	Search-based	Genetic Algorithm, Compositional Pattern-producing Network	Offline
Pontes et al. [31]	No	No	Search-based	Particle Swarm Optimization	Online
Hullett and Mateas [16]	Yes	No	Constructive	Hierarchical Task Network	Offline
Grappiolo et al. [11]	Yes	Gameplay-based	Search-based	Genetic Algorithm, Artificial Neural Network	Offline
Teng and Bidarra [37]	No	No	Constructive	Breadth-First Search	Offline
Calimeri et al. [5]	No	No	Constructive	Answer Set Programming	Offline
Togelius et al. [38]	No	Gameplay-based	Search-based	Cascading Elitism Algorithm	Offline
Li et al. [21]	No	No	Search-based	Genetic Algorithm	Offline
Lopes et al. [25]	No	Gameplay-based	Constructive	Semantic Layout Solver	Offline
Bakkes et al. [3]	No	Objective, Subjective	Search-based	Gradient Ascent	Online, Offline

4.2.2 Personalized Learning

This subsection comprises a discussion of papers related to personalized learning, and a comparison can be found in table 4.3. The distinctions used are *User Profile, Intelligent Tutoring System (ITS), Methods* and *Online Adaptation.* User profile relates to how the system updates and creates the student's information, ITS concerns whether the paper labels the system accordingly, whether methods describe the theoretical concepts utilized, and online adaptation refers to if the system facilitates updating the learning recommendations during run-time.

Adaptive Tutoring on a Virtual Reality Driving Simulator

Ropelato et al. [33] introduce an adaptive tutoring system within a virtual reality driving simulator. To create the game world, the system utilizes Unity as a 3D game engine and employs manual creation and CityEngine to generate realistic 3D content, including a city environment. It incorporates AI-controlled cars to simulate real traffic scenarios and applies an Intelligent Tutoring System (ITS). This tutoring system guides drivers through a sequence of activities tailored to their skill level, resulting in an interactive and effective training system for enhancing driving skills.

The ITS utilizes the concept of the Zone of Proximal Development (ZPD) to optimize the training sequence of various driving activities. These activities are organized based on exercise type and difficulty level and are used to continuously evaluate the driver's performance in different activities. The system selects the following activity to be trained based on the driver's performance. It provides directions to the next destination by finding the closest location where the chosen activity can be chosen by using Dijkstra's shortest path algorithm. By adapting the training sequence, the ITS ensures that the driver is consistently challenged at an appropriate level, maximizing their learning potential. The system incorporates predefined activities that are relevant to driving skills, such as stable driving on straight and curved roads, turning at junctions, complete stops, maintaining a constant speed, and reacting to unexpected situations where each activity is evaluated based on specific criteria. In contrast to the objective of this thesis, where the goal is to create an adaptive tutoring system by generating a personalized game world, the paper by Ropelato et al. focuses on developing an adaptive tutoring system that guides the student through a static game world. The emphasis lies in providing guidance and support within the existing game environment rather than dynamically generating personalized game worlds for individual students. Both approaches aim to enhance the learning experience, albeit with different strategies and objectives. In summary Ropelato et al. introduces an Intelligent Tutoring System that optimizes the training sequence of driving activities based on a student's skill level, which is relevant to the overall objective of this thesis.

Personalized e-learning system using Item Response Theory

Chen et al. introduces an architecture for a personalized e-Learning System based on Item Response Theory called PELIRT. The system consists of front-end and back-end components where the front-end is represented by an Interface Agent that handles communication with learners and records their behavior, while the back-end analyzes learner abilities and selects appropriate course materials based on estimated abilities. The system includes an interface agent for learner interaction, a personalized agent with feedback and course recommendation components, and databases for user accounts, user profiles, and course materials. The difficulty parameters of course materials are adjusted through a combination of expert judgment and collaborative voting by learners, and learner abilities are estimated using maximum likelihood estimation. The course recommendation agent recommends suitable course materials based on learner abilities and the information function, which is based on IRT. Overall, PELIRT offers a personalized learning experience by adapting the course materials to each learner's ability level. The approach introduced by Chen et al. implements an architecture for personalized learning that adjusts the difficulty of the content provided to the learner, which is relevant to the research questions **RQ1** and **RQ3**.

Constructing a personalized e-learning system based on genetic algorithm and case-based reasoning approach

Huang et al. introduces a novel approach to using mastery learning, genetic algorithms, and case-based reasoning to create a personalized e-learning system called PLS-ML. Mastery learning is an educational method that focuses on varying the learning material based on student ability and repeats learning material until the students have learned the material, which is referred to as reaching the mastery level. If learners fail to reach the mastery level in a unit, the system suggests personalized curriculum sequencing using a genetic algorithm (GA). If they achieve mastery, they move on to enrichment activities with additional topics. The results of each curriculum sequencing and formative assessment are stored in case-based reasoning (CBR). When learners pass the second formative assessment, they progress to the next unit. The estimation of curriculum difficulty parameters involves a curriculum modeling process where teachers analyze the primary concepts and design test items for each concept. Pre-tests are conducted with examinees, and item response theory and statistics-based BILOG programs are used to determine the difficulty parameters for the test items. The curriculum is designed based on the content of the test items, assuming that the difficulty of the curriculum corresponds to the difficulty of the test item, and a curriculum relation degree is calculated using the vector space model (VSM). Each curriculum is represented as a vector, and its relevance to user queries is measured through matching functions. Overall, the PLS-ML system aims to provide personalized curriculum sequencing and materials to learners based on their individual requirements, which could be transferable to personalizing driving lessons.

A Heuristic Algorithm for planning personalized learning paths for context-aware ubiquitous learning

Hwang et al. [17] argue in *A Heuristic Algorithm for planning personalized learning paths for context-aware ubiquitous learning* that students might fail to acquire the intended amount of knowledge in a conventional authentic learning environment due to a lack of attention, quality information, and personalization. They propose a context-aware ubiquitous learning (u-learning) environment to circumvent this unfortunate consequence. Such an environment refers to a learning environment that leverages technology to provide educational experiences that are personalized and adaptive. Hwang et al. highlights two key challenges that need to be addressed to optimize learning. The first relates to finding an ideal order to learn the objectives, hereby facilitating the student to understand connections in the field of study. The next challenge addresses the fact that learning quality drops significantly when too many people are educated simultaneously in real-world environments. In this particular case, they propose a heuristic algorithm for determining a personalized learning path taking both challenges into account. This process comprises two key considerations. They start with calculating the relevance between each pair of learning objectives and proceed to find an optimal path through the learning environment framing it as a compound traveling salesman problem considering both objective relevance and objective crowding. When testing the implementation in real-world environments, the initial experimental results showed good promise in motivation, interactivity, and effectiveness. Related to our research objective, this paper addresses the customization demands, especially in line with **RQ1**, which examines the most effective approaches to maximize learning outcomes. In summation, while the crowding issue described here may not be significant in a virtual environment, the relevance of the objectives might have transferable implications.

Table 4.3: Comparison of Personalized learning

Title	**User Profile**	**ITS**	**Methods**	**Online Adaptation**
Ropelato et al. [35]	Gameplay Feedback	Yes	Djikstra's, Zone of Proximal Development	Yes
Chen et al. [7]	Gameplay Feedback	Yes	Item Response Theory	Yes
Huang et al. [15]	Gameplay Feedback	Yes	Item Response Theory, Genetic Algorithm, Case-based Reasoning	Yes
Hwang et al. [17]	Questionnaire	No	Heuristic Object Relevance, Traveling Salesman	Yes

4.3 Summary

The literature presented in this chapter covers various studies and approaches related to procedural content generation (PCG) and personalized learning in multiple different contexts. Concerning PCG, the reviewed articles highlight different algorithms and frameworks for generating game content. The methods introduced cover various approaches relating to the taxonomy from section 3.4 and present applications within the context of serious games and personalization. Regarding the literature on personalized learning, the focus is on optimizing individual learning outcomes through adaptation to individual needs and preferences. The articles illustrate different methods for enhanced learning outcomes and optimal guidance. In the two lists below, all of the key findings are highlighted.

Procedural Content Generation

Semantics Calimeri et al. [5], Lopes et al. [25], Lopes and Bidarra [23] and Teng and Bidarra [37] all demonstrate the potential of using semantic rules and relationships to generate content successfully.

Personalized EA Togelius et al. [38], Hartsook et al. [14] and Raffe et al. [32] employ various search-based evolutionary algorithms to generate personalized content for a given individual procedurally.

Traffic Scenario GA Li et al. [21] establishes the capability of using a genetic algorithm to generate traffic scenarios by encoding the traffic scenarios as a genotype.

Quality Building Blocks Shi and Chen [35] and Teng and Bidarra [37] introduces the notion of quality building blocks. Even though they are created differently, they both prove the value of utilizing these in content generation.

Semantics and Personalization Lopes et al. [25] and Lopes and Bidarra [23] demonstrate two possible methods to employ semantics for tailoring content to player models.

Serious Games Hullett and Mateas [16] and Grappiolo et al. [11] showcase two applications of PCG to serious games with success.

Personalized Learning

Dijkstra's Optimal Path Ropelato et al. [33] presents a method for finding an optimal route with respect to learning outcome through the environment utilizing Dijkstra's algorithm for the shortest path.

Item Response Theory Chen et al. [7] and Huang et al. [15] introduce the idea of Item Response Theory for deciding which skill to train next.

Objective Relevance Hwang et al. [17] discusses the importance of ordering learning objectives and proposes a heuristic approach to solve this.

Overall, the existing literature forms a strong foundation by exploring various approaches that offer valuable insights into the procedural generation and customization of game content. When comparing these studies to our research objective of examining the feasibility of utilizing WAY's Unity-based simulator to automatically generate personalized virtual driving lessons based on individual skill levels, we find that they touch upon certain aspects of our objective. However, none of them specifically address both the generation and optimization of content to enhance learning outcomes. Furthermore, it is worth noting that there is no such previous research conducted in collaboration with WAY, which adds to the novelty of our work. Thus, this literature review serves the dual purpose of providing an overview of the field and stating the novelty of our research.

Chapter 5

Model

This chapter provides an overview of the system requirements and introduces the architecture for our proposed system. The architecture is divided into four components: Player Experience Model, Content Quality, Content Representation, and Content Generation. Building upon the concepts established in earlier chapters, the chapter focuses on developing a system capable of addressing the research questions outlined in section 1.2. Additionally, the chapter delves into calculating an optimal route through the generated game world. Overall, this chapter lays the groundwork for the implementation and functionality of the system.

5.1 Requirements

A dynamic representation of the world and its building blocks is necessary to generate personalized traffic situations and subsequently answer the research questions defined in section 1.2. When designing an architecture for this purpose, we have identified six requirements that must be fulfilled.

R1 Each skill trained by the skill model must be effectively trained by at least one predefined traffic situation within the system.

R2 The system should be capable of accurately representing all traffic situations.

R3 The system should dynamically generate traffic situations based on the output of the skill model, adapting to the driver's proficiency

R4 The system should ensure that the student encounters the desirable traffic situations in an order that ensures optimized learning.

R5 The system should be able to adjust its content in real-time to adapt to the skill level of the student.

R6 The difficulty level of each traffic situation should be adjustable to align with the driver's skill level, ensuring an appropriate level of challenge.

These system requirements provide a solid foundation for building a model capable of answering our defined research questions. More specifically, system requirements **R1** and **R2** relate to **RQ2**, **R3**, **R4** and **R5** relate to **RQ1** and the remaining system requirement **R6** is related to **RQ3**.

5.2 Architecture

To design a system that can meet the requirements introduced in the previous section, the architecture that is proposed in this thesis is based on the incorporation of the four components from Experience Driven Content Generation introduced in section 3.5, namely Player Experience Model, Content Quality, Content Representation, and Content Generation. Along with these components, we aim to follow the five design principles introduced in section 3.1 to maximize student learning. The system is required to be implemented in WAY's virtual simulator. Considering the taxonomy presented in section 3.4, our proposed solution addresses the task of generating traffic situations for driving students using *offline*, *constructive*, and *stochastic* content generation. Offline generation, in this context, refers to the process of generating an entire lesson before the student starts the simulator. The characteristics of the Skill Model determine the use of offline generation, in contrast to online generation. As mentioned in section 2.1, the skill level inference by the Skill Model is done after a lesson ends and not in real-time. Therefore, the most relevant generation type is *offline* generation. To promote variation, the system will use *stochastic* generation. By incorporating randomness, it introduces diversity and unpredictability into the generated outputs. This variability allows for the exploration of different possibilities and the generation of unique, customized results. The system will generate *necessary* content that is essential for the simulation, focusing on elements directly relevant to the objectives and challenges. An overall architecture of our proposed system can be seen in Figure 5.1. The Skill Network introduced in section 2.3 provides a student's skill levels to the Player Experience Model, which is then fed to the Content Generator. The Content Generator calculates a utility for each tile in the Tile Database using the evaluation function and the skill levels from the player experience model. The estimated utilities are then used to generate a personalized lesson that the student can play. After a lesson, new skill levels are inferred, and the process can be repeated.

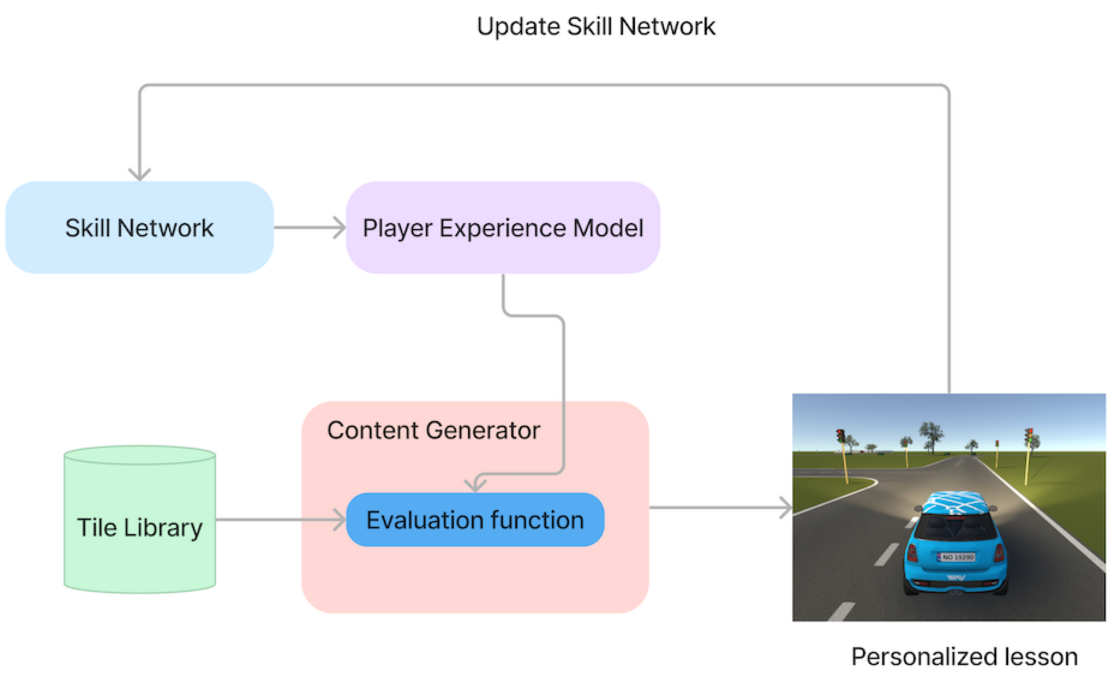


Figure 5.1: Model Diagram

5.2.1 Player Experience Model

In section 3.5, the concept *Player Experience Modeling* (PEM) and the three different types: subjective, objective, and gameplay-based PEM are introduced. Our proposed solution utilizes gameplay-based PEM to model the player experience. The data on which we build our PEM come from the WAY skill model, as described in Section 2.3. The skill network evaluates the current driver on 52 different skill metrics in real time based on how the student is driving and gives the student a score between 0 and 1 for each skill. Due to limited time, we have chosen to narrow the set of skills down to a subset of 15 skills, where a sample of the skills with belonging skill levels can be seen in table 5.1. The skill levels provide an accurate representation of the student's current skill level and serve as the foundation for creating an accurate Player Model. In order to personalize traffic situations in the context of driving skills, it is necessary to consider not only specific skills but also general aspects. This can be achieved by incorporating content that is not directly tied to a specific skill but affects multiple skills. To account for this general skill level, the player model also contains the parameter *Average Skill Level*, which comprises the average of all skill levels. This parameter provides a measure of the student's overall proficiency in driving. By considering both specific skills and the general skill level, a more comprehensive evaluation of the student's driving abilities can be obtained, allowing for a more personalized and accurate representation of their skillset.

Name	**Skill Level**
new_start_mastery	0.553
driving_curves_mastery	0.893
tunnel_driving_mastery	0.789
gap_chance_mastery	0.378
intersection_mastery	0.867
yielding_rules_mastery	0.643
traffic_lights_rules_mastery	0.512
queue_driving_mastery	0.940
risky_animal_crossing_handling_mastery	0.489
highway_driving_mastery	0.278
overtake_mastery	0.448
speed_limit_mastery	0.992
roundabout_mastery	0.150
risky_pedestrian_handling_mastery	0.489
emergency_brake_mastery	0.250

Table 5.1: Example of skill levels returned from the Skill Network

5.2.2 Content Representation

To satisfy the system requirements **R1** and **R2**, an accurate content representation must be created to represent the game content in an efficient way that allows for the dynamic creation of lessons. With that in mind, the following components and semantics are proposed. The world is represented as a *lesson* which can contain various amounts of *chunks*, which in turn consist of multiple *tiles*. The class diagram in figure 5.2 shows an overview of the individual components and their semantics.

Tile

In the game world constituting a driving lesson, *tiles* are the fundamental building blocks. Each tile is a self-contained unit that encapsulates a specific scenario and is designed to represent various traffic situations the students could encounter. Inspired by the semantics introduced in section 4.2.1, the semantics of a tile, as seen in table 5.2, is used to represent the name of the tile, the locations where roads are entering or exiting the tile **R**, the specific skills trained by the tile **S**, its associated difficulty **D**, and the different distractors the tile can contain **d**.

Class Tile:
Name **N**
Incoming/outgoing road(s) **R**
Can train skill(s) **S**
Difficulty **D**
Can contain distractor(s) **d**

Table 5.2: Tile class schema

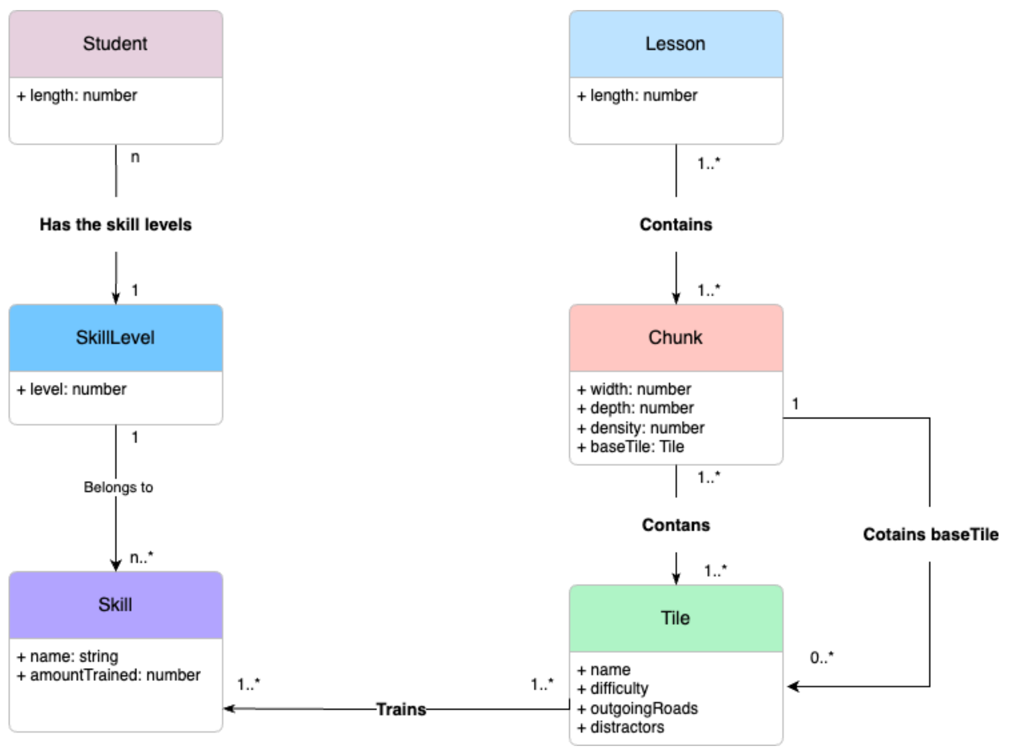


Figure 5.2: Class diagram

Where roads are exiting and entering a tile is used when tiles are combined into a chunk to ensure that the game world is constructed in a cohesive and interconnected manner. Each tile in the game serves the purpose of training the player in at least one essential driving skill. The tile semantics cover what skill a given tile trains and with what efficiency it trains the given skill, later referred to as the tile's *learning efficiency* of a given skill. To accommodate every student's general skill level, a predetermined difficulty, represented by a number between 0 and 1, is assigned to each tile. The is set to either one of the values from table 5.3.

Difficulty	**Level**
Beginner	0.15
Novice	0.30
Intermediate	0.45
Experienced	0.60
Expert	0.75

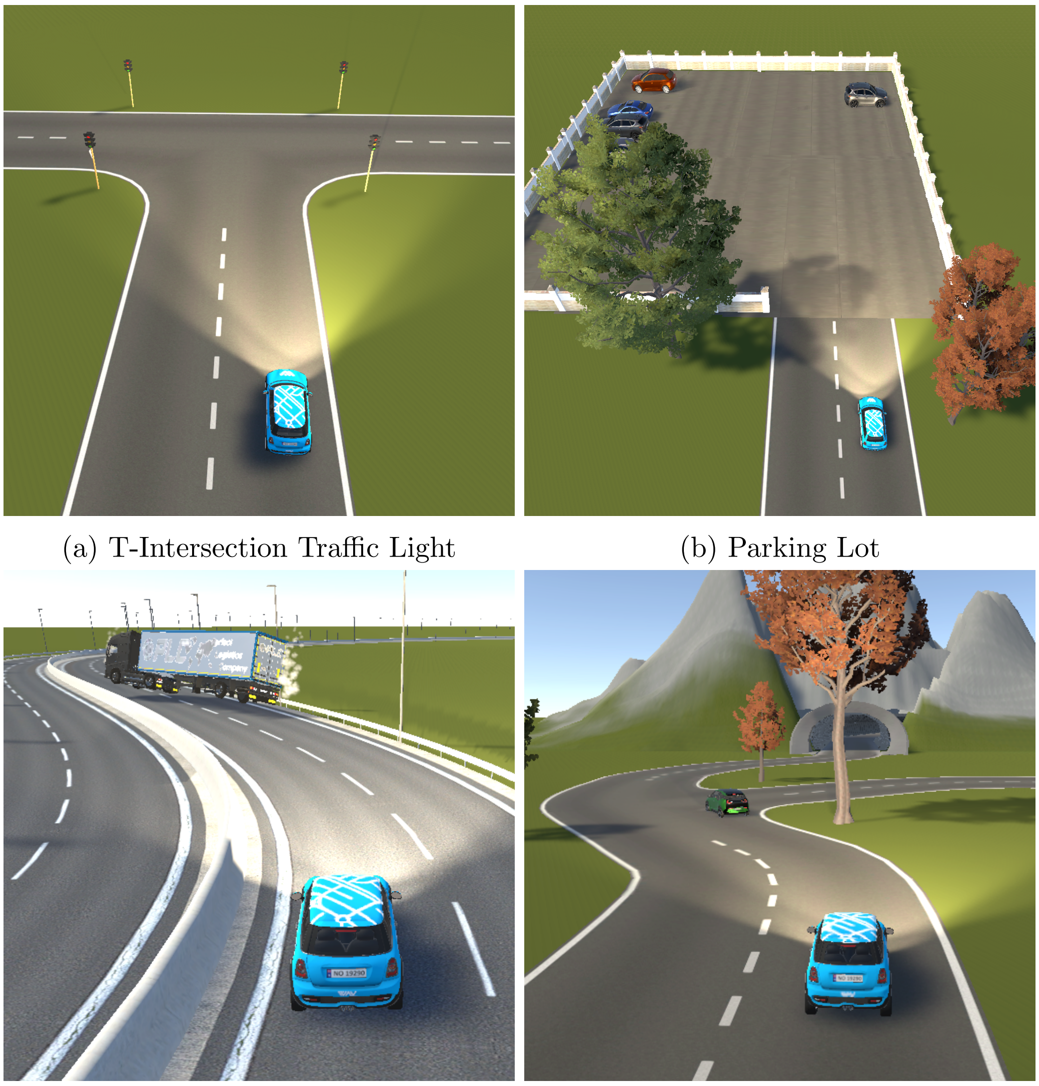
Table 5.3: Tile Difficulty

This allows for adjustment of the game world and its challenges based on the player's proficiency, ensuring an appropriate and engaging experience that aligns with their skill level. A tile can also contain one or more *distractors*. Distractors are additional elements that can be added to a tile to increase its difficulty. With these values, the compatibility between the tiles and the different distractors can be controlled. The implemented distractors are traffic and pedestrians. By designing tiles this way, we can create individual pieces of the game world that create a consistent experience. Examples of tiles can be observed in figure 5.3, while the instantiation of the available tiles is presented in Table 5.4. The column labeled *Outgoing Roads* provides an overview of the initial directional configurations, but it allows for the rotation of tiles to accommodate the generated lesson.

Tile	**Difficulty**	**Outgoing Roads**	**Distractors**
U-Turn Parking Lot	0.15	Down	Pedestrians
T-Intersection Traffic Lights	0.45	Down, Left, Right	Pedestrians, Cars
X-Intersection Traffic Lights	0.45	Up, Down, Left, Right	Pedestrians, Cars
Turn	0.15	Down, Left	Pedestrians, Cars
T-Intersection	0.30	Down, Left, Right	Pedestrians, Cars
Straight	0.15	Up, Down	Pedestrians, Cars
Roundabout	0.45	Up, Down, Left, Right	Pedestrians, Cars
Straight Curved	0.15	Up, Down	Pedestrians, Cars
T-Intersection Curved	0.30	Down, Left, Right	Pedestrians, Cars
Turn Curved	0.15	Down, Left	Pedestrians, Cars
T-Intersection Yield	0.45	Down, Left, Right	Pedestrians, Cars
T-Intersection Stop	0.45	Down, Left, Right	Pedestrians, Cars
X-Intersection	0.30	Up, Down, Left, Right	Pedestrians, Cars
X-Intersection Crosswalk	0.45	Up, Down, Left, Right	Pedestrians, Cars
Straight Road Crosswalk	0.30	Up, Down	Pedestrians, Cars
Risky Animal	0.60	Up, Down	-
Highway	0.15	Up, Down	Cars
Tunnel	0.30	Up, Down	Cars
Overtake	0.45	Up, Down	-
Queue	0.30	Up, Down	-
Emergency	0.75	Up, Down	Cars
Advanced Highway	0.60	Up, Down	Cars

Table 5.4: Tile Details

Creating tiles as building blocks in Unity largely relies on the existing method employed by WAY, as described in 2.2. The physical roads are generated with EasyRoads3D, and WAY's in-house software is utilized to create the non-physical road network to provide consistency for cars and pedestrians.


(a) T-Intersection Traffic Light (b) Parking Lot

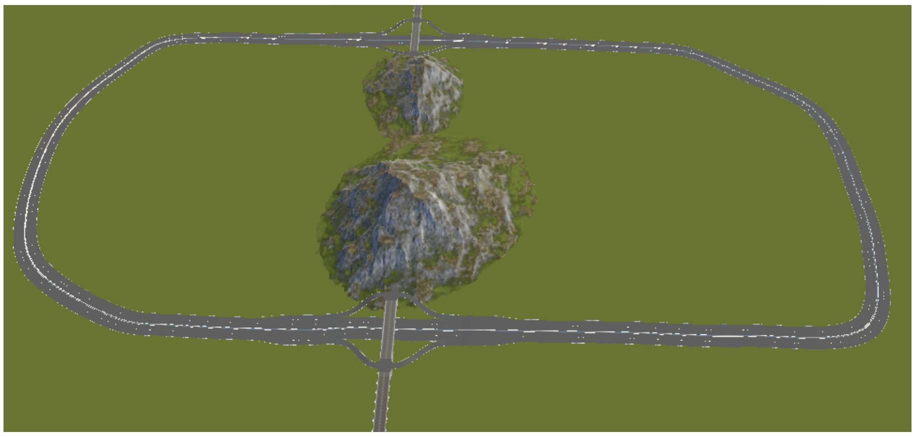
(c) Emergency (d) T-Intersection Curved

Figure 5.3: Example of Tiles

Chunk

In section 3.1, the design principle *Manage the learner's cognitive load* is introduced. To optimize learning, learning material should be organized into smaller chunks that gradually become more complex. To comply with this, the component *chunk* is introduced. A chunk consists of multiple tiles organized in a grid. The tiles within the chunk form a road network that the student can traverse, and where the focus is to train on a limited set of skills. Figure 5.4 shows examples of two chunks.

Another reason to introduce the concept of chunks is that tiles alone are not enough to answer the research question **RQ2**. To create traffic situations that train specific skills, such as the skills *overtake_mastery* and *tunnel_driving_mastery*, or to add traffic that seems coherent and natural, consistency is needed over several *tiles*, and *chunks* are used to obtain this consistency. In addition, using chunks to maintain consistency over multiple tiles is also beneficial to create traffic situations that require roads with multiple lanes. To avoid rapid changes in the number of lanes


(a) Example of Chunk training highway skills

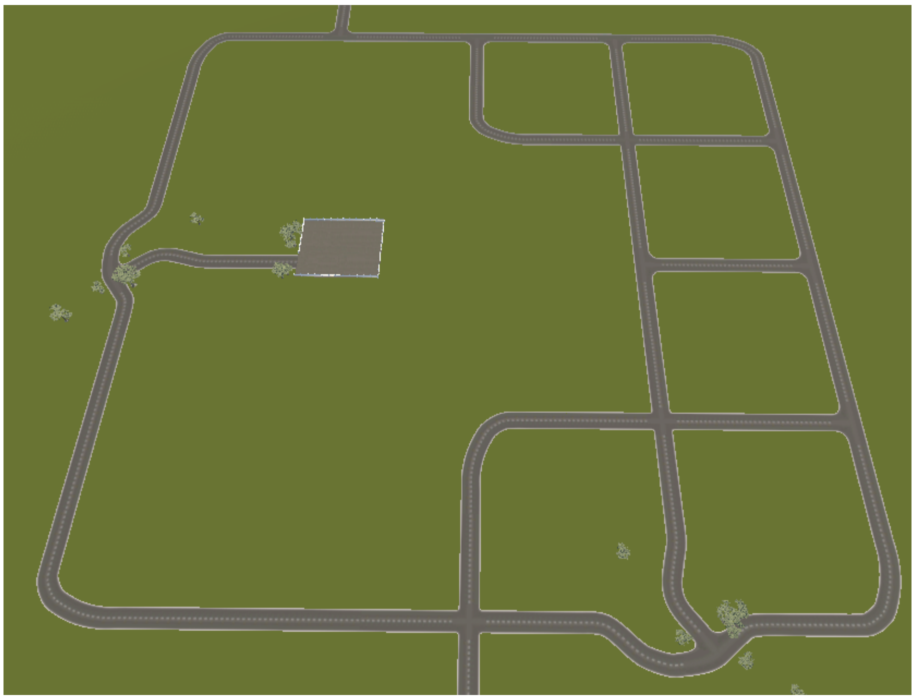

(b) Example of Chunk training intersection skills

Figure 5.4: Example of Chunks

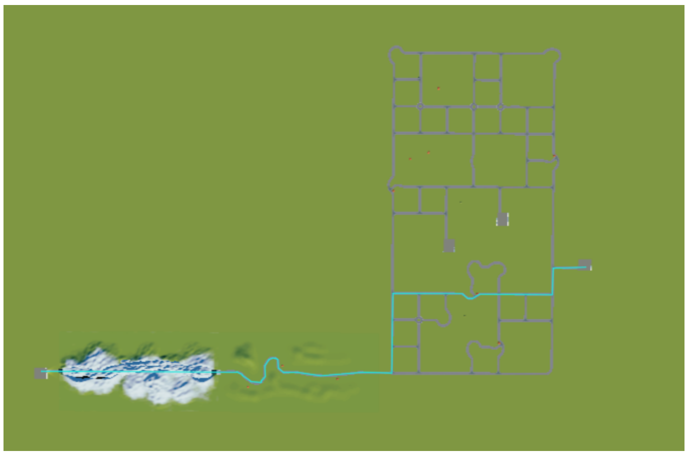
between different tiles, whole chunks can be generated with the same number of lanes to maintain consistency and continuity. To ensure that the difficulty level of driving lessons matches the user's skill, the chunks can be adjusted by changing parameters such as the amount of traffic and the occurrence of pedestrians. Choosing the amount of traffic entails deciding the number of vehicles and their movement patterns, taking into account factors such as traffic flow, congestion, and road capacity to help simulate realistic traffic scenarios. Adjusting the occurrence of pedestrians involves determining the amount, their behaviors, and their interactions with the vehicles and the environment. Including pedestrians adds another layer of realism and complexity to the simulation. Traffic and pedestrians can then be added to all tiles within the chunk that are compatible with the different distractors. By modifying these parameters, the model can introduce more distractions and, thereby, the amount of stress imposed on the student, subsequently enhancing the level of difficulty for the task without directly targeting any particular skills. A proposed mapping between general skill, traffic, and pedestrian levels can be seen in table 5.5. A chunk's width, length, and density can also be adjusted. The width and length determine how large each chunk will be and, subsequently, how fast a student can move between different chunks. The density in this context refers to how dense traffic situations are placed in the game world. The density is used to increase the probability of certain traffic situations. When the grid that organizes the tiles is created, tiles with three or four exit points, such as t-intersections, x-intersections, and roundabouts, demand a higher density because they require more roads. Because of this, increasing the density increases the occurrence of those tiles.

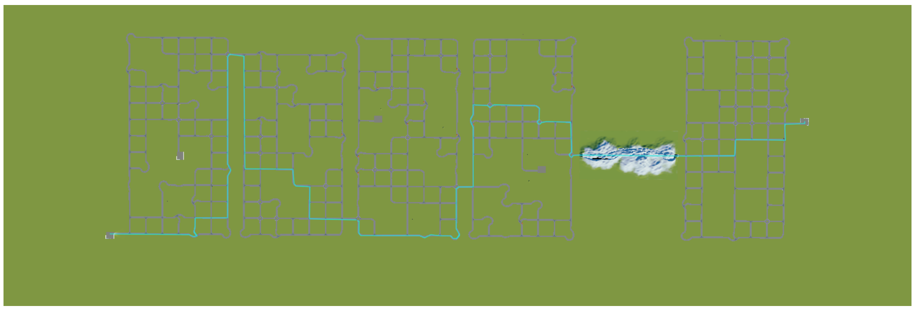
General skill level	**Traffic**	**Pedestrians**
Beginner (0 - 0.3)	None	None
Intermediate (0.3 - 0.45)	Low	None
Advanced (0.45. - 0.6)	Moderate	Low
Expert (0.6 - 1)	High	Moderate

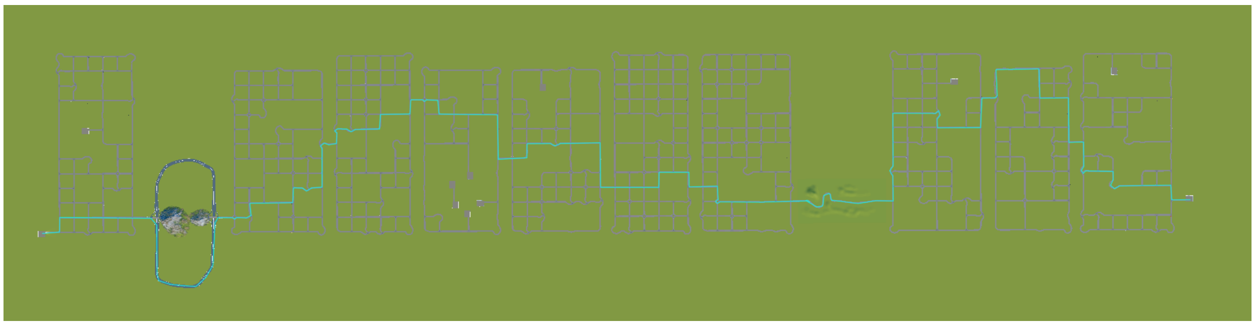
Table 5.5: General skill level and corresponding obstacle levels

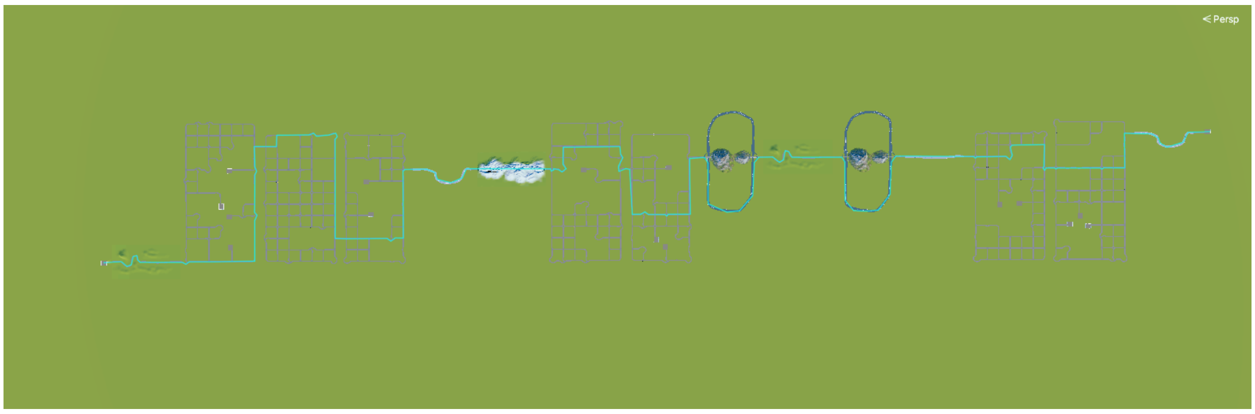
Lesson

A *Lesson* in our proposed model is a sequence of *chunks* organized to maximize the learning outcome for a student. As discussed earlier, each chunk focuses on training a limited set of skills, making it crucial to arrange these chunks strategically in a lesson to provide comprehensive and incremental learning experiences. Several principles guide the organization of chunks into lessons. First, lessons are designed to have an appropriate length. Lessons that are too short may not provide enough practice, and the loading time between lessons could be bothersome. Lessons that are too long may lead to fatigue and loss of focus. In addition to the lack of focus, the skill model is only updated after each lesson. The result of that is that the skill levels will be increasingly outdated the longer the student drives. In figure 5.5, four lessons with different lengths can be seen, with their corresponding number of chunks and estimated duration in minutes. An optimal path, which will be introduced in section 5.3, true the lesson is also visualized.


(a) Lesson with 3 chunks, 4.5 minutes estimated duration.


(b) Lesson with 6 chunks, 9 minutes estimated duration.


(c) Lesson with 9 chunks, 13.5 minutes estimated duration.



(d) Lesson with 15 chunks, 22.5 minutes estimated duration.

Figure 5.5: Different lesson lengths

Second, the sequence of chunks in a lesson is designed to provide variety to keep the learner engaged. The lesson design also considers the interleaving of different skills, in line with research showing that interleaved practice can lead to better learning outcomes than blocked practice [10]. Combining these principles enables the generation of lessons that are engaging, effective, and personalized for each learner's skill level. An example of a generated lesson can be seen in figure 5.6

5.2.3 Content Quality

To decide what constitutes high content quality, a high-quality tile is defined as one that promotes the optimization of learning and replayability. A set of equations that maps the skill levels, the available tiles, and their corresponding difficulties to a scalar that reflects the tile's suitability is presented.

The initial task involves designing a utility function that effectively captures the compatibility between skill levels and previously chosen tiles. This utility function plays a key role in determining which skills should be prioritized. To facilitate understanding of the formula, the following parameters are introduced:

- U_s represents the utility for a skill s.
- M_s represents the student's mastery of the skill s. This number is extracted from the output of the skill model and is a continuous number between 0 and 1, where 0 represents no experience with that skill, and 1 represents full mastery.
- u represents the weight with which the inverse mastery should be amplified.
- F_s represents the frequency of skills s. F_s is the accumulated amount that this skill has been trained in the lesson. This parameter is added to avoid repetition.
- v represents the weight with which a skill frequently chosen should be punished. Increasing v promotes more diversity in skill representation.

$$U_s = (1 - M_s)^u - (F_s \times v) \tag{5.1}$$

The utility U_s described in equation 5.1 consists of two terms. The first term contains the inverted learning mastery parameter M_s raised to the power of u. Taking the inverse relates to calculating *1 - M_s* in this context and results in higher values for skills in which the students perform poorly. This is exponentiated by u to amplify this property further, creating a larger difference between high-mastery skills and low-mastery skills. The second term consists of the skill frequency F_s times the skill frequency penalty weight w. This expression is subtracted from the first to penalize skills trained earlier in the lesson. Lastly, the method ensures that no skill can be trained in succession, adhering to the interleaving introduced in section 5.2.2.

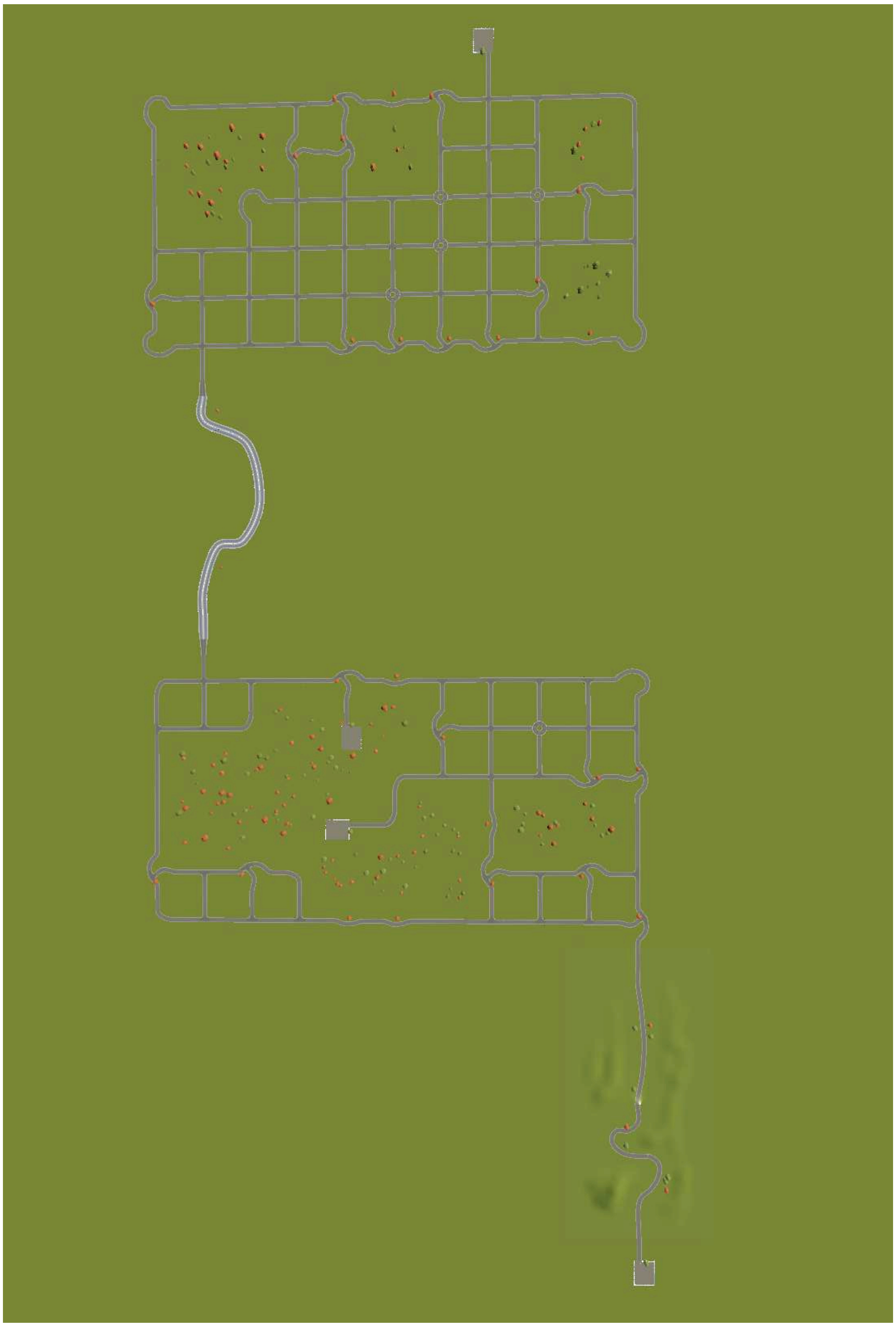


Figure 5.6: Generated lesson

Next, we need to devise a method to rate tiles according to how well their level of difficulty matches the students' current mastery of the corresponding skills. The following parameters serve as the foundation for this operation:

- β_t represents the difficulty alignment reward for tile t.
- M_ϕ represents the student's level of mastery of the skill ϕ that is trained with the highest efficiency by tile t.
- T_d represents the difficulty d corresponding to the tile.
- N_0 represents the reward or utility when the difficulty level matches the skill level.
- λ represents the decay rate and decides how quickly the reward decreases.

$$\beta_t = N_0 \times e^{-\lambda \times |M_\phi - T_d|} \tag{5.2}$$

Equation 5.2 determines the reward β_t a tile t should receive. The expression is known as the exponential decay function. N_0 determines the reward for identical difficulty and skill level, whereas λ represents the rate at which the reward decays. The absolute value is applied to the difference between the tile's difficulty T_d and the student's mastery of the main skill M_ϕ to ensure a positive number.

The remaining step comprises a measure for tile utilities, and subsequently, this set of parameters is defined:

- U_t represents the utility U for tile t.
- $s \in t$ represents all the skills that tile t trains.
- L_s represents the tiles learning efficiency of skill s. This continuous number between 0 and 1 tells us to what degree this tile trains the given skill.
- n_t represents the number of skills n that tile t trains.

$$U_t = \beta_t + \frac{\sum_{s \in t} L_s \times U_s}{n_t} \tag{5.3}$$

The equation 5.3 determines a given tile t's utility. The formula summarizes the skill utilities U_s multiplied by the learning efficiency L_s for each skill that tile t trains. This score expression is divided by the number of skills trained n_t to penalize tiles that train a large set of skills. Furthermore, the difficulty reward β_t is added to the score. The resulting scalar U_t represents how well the given tile fits a student. Lastly, similar to skill utilities, the interleaving concept is incorporated, ensuring that the base tile isn't selected successively.

Lesson difficulty

To analyze if the content that is represented and generated is of the appropriate difficulty level, to ensure that **RQ3** is met, we need a way to set a difficulty score for a given lesson. We use the average base tile difficulty as a metric to analyze the difficulty of a given lesson. The average base tile difficulty is calculated by the formula seen in equation 5.4, where the following parameters are used:

- L_d represents the difficulty score for lesson L.
- T_{id} represents the difficulty of tile T_i.
- T_{iD} represents the number of distractors that are added to tile T. This number is dependent on the average skill level and the data in table 5.5.
- w represents the weight that should be multiplied by the number of distractors.

$$L_d = \frac{\sum_{i=1}^{n} T_i{}_{d+T_iD\times w}}{n} \tag{5.4}$$

5.2.4 Content Generation

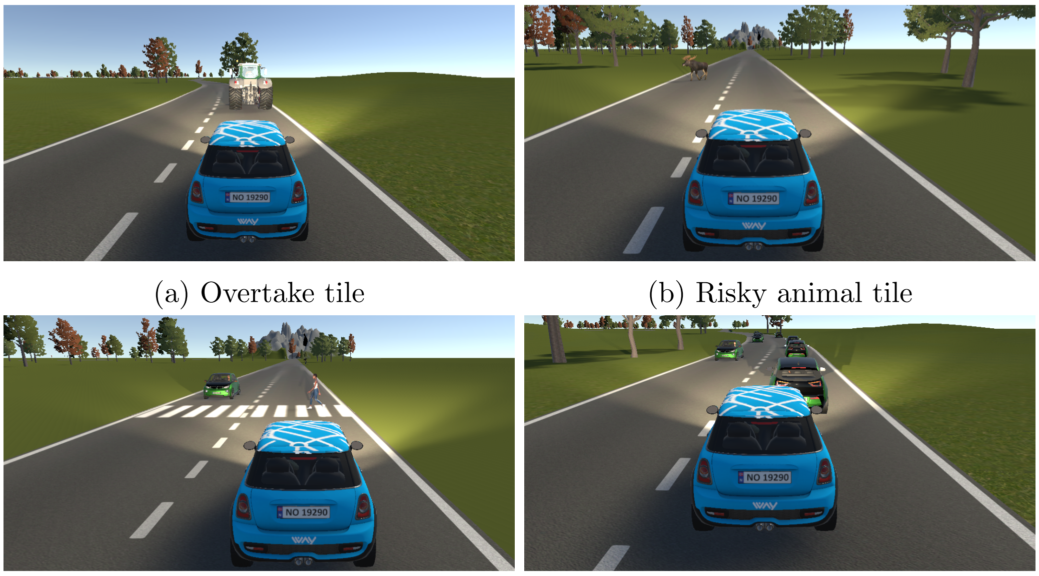
Using the content representation proposed in Section 5.2.2 and the utility function discussed earlier, we can combine tiles and chunks in various sequences while adjusting the difficulty levels. This allows us to create an optimal learning experience tailored to the individual skill level of each student.

Dynamic Content

Although the majority of this chapter has focused on the generation of static content in the form of tiles and chunks to construct consistent, drivable game worlds, the lesson generation process also includes the generation of dynamic content. Dynamic content refers to all types of content that is moving and interacting with the game world except the car the student controls. This includes objects such as pedestrians, vehicles of different sorts, and animals. In this system, dynamic content is separated into two types: tile-based events and distractors.

Tile-Based Events Tile-based events are dynamic content that is located on a specific tile, has a predefined path that it follows and starts to interact with the world when the student is approaching the tile. Numerous skills in the skill network require particular events to happen at the correct time to provide students with the necessary situation for skill development. For instance, the practical training of an overtaking maneuver necessitates a slow-moving vehicle in front, complemented by an appropriately structured road conducive to overtaking.

In order to facilitate these tailored learning experiences, we have created a variety of tiles such as Overtake, Risky Animal, Straight Road Crosswalk, X-Intersection Crosswalk, and Queue as displayed in figure 5.7. Each of these tiles is designed to trigger specific events when the student driver approaches, providing the relevant scenarios needed for effective skill training.



(a) Overtake tile (b) Risky animal tile

(c) Crosswalk tile with pedestrian (d) Queue

Figure 5.7: Dynamic Tiles

Distractors Distractors are additional dynamic elements designed to introduce complexity into the learning environment, thereby increasing the level of difficulty. These distractors aim to emulate the unpredictability of real-world driving and stimulate higher-level cognitive processing in student drivers. We have categorized distractors into two main types: traffic and pedestrians. These distractors can be incorporated into any compatible tile, further enriching the simulation environment and adding to the realistic driving experiences for the student. By introducing distractors, the system can adjust its content to answer the system requirement **R6**.

Through the inclusion of both static and dynamic content in the lesson-generation process, we have sought to create a comprehensive, immersive, and effective driver training environment. The interplay between these elements allows for a rich and engaging experience that is tailored to the unique learning needs of each student.

Generating a Lesson

The process of generating a personalized composition of tiles and chunks is described in algorithm 1. This algorithm outlines the steps in constructing a lesson considering the available tiles, the student's skill level, and the number of chunks that should be included in the lesson. The first step is to identify the base tile to be included for each chunk. The function *GetBaseTile(tiles)* is responsible for selecting the most appropriate tile based on factors such as the student's skill level and previously chosen tiles. The GetBaseTile function determines the most suitable tile from the available tiles provided. This is achieved by calculating a 'utility' or value for each tile, utilizing equations 5.1, 5.2, and 5.3. This process entails a series of operations, with the parameters and equations defined below.

- S_p represents the normalized probability for each skill S to be chosen as the next skill to train.
- T_p represents the normalized probability for each tile T to be chosen as the next base tile.
- D_s represents the drawn skill based on the probabilities S_p.
- D_t represents the drawn tile based on the probabilities T_p.
- $s \in S_n$ represents every skill s in the skill network S_n.
- $t \in D_s$ represents every tile t that trains the drawn skill D_s

$$S_p = \frac{U_s}{\sum_{s \in S_n} [U_s]} \tag{5.5}$$

The initial step, depicted in Equation 5.5, involves normalizing the utilities U_s obtained from equation 5.1 for each available skill. This normalization process transforms the utilities into skill probabilities S_p, which are then utilized in the following step.

$$D_s \sim (S_1, S_2, \ldots, S_p) \tag{5.6}$$

These probabilities are subsequently utilized to randomly select the desired skill to be trained next, as outlined in equation 5.6. Once the skill is determined, the tile utilities U_t are calculated for the tiles that facilitate training of the selected skill, following the formulas presented in equations 5.2 and 5.3.

$$T_p = \frac{U_t}{\sum_{s \in S_t} [U_t]} \tag{5.7}$$

After all of the tile utilities U_t are obtained, they are normalized using equation 5.7 to derive tile probabilities T_p for the purpose of tile selection.

$$D_t \sim (T_1, T_2, \ldots, T_p) \tag{5.8}$$

Lastly, a base tile is determined using equation 5.8. This approach guarantees that the higher the utility, the higher the probability of selection while still allowing for randomness in the process.

The drawn base tile subsequently serves as the foundation to generate a chunk using the *GenerateChunk(baseTile, skillNetwork)* function. Following this algorithm allows for the creation of a personalized learning experience that combines relevant tiles and chunks to align with the student's abilities and learning objectives. This approach maximizes the effectiveness of the learning process and enhances the student's overall educational outcomes. An example of a generated lesson can be seen in figure 5.6.

Algorithm 1 Generate lesson

```
1: function GenerateLesson(tiles, skillNetwork, numberOfChunks)
2:     for $i \leftarrow 0$ to $numberOfChunks$ do
3:         $baseTile \leftarrow$ GetBaseTile(tiles)
4:         GenerateChunk(baseTile, skillNetwork)
5:     end for
6: end function
```

Generating a Chunk

During the generation of a chunk, five key considerations need to be taken into account. Firstly, the size of the chunk needs to be determined. The second consideration pertains to the density of traffic situations. Thirdly, the amount of traffic within the chunk must be considered. The fourth consideration is the presence of pedestrians within the chunk, and lastly, it is essential to determine which tiles should be included in the chunk. Determining which tiles should be included involves selecting the optimal tiles to maximize the optimization of learning outcomes. The procedure of generating a chunk is described in algorithm 2.

Algorithm 2 Generate chunk

```
1: function GenerateChunk(baseTile, skillNetwork)
2:     $density, size \leftarrow$ GetChunkDimensions(baseTile)
3:     $traffic, pedestrians \leftarrow$ GetDistractors(skillNetwork)
4:     $grid \leftarrow$ GenerateGrid(baseTile, skillNetwork, density, size)
5:     return $grid, traffic, pedestrians$
6: end function
```

GetChunkDimensions retrieves the density and chunk size directly from a base tile configuration mapping, where a base tile constitutes the density and dimensions of the corresponding chunk. The mapping between base tiles and chunk dimensions is set to facilitate the generation of the base tile within the chunk. Typically, an intersection-like base tile will have a high density, whereas a base tile that facilitates straight driving will have a smaller width.

GetDistractors use the general skill level to decide the appropriate difficulty and subsequently sets the amount of traffic and pedestrians, corresponding to table 5.5. *GenerateGrid* involves a more complex process, as it needs to factor in some of the previous key considerations. This function initializes a grid with the defined size and proceeds to populate the grid with tiles. Density decides the number and placement of blank grass tiles, and the remaining choices are made utilizing the skill network. Even though the base tile serves as the main training objective, it is also beneficial to optimize other tiles in the lesson. This is done similarly to how base tiles are chosen, only differing in a lack of discounting selected tiles.

By addressing these five concerns, namely chunk size, density, traffic, pedestrians, and tile selection, the generation process can effectively create chunks that offer varied and realistic traffic situations tailored to the specific needs of the simulation.

5.3 Pathfinder

Once the lesson is entirely created, a path that guides the player around the world is desirable. While the generated chunks with included base tiles are tailored to train a given skill, they also include various tiles that provide training opportunities for other skills. To guarantee that the individual currently driving visits all of the selected base tiles, an algorithm for ensuring this is proposed.

To accomplish this, the lesson is converted to a *search graph* with corresponding *graph nodes*. Each *graph node* is associated with a position, a cost, and a belonging tile, where position addresses the placement on the grid, cost relates to how expensive it is to travel to a given node, and tile addresses the tile placed on this position. A graph node containing a base tile is called a *base tile node*. Once the search graph is initialized, *FindShortestPath*, as seen in algorithm 3, is invoked with the search graph, the start node, and the base tiles provided from the initial generation as the arguments.

Algorithm 3 Optimal Path

```
1: function FindShortestPath(searchGraph, start, baseTiles)
2:     bestPath ← empty list
3:     nodesToVisit ← FindNodesToVisit(baseTiles, searchGraph, start)
4:     for $i$ from 0 to length(nodesToVisit) $-$ 1 do
5:         end ← GetNode(searchGraph, nodesToVisit[$i$])
6:         path ← Djikstra(searchGraph, start, end)
7:         start ← end
8:         append path to bestPath
9:     end for
10:    return bestPath
11: end function
```

The algorithm is now aware of which tiles the individual should visit, but neither the location nor path to the tiles is known. To find the location of each base tile, *FindNodesToVisit* is initiated. This function is responsible for providing an array of nodes that contain the desired base tiles in a reasonable order. The method employs the *Manhattan Distance*, defined in Equation 5.9, as a heuristic to determine the most favorable nodes to visit.

$$\text{ManhattanDistance}(start, end) = |start\_x - end\_x| + |start\_y - end\_y| \tag{5.9}$$

The function operates by selecting the base tile node that has the lowest Manhattan distance from the starting node and appends it to the array. Subsequently, the starting node is updated to the last base tile node added to the array, and the operation is repeated. Once the algorithm knows which nodes to visit, along with the location and order of these base tile nodes, it iterates over these nodes and, for each node, runs Dijkstra's algorithm [8] for the shortest path to find the optimal path to a given base tile node. The starting node for each iteration is subsequently changed to the target node of the previous path search. An example of a path guiding the student through a lesson with the base tiles Roundabout, Highway, T-Intersection, and Overtake can be seen in figure 5.8. By ensuring that each base tile is visited in a reasonable order, this algorithm provides the student with an effective and suitable path through the game world, thereby addressing **R4**. As a consequence of this path, the system also prevents prolonged driving on the same chunk.

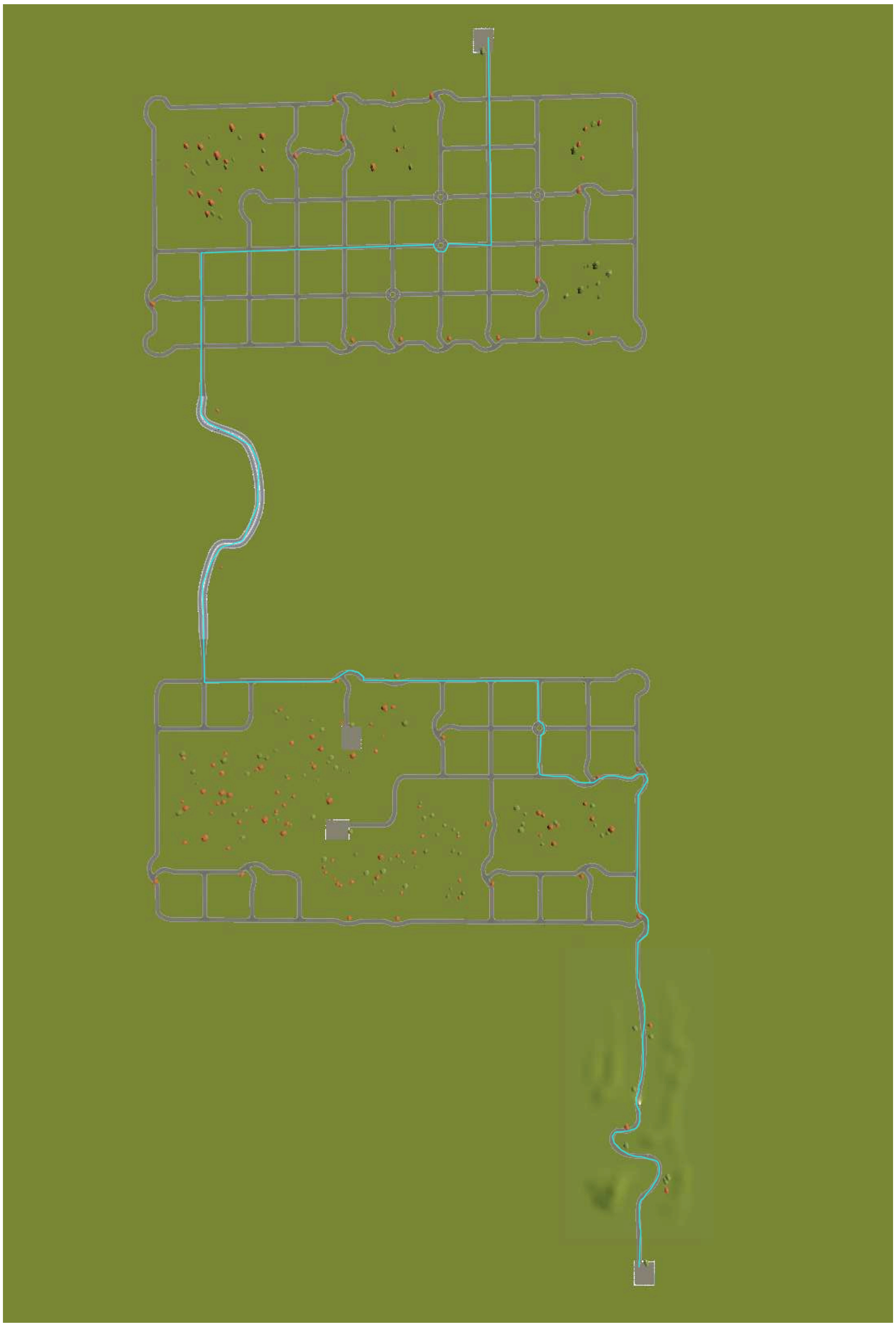


Figure 5.8: Visualized path through a lesson

5.4 Summary

In this chapter, a set of system requirements derived from the research questions are introduced. These system requirements form the basis for the architecture. The architecture design follows the Experience Driven Content Generation framework and is subsequently divided into Player Experience Modeling, Content Representation, Content Quality, and Content Generation. In Player Experience Modeling, the utilization and integration of the existing skill network from section 2.3 is explained. When it comes to Content Representation, it is paramount that the representation supports the capability of representing all traffic situations, thus facilitating answering **RQ2**. Content quality introduces essential utility functions utilized in Content Generation to ensure optimal content creation based on the student's skill level. Lastly, a pathfinder algorithm is presented that aids the system in ensuring that desirable traffic situations are encountered.

Chapter 6

Experiment and results

This chapter presents the conducted experiment and the resulting outcomes. The chapter is divided into a section describing the experimental plan, followed by an elaboration of the experimental setup, where four distinct scenarios and vital configurations are introduced. The last section presents the experimental results.

6.1 Experimental Plan

The goal of the experiment conducted in this thesis is to determine if the system designed in chapter 5 can answer the research questions **RQ1** and **RQ3** introduced in section 1.2. To reiterate, the two relevant questions are listed below.

1. How can we facilitate the creation of personalized driving lessons to optimize learning?
2. How can we ensure an appropriate level of difficulty for each specific traffic situation?

To address the questions, we aim to create diverse scenarios that reflect students with varying skill levels. Subsequently, we will use the proposed system to generate lessons tailored to the skill levels in each scenario and then make a comparison between the generated lessons. This comparative analysis will enable us to gain insights into the effectiveness of the different lessons for students with distinct levels of proficiency. Creating our own scenarios is motivated by the following two factors:

1. **Controlling Variables:** In a simulation, the researcher has complete control over the conditions and variables. By creating scenarios, we can create diverse skill models to better highlight the strengths and weaknesses of the system.
2. **Reproducibility:** Simulation-based research often offers better reproducibility compared to real-world experiments. By creating your own scenarios, you can precisely recreate and replicate specific driving situations, ensuring consistent testing conditions.

6.2 Experimental Setup

This section will provide an overview of the experimental setup and the implementation of the experiment, including the input data used. Included in the experimental setup are the creation of scenarios and the definition of learning efficiency configurations. The experiments will involve the generation of a total of 40 customized driving lessons, with 10 lessons designed for each of the four distinct scenarios and each lesson containing 15 chunks.

6.2.1 Scenarios

To optimize the impact of our findings, we have devised four distinct scenarios, each depicting a student with a corresponding skill model that emulates the Skill Network's output. By manipulating the variables and constructing our own scenarios, we can effectively highlight the system's advantages and limitations. This shifts the primary concern in scenario design away from realism and instead emphasizes exploring edge cases that unveil these strengths and weaknesses.

Scenario 1 - Low Proficiency

The first scenario depicts a novice student who lacks prior driving experience. This is reflected in their skill levels, all of which are below average values, as illustrated in table 6.1. Seeing as this student has rather similar skill levels for each skill, the model should manage to generate a diverse portfolio of lessons with suitable difficulty.

Name	**Skill Level**
new_start	0.23
driving_curves	0.35
tunnel_driving	0.16
gap_chance	0.22
intersection	0.24
yielding_rules	0.20
traffic_lights_rule	0.19
queue_driving	0.29
risky_animal_handling	0.19
overtake	0.15
speed_limit	0.29
roundabout	0.21
risky_pedestrian_handling	0.14
highway_driving	0.26
emergency_brake	0.19

Table 6.1: Scenario 1 - Skill Levels

Scenario 2 - Low Proficiency with Strengths

The second scenario exemplifies a student with below-average overall proficiency but with a few outlier skills in which the student excels. The skill levels for this scenario are outlined in table 6.4. Specifically, the student demonstrates strong performance in *yielding_rules*, *roundabout*, and *highway_driving*. To optimize the learning experience, the generated portfolio of lessons should prioritize training for the remaining skills, focusing on appropriate difficulty levels while avoiding notable training in the aforementioned proficient skills.

Name	**Skill Level**
new_start	0.23
driving_curves	0.21
tunnel_driving	0.31
gap_chance	0.28
intersection	0.33
yielding_rules	**0.86**
traffic_lights_rules	0.35
queue_driving	0.26
risky_animal_handling	0.28
overtake	0.25
speed_limit	0.25
roundabout	**0.82**
risky_pedestrian_handling	0.34
highway_driving	**0.78**
emergency_brake	0.25

Table 6.2: Scenario 2 - Skill Levels

Scenario 3 - High Proficiency with Weaknesses

The third scenario, represented by the skill levels in table 6.3, features a student with an overall high average proficiency. However, this student also possesses a few outlier skills with low values. Notably, the skills of *tunnel_driving*, *queue_driving*, and *highway_driving* are areas in which the student performs poorly. The generated results are expected to exhibit a bias toward prioritizing the training of these skills with an adequate level of difficulty.

Name	Skill Level
new_start	0.67
driving_curves	0.69
tunnel_driving	**0.32**
gap_chance	0.56
intersection	0.79
yielding_rules	0.85
traffic_lights_rule	0.78
queue_driving	**0.25**
risky_animal_handling	0.70
overtake	0.68
speed_limit	0.84
roundabout	0.74
risky_pedestrian_handling	0.92
highway_driving	**0.19**
emergency_brake	0.77

Table 6.3: Scenario 3 - Skill Levels

Scenario 4 - High Proficiency

The fourth and final scenario depicts a highly experienced student who has accumulated significant driving experience, resulting in elevated skill level values as outlined in table 6.2. To adapt to the needs of this proficient learner, the generated driving lessons should ideally be diverse in nature and offer appropriate levels of difficulty that align with the student's advanced skill set.

Name	Skill Level
new_start	0.73
driving_curves	0.68
tunnel_driving	0.65
gap_chance	0.63
intersection	0.82
yielding_rules	0.74
traffic_lights_rules	0.81
queue_driving	0.64
risky_animal_handling	0.70
overtake	0.63
speed_limit	0.80
roundabout	0.77
risky_pedestrian_handling	0.68
highway_driving	0.66
emergency_brake	0.72

Table 6.4: Scenario 4 - Skill Levels

6.2.2 Parameters and Configurations

During the execution of the experiments, it is necessary to determine the previously introduced parameters and some notable configurations. Subsection 5.2.3 introduces equation 5.1, which calculates the skill utility u_s, and equation 5.2, which computes the difficulty alignment reward β_t. The selection of the desired skill for training involves the utilization of the skill utility, which depends on the parameters u and v. Once the skill is chosen, all suitable base tiles are assigned a utility score U_t as described in equation 5.8. The difficulty alignment reward influenced by N_0 and λ is used to boost the scores for these base tiles according to difficulty compatibility. The weight u is used to represent the difficulty of the distractors when calculating lesson difficulty. In the experiments, these parameters are defined based on the values presented in table 6.5.

Parameter	**Value**
u	2
v	0.05
w	0.005
N_0	0.1
λ	5

Table 6.5: Parameter Values

This choice of parameters should ensure that there exists a reasonable relationship between the skill utilities, base tile utilities, and difficulty alignment rewards. The consequent equations can be seen below.

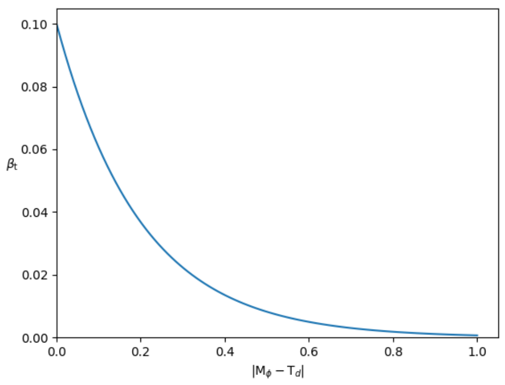
$$U_s = (1 - M_s)^2 - (F_s \times 0.005) \tag{6.1}$$

$$\beta_t = 0.1 \times e^{-5 \times |M_\phi - T_d|} \tag{6.2}$$

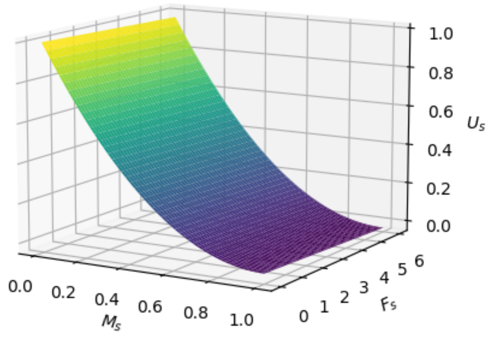
$$U_t = \beta_t + \frac{\sum_{s \in t} L_s \times U_s}{n_t} \tag{6.3}$$

Figure 6.1a shows the resulting rewards for difficulty match $|M_\phi - T_d|$, and figure 6.1b shows how the skill utilities are affected by skill mastery M_s and skill training frequency F_s.

Furthermore, the learning efficiency L_s configuration introduced in subsection 5.2.2 needs to be determined. This configuration indicates the effectiveness of a specific type of tile in training a particular skill, and the mapping is estimated based on empirical experience. A comprehensive overview of these values can be found in table 6.6.



(a) Difficulty Alignment Reward



(b) Skill Utilities

Figure 6.1: Equations

Tile	**Speed Limit**	**Risky Pedestrian**	**Driving Curves**	**Intersection**	**Roundabout**	**New Start**	**Yielding Rules**	**Gap Chance**	**Traffic Lights**	**Highway Driving**	**Overtake**	**Queue Driving**	**Emergency Brake**	**Tunnel Driving**	**Risky Animal Handling**
Straight	0.1	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Straight Crosswalk	0.1	0.6	-	-	-	-	-	-	-	-	-	-	-	-	-
Straight Curved	0.1	-	0.6	-	-	-	-	-	-	-	-	-	-	-	-
Turn	0.1	-	0.6	-	-	-	-	-	-	-	-	-	-	-	-
Turn Curved	0.1	-	0.6	-	-	-	-	-	-	-	-	-	-	-	-
T-Intersection	0.1	-	-	1.0	-	-	-	-	-	-	-	-	-	-	-
X-Section	0.1	-	-	1.0	-	-	-	-	-	-	-	-	-	-	-
Roundabout	0.1	-	-	-	1.0	-	-	-	-	-	-	-	-	-	-
U-Turn	0.1	-	-	-	-	0.1	-	-	-	-	-	-	-	-	-
T-Intersection Curved	0.1	-	0.5	1.0	-	-	-	-	-	-	-	-	-	-	-
T-Intersection Yield	0.1	-	-	0.9	-	-	1.0	0.5	-	-	-	-	-	-	-
T-Intersection Stop	0.1	-	-	0.9	-	0.5	0.5	-	-	-	-	-	-	-	-
X-Intersection Traffic Lights	0.1	-	-	0.9	-	0.4	-	-	1.0	-	-	-	-	-	-
T-Intersection Traffic Lights	0.1	-	-	0.9	-	0.4	-	-	1.0	-	-	-	-	-	-
X-Intersection Crosswalk	0.1	-	-	1.0	-	0.4	0.4	-	-	-	-	-	-	-	-
Highway	0.1	-	-	-	-	-	-	-	-	1.0	-	-	-	-	-
Advanced Highway	0.1	-	-	-	-	-	-	0.5	-	1.0	-	-	-	-	-
Overtake	0.2	-	-	-	-	-	-	-	-	-	1.0	-	-	-	-
Queue	-	-	-	-	-	-	-	-	-	-	-	1.0	-	-	-
Emergency	0.1	-	0.2	-	-	-	-	-	-	-	-	-	0.9	-	-
Tunnel	0.1	-	-	-	-	-	-	-	-	-	-	-	-	1.0	-
Risky Animal	0.1	-	-	-	-	-	-	-	-	-	-	-	0.4	-	1.0

Table 6.6: Skills Trained by Tiles

Lastly, subsection 5.2.4 introduces the concept of a base tile configuration mapping, where each base tile is linked to a set of dimensions that are used to generate the corresponding chunk. These mappings are created to effectively represent the desired training environment for each base tile. The process of creating these mappings involved testing different parameters to determine the most favorable outcomes. The mappings can be found in table 6.7, and tiles with the same foundational structure, such as *T-Intersection* with and without traffic lights, will share identical dimensions.

BaseTile	**Density**	**Width**	**Depth**
U-Turn	0.5	12	7
T-Intersection	0.7	13	7
X-Section	0.8	12	7
Turn	0.4	11	6
Straight	0.4	3	8
Roundabout	0.8	13	6
Tunnel	1.0	3	6
Emergency	1.0	2	6
Highway	1.0	1	6
Advanced Highway	1.0	11	6
overtake	1.0	3	6
Queue	1.0	3	6
Animal Crossing	0.6	12	6

Table 6.7: BaseTile Configuration

6.3 Experimental Results

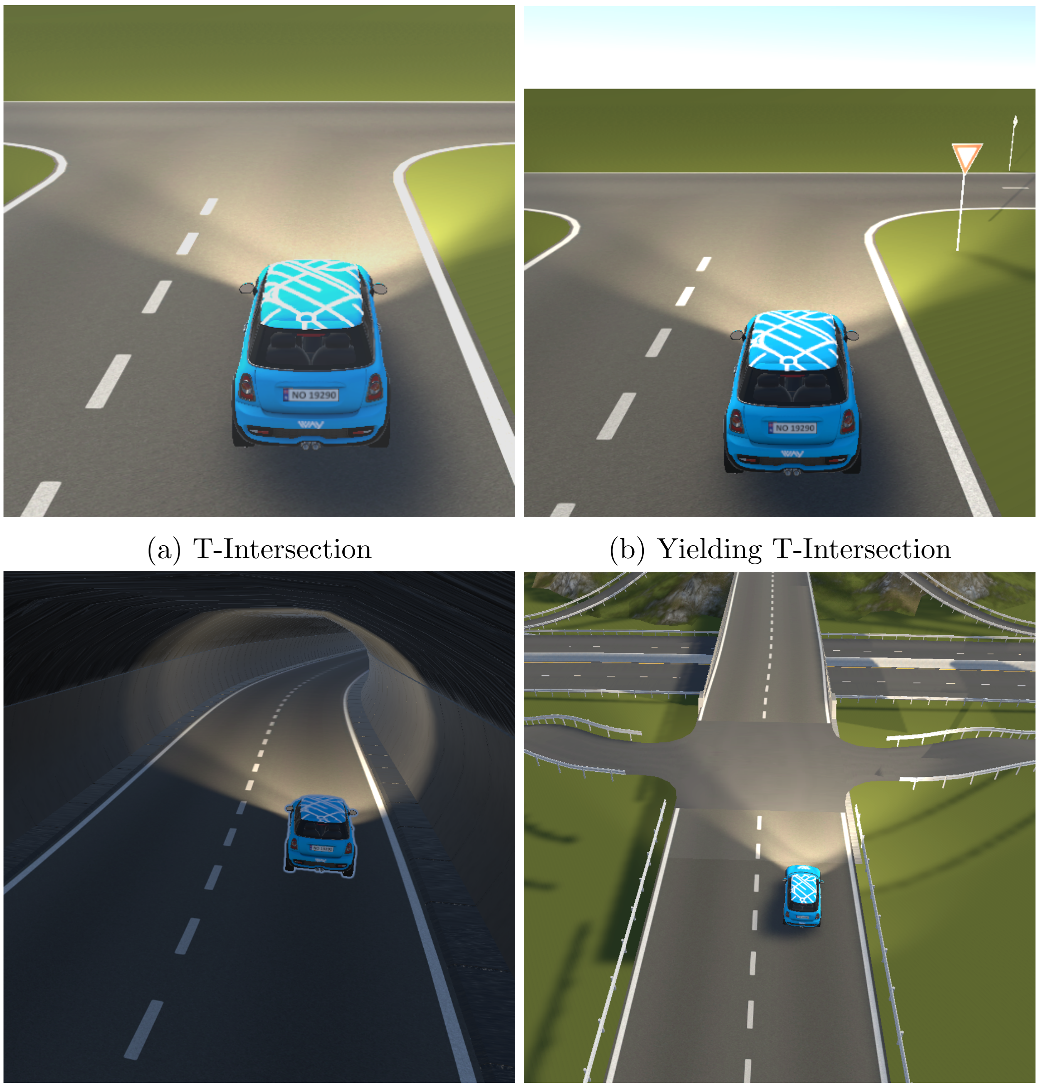
This section aims to present the application of our system when used with the four scenarios described in the previous section as input. To demonstrate the capabilities of the system, we have generated 10 lessons for every scenario. We will start by introducing one example of the generated lessons for each scenario, with screenshots of the lesson with the optimal path visualized, the name of the base tiles the lesson consists of, and some examples of interesting base tiles within the lesson. This section will also analyze and compare the generated lessons to gain insights into their effectiveness for students with different levels of proficiency through statistical analysis.

6.3.1 Generated Lessons

To provide more insight into the lessons that are generated, we here provide one example of a generated lesson from each of the four scenarios that were introduced in section 6.2, together with the optimal path that was calculated for the lesson, and the base tiles that the path takes the student to.

Scenario 1 - Low Proficiency

In *S1 - Low Proficiency*, the lesson seen in figure 6.3 is the first generated lesson with the calculated base illustrated. Figure 6.2 displays screenshots from four of the base tiles in the generated lesson. Here we can see that the lesson consists of a wide variety of tiles, with no traffic or pedestrians added.



(a) T-Intersection

(b) Yielding T-Intersection

(c) Tunnel

(d) Advanced Highway

Figure 6.2: Basetiles generated from S1 - Low Proficiency

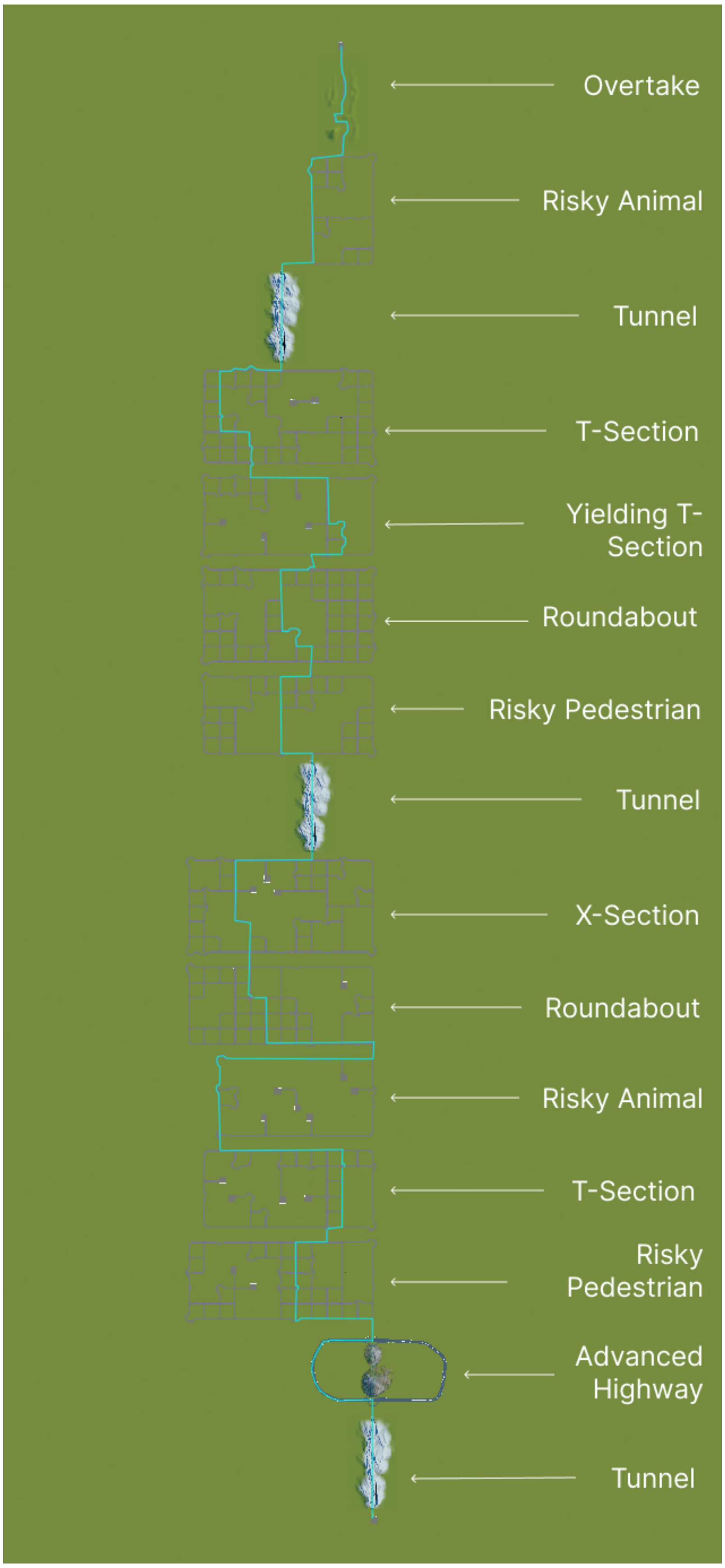


Figure 6.3: Lesson generated from S1 - Low Proficiency

Scenario 2 - Low Proficiency with Strengths

Figure 6.5 visualizes an example of a generated lesson with the optimal path visualized. In figure 6.4, the screenshots of four of the base tiles in the lessons are displayed. We notice that none of the three skills that the student is proficient in, yielding, roundabout, and highway, is trained.

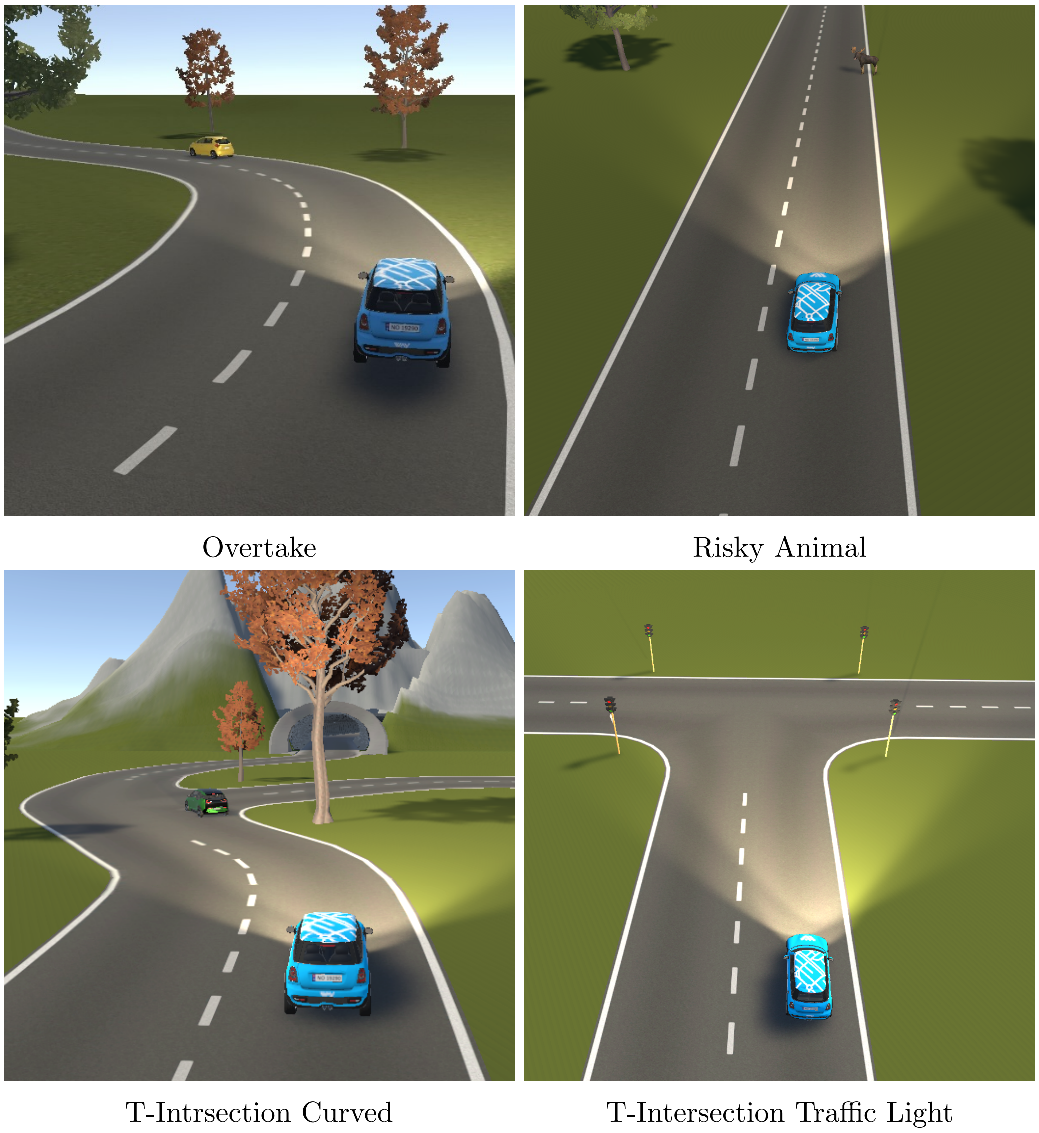


Figure 6.4: Base tiles generated from lesson S2 - Low Proficiency with Strengths

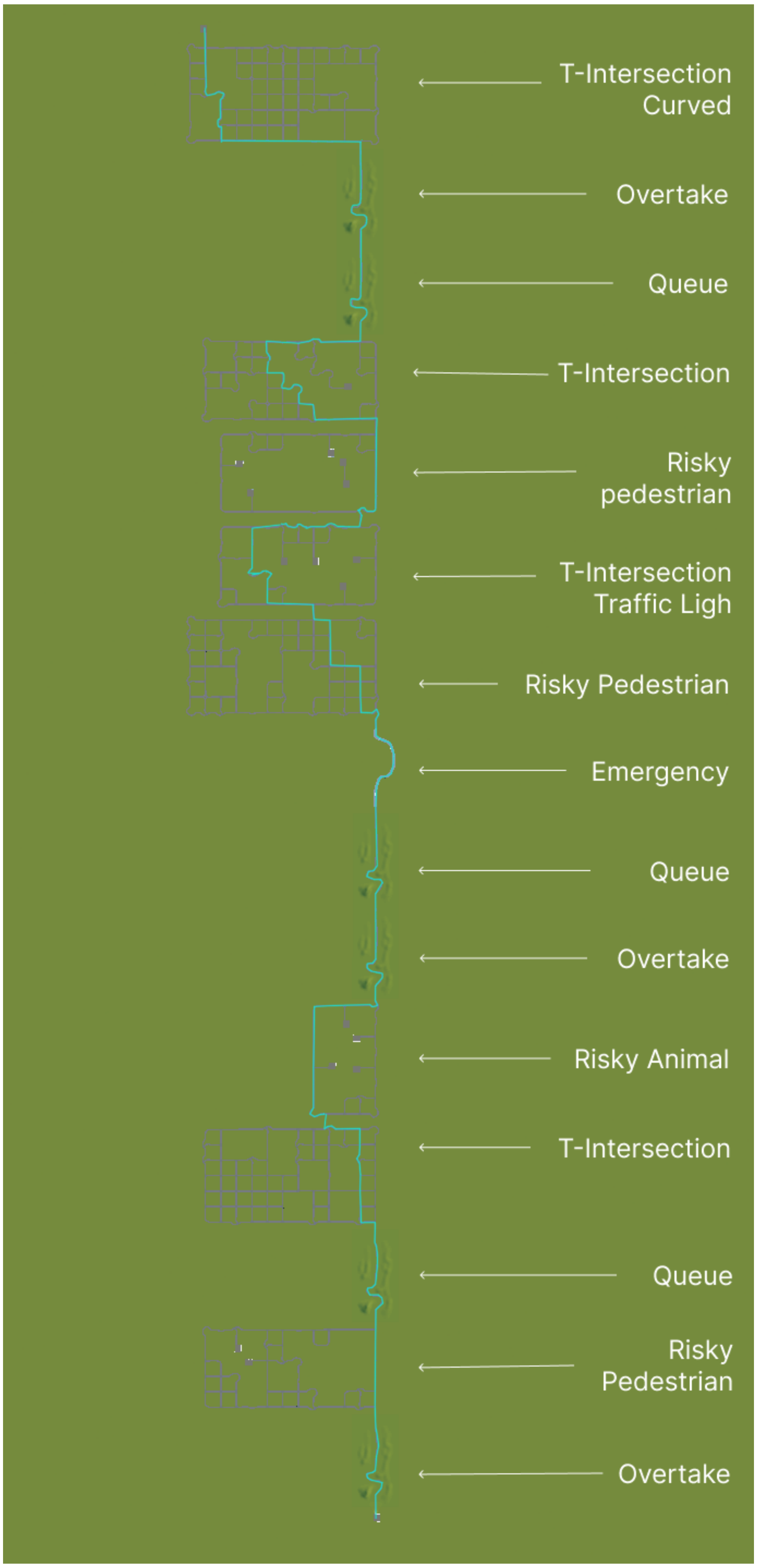
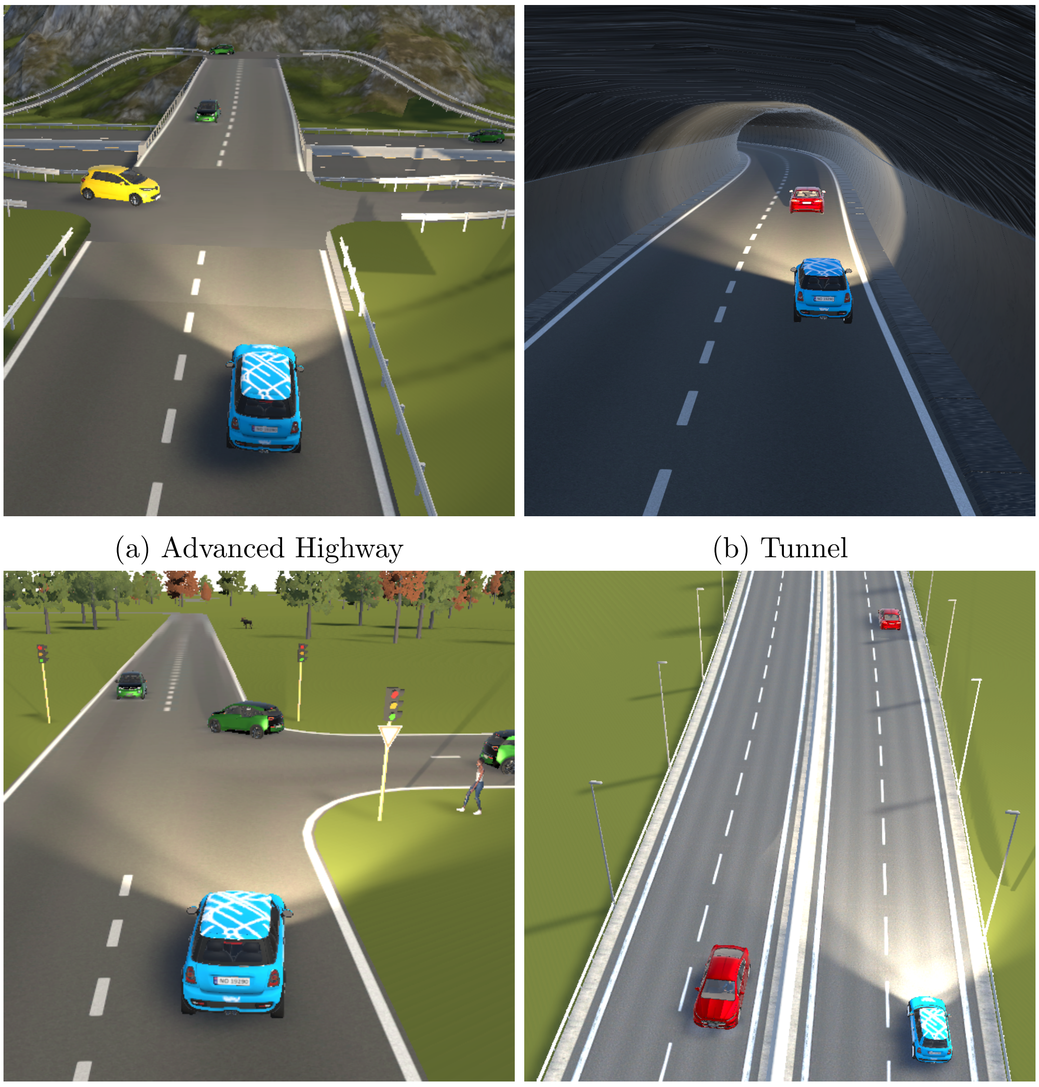


Figure 6.5: Lesson generated from S2 - Low Proficiency with Strengths

Scenario 3 - High Proficiency with Weaknesses

In figure 6.7, the generated lesson is depicted, accompanied by a visualized path to assist the student. The screenshots shown in figure 6.6 provide insights into the traffic situations, which incorporate challenging elements to increase the difficulty level. We notice a high occurrence within the tiles that train the skills the student is inexperienced in. This will be discussed more in section 6.3.2.



(a) Advanced Highway (b) Tunnel

(c) T-Intrsection Traffic Light (d) Highway

Figure 6.6: Base tiles generated from lesson S3 - High Proficiency with Weaknesses

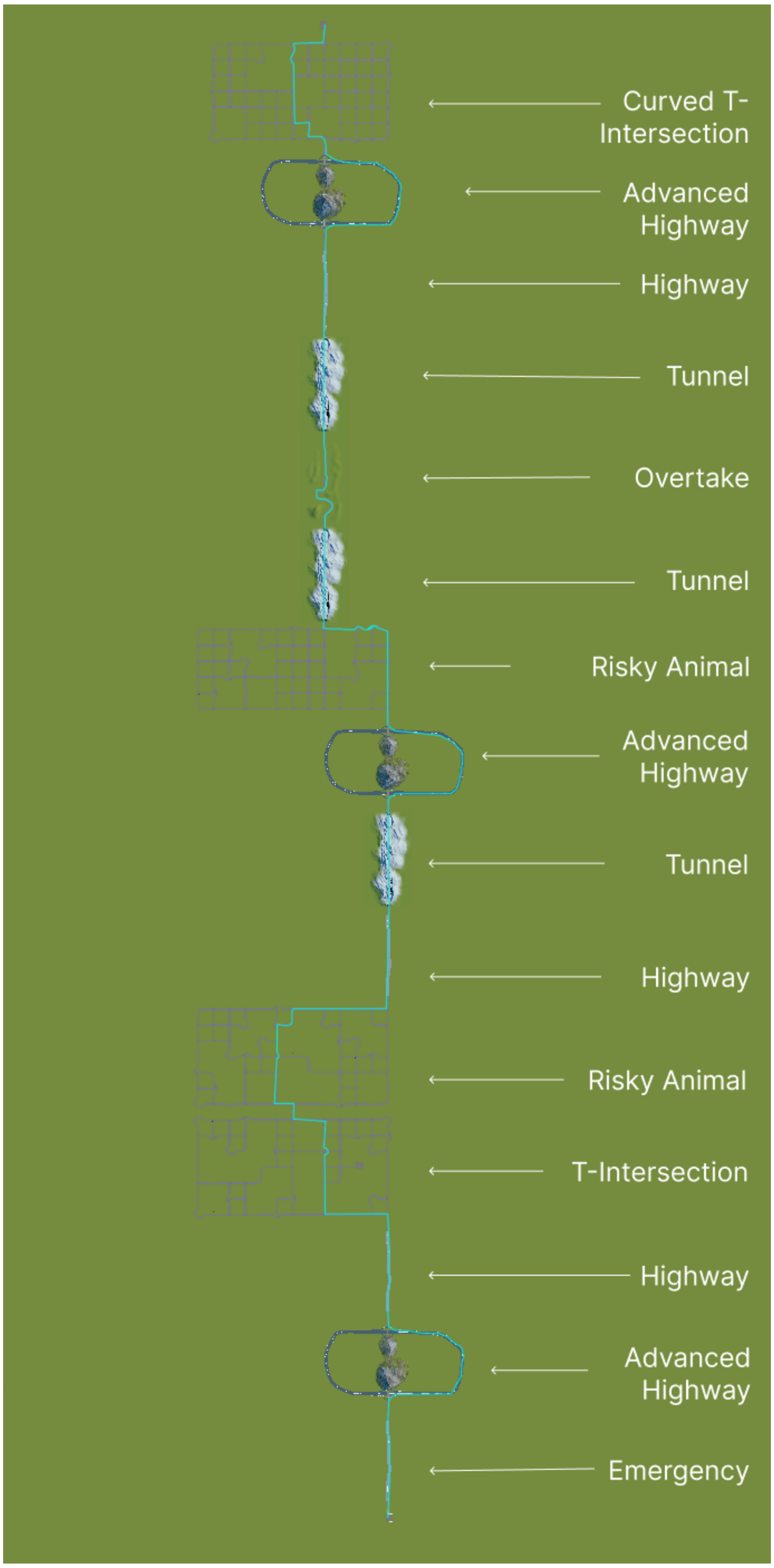
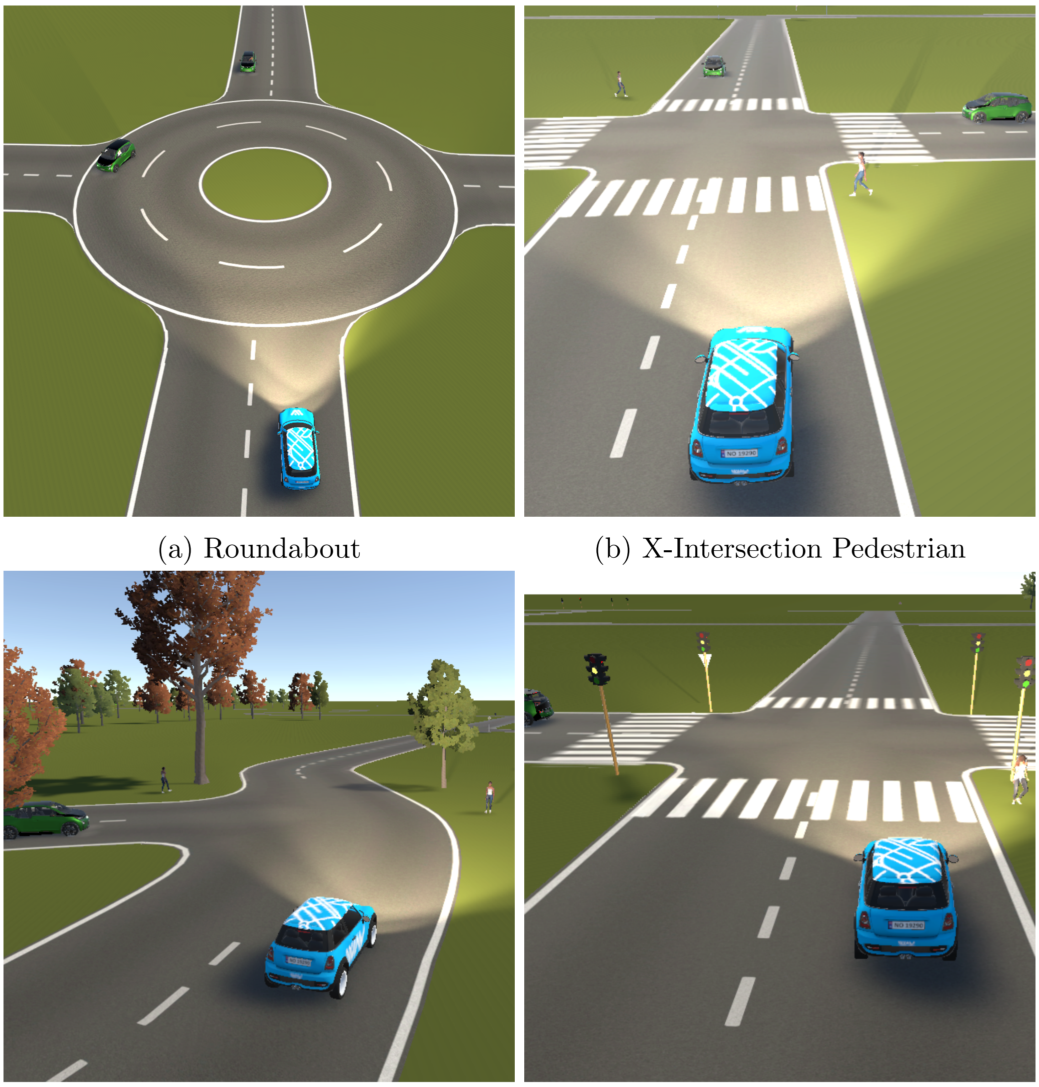


Figure 6.7: Lesson generated from S3 - High Proficiency with Weaknesses

Scenario 4 - High Proficiency

In *S4 - High Proficiency*, the first generated lesson can be observed in figure 6.9. Figure 6.8 showcases a selection of intriguing tiles from the lesson. The Roundabout Tile, depicted in figure 6.8a, serves as one of the base tiles in the lesson, featuring added traffic as a distractor to challenge the high skill level of the student. Figures 6.8c, 6.8d, and 6.8b display base tiles with additional traffic and pedestrians acting as distractors.



(a) Roundabout

(b) X-Intersection Pedestrian

(c) Curved T-Intersection

(d) X-Intersection Traffic Light

Figure 6.8: Base tiles generated from lesson High Proficiency

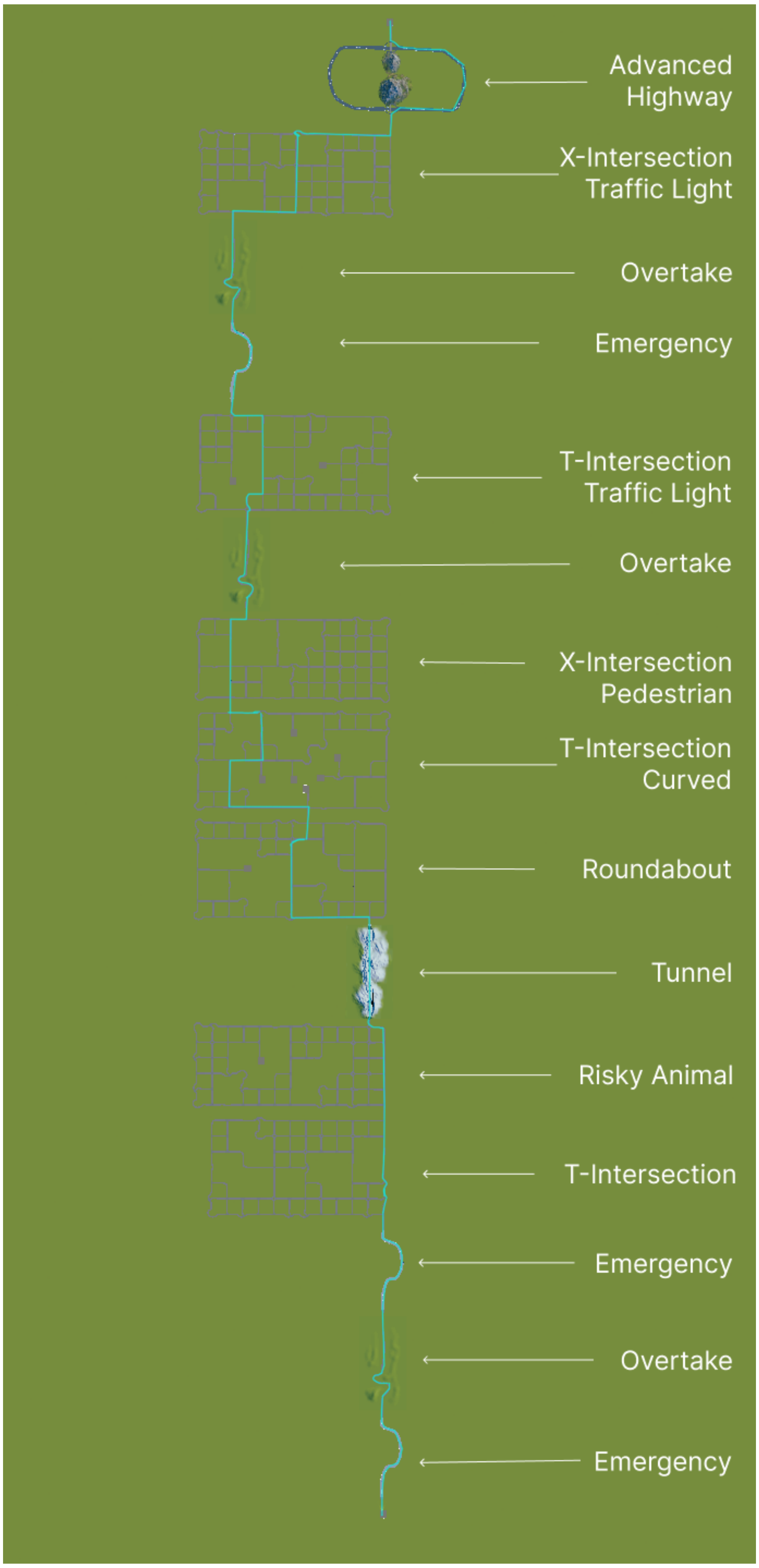


Figure 6.9: Lesson generated from Scenario 4 - High Proficiency

6.3.2 Statistical Analysis

The purpose of the statistical analysis is to gain insights that can address our research questions. We seek to investigate whether the lessons generated by the system exhibit a correlation with the skill levels of the students in order to determine if personalized lessons are being provided. Additionally, we aim to assess the system's ability to generate driving lessons with suitable difficulty levels by analyzing the difficulty levels of the generated lessons and exploring any potential correlations with the skill levels of the students.

Skill Coverage

The objective of figure 6.10 is to illustrate the connection between the skill level of the student and the skills targeted in the generated lessons. Radar charts are utilized to visually represent the skill level of the student in relation to the skills covered in each lesson. To determine the extent of training for each skill, the values described in table 6.6 are summed for every choice of base tile. The cumulative representation is based on the aggregation of data from the 10 lessons generated for each scenario, with the skill coverage normalized to improved visualization.

Based on the radar charts, several key observations can be made regarding the connection between the skill level of the student and the skills trained in the generated lessons. Firstly, it is evident that the intersection skill is heavily favored across scenarios *S1 - Low Proficiency*, *S2 - Low Proficiency with Strengths*, and *S4 - High Proficiency*, with no apparent reason found in the skill levels. Furthermore, in *S4 - High Proficiency*, there is a noticeable bias toward training certain skills despite the skill levels being evenly distributed.

Moreover, there appears to be some correlation between the skill levels of the student and the skills trained in *S2 - Low Proficiency with Strengths* and *S3 - Low Proficiency with Strengths*. For low skill levels, the radar charts demonstrate an increase in the training intensity and a subsequent decrease for high skill levels. For example, the skills *Highway driving*, *Queue driving* and *Tunnel driving* are trained more than all other skills in *S3 - High Proficiency with Weaknesses*, while the skill levels are significantly lower than the rest. This suggests a promising ability to target desired skills while avoiding excessive training in already proficient areas.

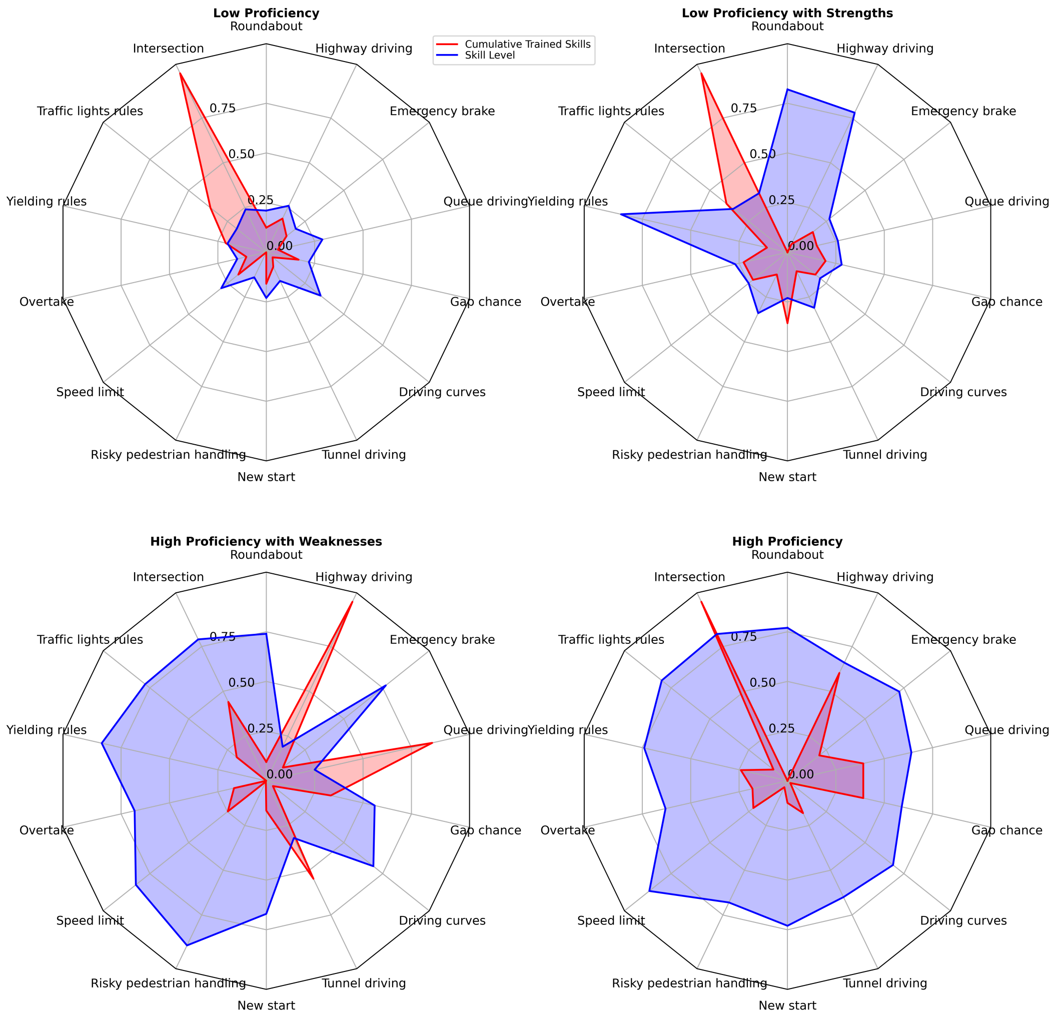
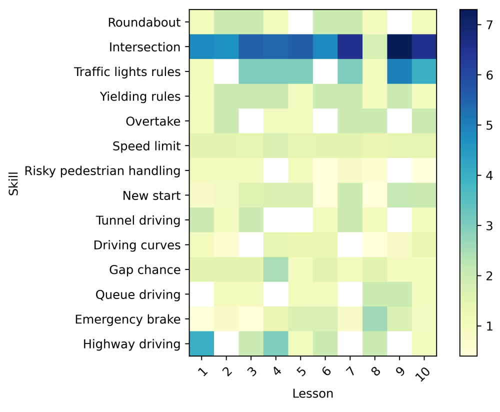


Figure 6.10: Radar diagram

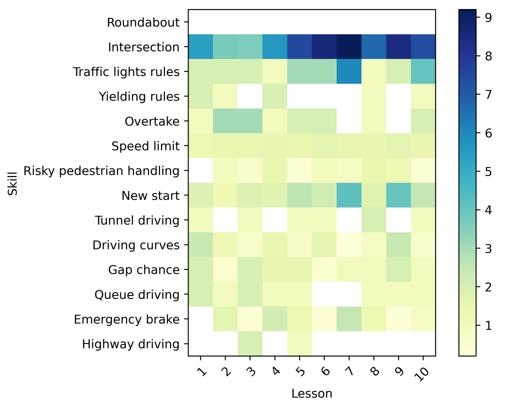
Skill Training Frequency

The heatmap depicted in Figure 6.11 offers a visual representation showcasing the frequency at which each skill is trained in the generated lessons. It is generated by accumulating the total training received by each skill throughout the lessons. This plot provides valuable insights into the patterns and trends observed in skill training, offering a clear view of the distribution and interplay of different skills throughout the lessons.

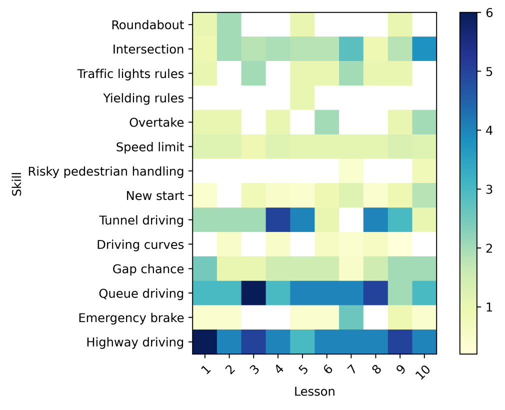
The observations from the plots indicate that the distribution of trained skills varies across the different scenarios. *S3 - High Proficiency with Weaknesses* displays a more focused selection of trained skills, while *S1 - Low Proficiency* and *S4 - High Proficiency* demonstrate a relatively balanced distribution. In *S2 - Low Proficiency* *with Strengths*, there is also a varied distribution of trained skills, but there is a tendency to avoid skills in which the student excels. These patterns align well with the corresponding skill levels, as scenarios *S1 - Low Proficiency*, *S2 - Low Proficiency with Strengths*, and *S4 - High Proficiency* require training across a broader range of skills compared to scenario *S3 - High Proficiency with Weaknesses*. Furthermore, the plot serves to substantiate the assertion of the intersection skill receiving an extensive amount of training, as it consistently appears at a high frequency throughout the lessons.



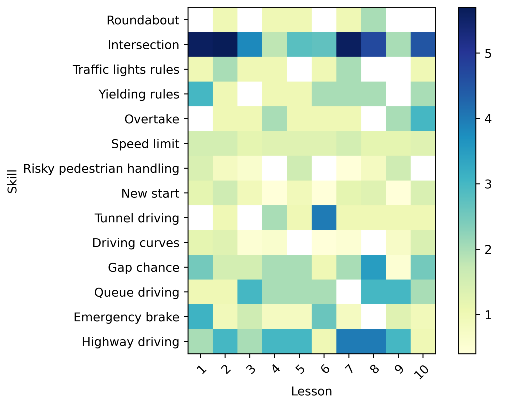
(a) S1 - Low Proficiency



(b) S2 - Low Proficiency with Strengths



(c) S3 - High Proficiency with Weaknesses



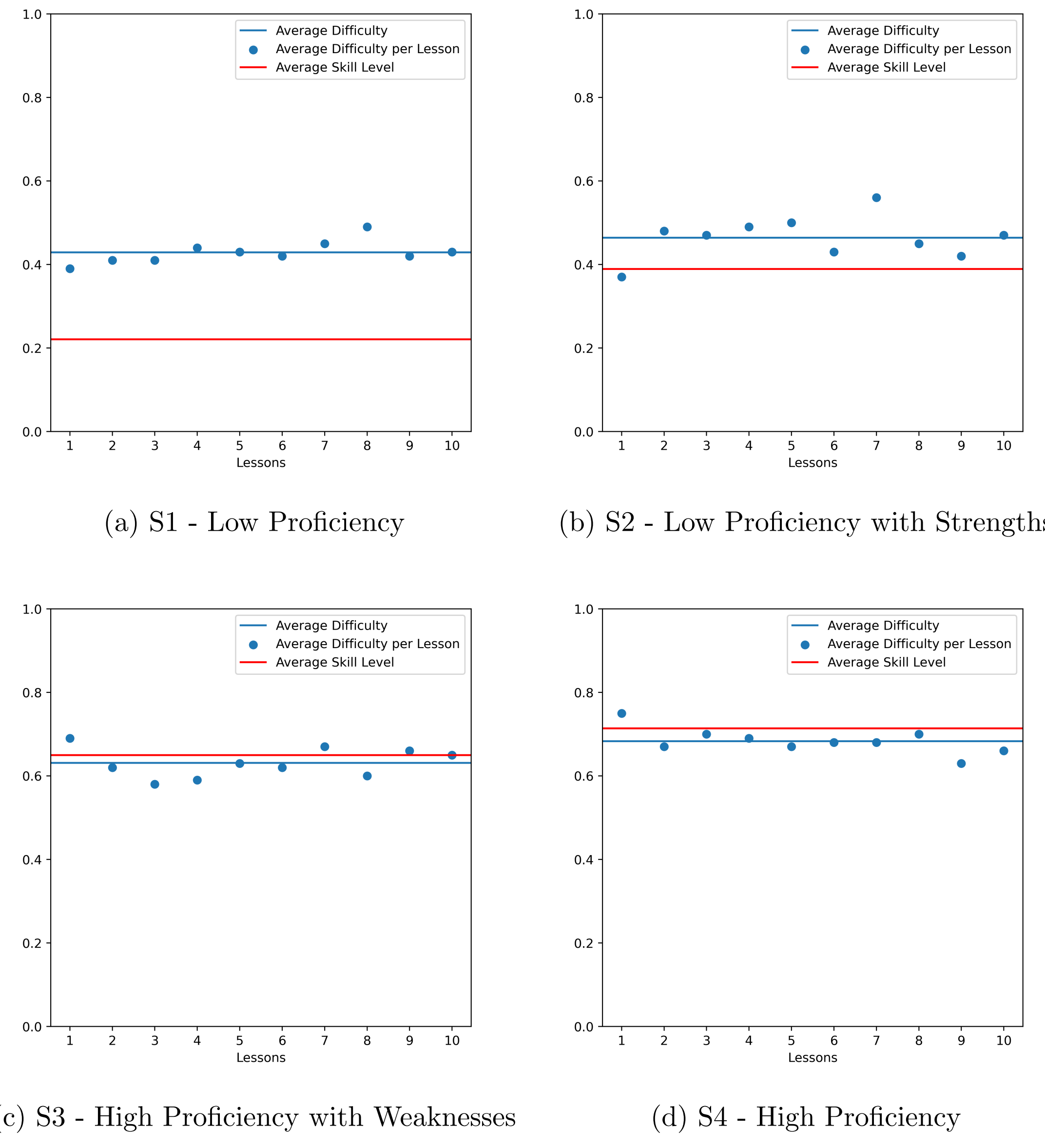
(d) S4 - High Proficiency

Figure 6.11: Heatmap base skills trained

Difficulty Adjustment

The aim of figure 6.12 is to visually represent the relationship between the difficulty level of the generated lessons and the skill levels corresponding to the different scenarios. Each plot in the figure presents the average skill level of the student in the given scenario, the average base tile difficulty per generated lesson, and the average base tile difficulty for the entire simulation. The lesson difficulty is calculated according to the formula 5.4. This visualization provides insights into the alignment between skill levels and difficulty levels, allowing for an assessment of the appropriateness of the generated lessons.

As illustrated in the plots, it is evident that the model effectively adjusts the difficulty level based on the overall skill level. Nevertheless, certain inconsistencies can be observed. Upon comparing the average difficulty and average skill level across the four scenarios, it becomes apparent that the model accurately aligns the generated content with the average skill level in *S2 - Low Proficiency with Strengths* and *S4 - High Proficiency*. However, it encounters difficulty in producing content that is sufficiently easy for *S1 - Low Proficiency*. Furthermore, in the case of *S3 - High Proficiency with Weaknesses*, it would be reasonable to expect a significantly lower average difficulty level, despite the accurate alignment with the average skill level. This expectation arises from the observation that the model succeeds in training the skills in which the scenarios showcase weaknesses, as depicted in figure 6.10. This should ideally result in considerably lower difficulty for those specific tiles compared to the overall skill level.



(a) S1 - Low Proficiency

(b) S2 - Low Proficiency with Strengths

(c) S3 - High Proficiency with Weaknesses

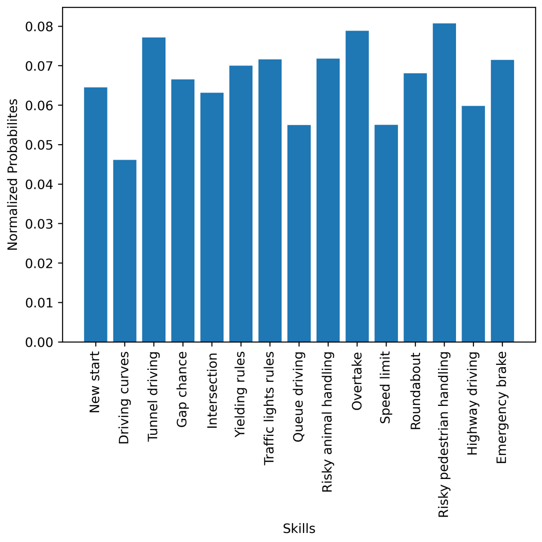
(d) S4 - High Proficiency

Figure 6.12: Difficulty plot

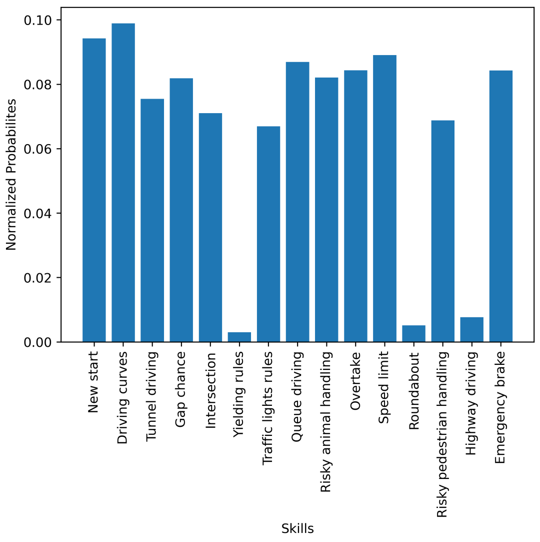
Skill Probabilities

Figure 6.13 presents the normalized cumulative probabilities depicting the likelihood of selecting specific skills. These probabilities are derived by accumulating the skill utilities associated with each chosen base skill across multiple simulations and subsequently normalizing them to ensure a total sum of one. The plot provides insights into the average probability of selecting each base skill whenever a base skill choice is encountered.

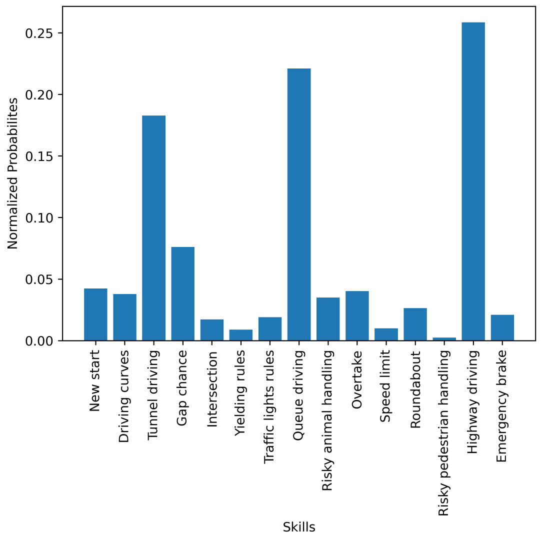
In *S1 - Low Proficiency*, the plot reveals a relatively uniform probability distribution, which aligns with the skill levels observed within the scenario. Furthermore, *S2 - Low Proficiency with Strengths* and *S3 - High Proficiency with Weaknesses* exhibit a preference for selecting skills in which the student performs poorly or avoiding skills in which the student excels, reflecting the underlying skill models. However, the graphical representation for *S4 - High Proficiency* demonstrates an uneven distribution, introducing some inconsistency with the generally comparable skill levels observed within the scenario.



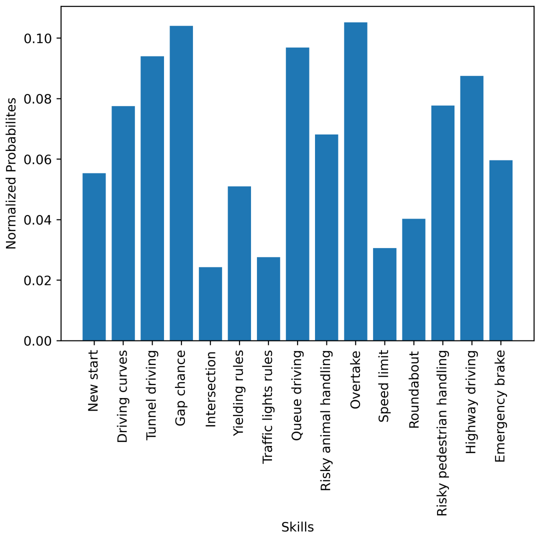
(a) S1 - Low Proficiency



(b) S2 - Low Proficiency with Strengths



(c) S3 - High Proficiency with Weaknesses



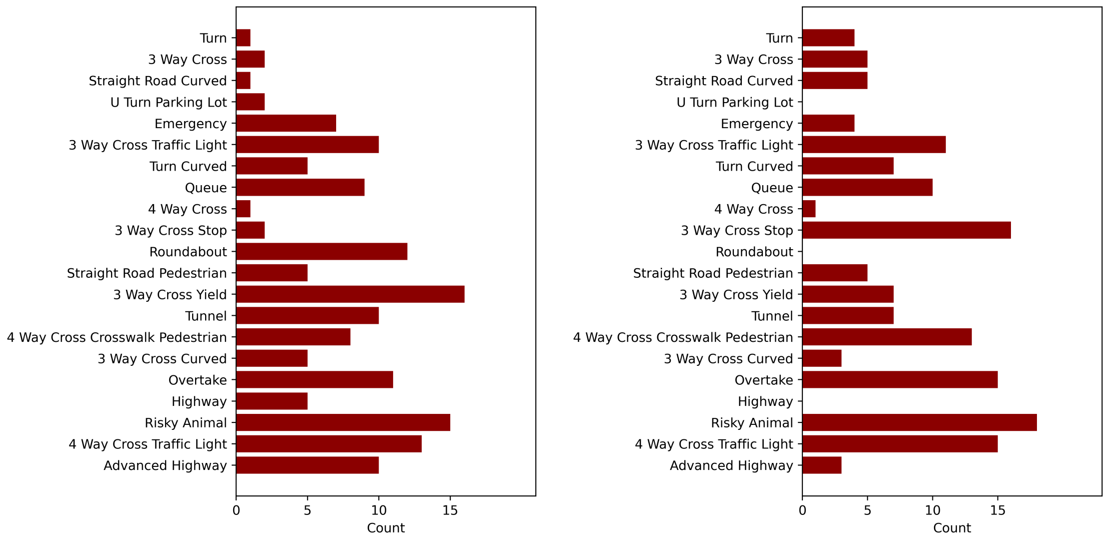
(d) S4 - High Proficiency

Figure 6.13: Skill Utilities plot

Base Tiles

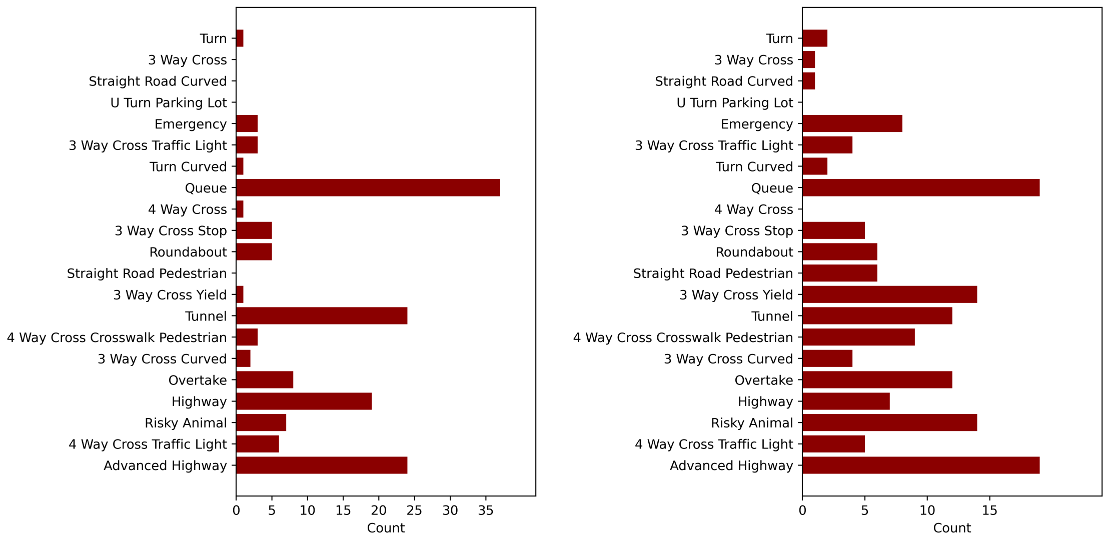
Figure 6.14 illustrates the accumulated base tile choices for each of the four scenarios. The plots depict the frequency of tile selections throughout the simulations, providing insights into the model's tendencies when it comes to choosing base tiles for each scenario. The bar chart allows for a visual comparison of the base tile choices across scenarios, highlighting the varying patterns and tendencies present within the model's decision-making process.

The plots display some noteworthy observations. Firstly, *S1 - Low Proficiency* exhibits the highest degree of diversity in tile selection, as indicated by the range of 15-16 occurrences for the most frequently chosen tile. *S2 - Low Proficiency with Strengths* and *S4 - High Proficiency* showcases a somewhat lower degree of diversity, aligning with the previous observations but deviating from the expectation for *S4 - High Proficiency*. Lastly, *S3 - High Proficiency with Weaknesses* demonstrates the most pronounced bias towards selecting certain tiles, with the most popular tile retrieving over 40 selections. In addition, neither *S1 - Low Proficiency*, characterized by relatively high diversity, nor *S4 - High Proficiency* exhibit a uniform distribution of selected base tiles. This lack of uniform dispersion persists despite the presence of evenly distributed skill levels in both scenarios.



(a) S1 - Low Proficiency

(b) S2 - Low Proficiency with Strengths



(c) S3 - High Proficiency with Weaknesses

(d) S4 - High Proficiency

Figure 6.14: Base Tiles

6.4 Summary

In this chapter, the experimental setup and results aimed at assessing the system's ability to address the research questions have been presented. Four scenarios have been defined to represent students with different skill levels, and the generated lessons for each scenario are presented. Furthermore, a statistical analysis was performed on the resulting data, with all of the key observations summarized in the list below:

Skill Coverage Key observations from the radar chart in figure 6.10:

- **SC1** The intersection skill is prominently favored in *S1 - Low Proficiency*, *S2 - Low Proficiency with Strengths*, and *S4 - High Proficiency*, despite no apparent correlation with skill levels.
- **SC2** *S4 - High Proficiency* exhibits a bias towards training specific skills, disregarding the evenly distributed skill levels.
- **SC3** *S2 - Low Proficiency with Strengths* and *S3 - High Proficiency with Weaknesses* demonstrate a correlation between skill levels and skills trained, with increased training intensity for low skill levels and decreased intensity for high skill levels.

Skill Training Frequency Key observations from the heatmap in figure 6.11:

- **STF1** Scenario 3 shows promise in prioritizing the training of skills in which the student struggles.
- **STF2** *S1 - Low Proficiency*, *S2 - Low Proficiency with Strengths*, and *S4 - High Proficiency* exhibit a more balanced distribution of trained skills, aligning with the skill levels.
- **STF3** The intersection skill receives extensive training across all scenarios, reinforcing the observations from the radar chart in Figure 6.10.

Difficulty Adjustment Key observations from the line chart in figure 6.12:

- **DA1** The model generally adjusts the difficulty level based on the overall skill level, showing promise in its adaptation.
- **DA2** The model encounters challenges in generating sufficiently easy content for *S1 - Low Proficiency.*
- **DA3** *S3 - High Proficiency with Weaknesses* should ideally have a significantly lower average difficulty due to weaknesses in specific skills training, as observed in Figure 6.10.

Skill Utilities Key observations from the bar chart in figure 6.13:

- **SU1** *S1 - Low Proficiency* displays a relatively uniform probability distribution that corresponds to the skill levels.

SU2 *S2 - Low Proficiency with Strengths* and *S3 - High Proficiency with Weaknesses* show a preference for selecting skills in which the student performs poorly and avoiding skills in which the student excels, aligning with the related skill levels.

SU3 *S4 - High Proficiency* exhibits an uneven distribution of skill selection, deviating from the related skill levels.

Base Tiles Key observations from the horizontal bar chart in figure 6.14:

BT1 *S1 - Low Proficiency* demonstrates the highest degree of diversity in tile selection, reflecting the skill levels.

BT2 *S2 - Low Proficiency with Strengths* and *S4 - High Proficiency* display similar degrees of tile diversity, which is lower than *S1 - Low Proficiency* but higher than *S3 - High Proficiency with Weaknesses.*

BT3 *S3 - High Proficiency with Weaknesses* shows the most pronounced bias towards selecting certain tiles that train the skill levels with low proficiency.

BT4 Neither *S1 - Low Proficiency* nor *S4 - High Proficiency*, both with evenly distributed skill levels, exhibit a uniform dispersion of selected base tiles.

Overall, these findings enhance our understanding of the system's capabilities in optimizing learning outcomes in the context of virtual driving education. The observations will be discussed in detail in the next chapter.

Chapter 7

Discussion

This chapter delves into the discussion and evaluation of the implementation and subsequent results. Initially, the research questions will be examined, followed by a comprehensive evaluation of the model. Finally, the validity of the research will be deliberated upon.

7.1 Research Questions Revisited

This section will revisit the research questions in light of the results and implemented model. Reflections on how these affect the satisfiability of the research question will be presented.

RQ1 *How can we facilitate the creation of personalized driving lessons to optimize learning?*

Personalized driving lessons were generated through a system that utilized an individual student's skill levels to determine the selection of tiles. The observations presented in section 6 offer insights as to whether the system succeeds in personalizing content in an intelligent manner.

When analyzing the radar chart in figure 6.10 and heatmap in figure 6.11, observations **SC1** and **STF3** show a disproportionately high frequency of intersection training, which suggests potential biases in the model's generation of traffic scenarios. This could be due to an excessive amount of tiles that train the intersection skill. Of the implemented tiles, 8 out of 24 tiles trained the intersection skill, which likely contributes to the high amount of accumulated training.

Furthermore, in *S1 - Low Proficiency*, which represents a student with a generally low proficiency across all skill levels, the anticipated outcome would be an equal distribution of training for every skill. The observation **STF2** from heatmap in figure 6.11 indicates this by presenting balanced distribution of trained skills. Additionally, observations **SU1** and **BT1** from the bar chart in figure 6.13 displaying the skill probabilities and the horizontal bar chart from figure 6.7 for selected base tiles reinforce this impression further. The results for *S4 - High Proficiency* should display similar attributes, seeing as *S4 - High Proficiency* contains evenly distributed skill levels with high proficiency. There are, however, some irregularities in the observations, indicating otherwise. In the radar chart 6.10, the results exhibit a noticeable bias towards training specific skills as described in observation **SC2**. The skill probability bar chart 6.13 substantiates this observation with **SU3**. A possible explanation can be found in the skill utility formula 6.1. To elevate the probabilities of selecting skills in which a student shows low proficiency, the inverse skill level is raised to a factor w. In the experiments, this factor was set to 2, as seen in table 6.5. The operation succeeds in amplifying the probabilities but is not stable in terms of maintaining the relationship across the scenarios. This is due to the non-linear nature of squaring, which magnifies differences between smaller values to a greater extent than between large values.

The analysis of *S2 - Low Proficiency with Strengths* and *S3 - High Proficiency with Weaknesses* yield promising observations. The skill coverage radar chart in 6.10 displays prowess in training the skills in which *S3 - High Proficiency with Weaknesses* lacks proficiency and avoiding the skills that *S2 - Low Proficiency with Strengths* masters, underscored in observation **SC3**. The heatmap 6.11 and bar chart 6.13 reinforce this belief with observations **STF1**, **STF2**, and **SU2**, all indicating that the model accomplishes personalizing training in a promising manner targeting these two cases.

When addressing the concept of optimized learning, the facilitation of the design principle "Manage the learner's cognitive load" discussed in subsection 3.1 aims to enhance learning by effectively regulating the cognitive load associated with each chunk. While this approach intends to optimize the learning process, determining its effectiveness without conducting comprehensive testing on real-world subjects is challenging.

RQ2 *How can we effectively represent all traffic scenarios and combine them into a driving lesson?*

In chapter 5, a design for a system to construct traffic scenarios was presented. By creating premade tiles in Unity and combining them into chunks, which in turn get combined into a lesson, the system offers a robust representation capable of training a wide variety of skills. When implementing these building blocks, a hybrid model incorporating the concepts of *static chunks* and *dynamic chunks* was created and utilized. The static chunks represented traffic scenarios that spanned over larger areas, such as a highway or a tunnel. These chunks were designed as single-tile chunks, serving as the foundation for the entire traffic situation. Dynamic chunks were implemented as described in chapter 5 by combining multiple tiles into a grid, ensuring all the roads are connected. Ideally, the larger static chunks would also be generated dynamically. This would require significantly higher emphasis on consistency across tiles, as traffic situations such as emergency and overtake rely on a cohesive set of connected tiles to function correctly.

Furthermore, the model's exclusive dependence on human input for defining the tile configurations for size in table 6.7, learning efficiency in table 6.6, and difficulty in table 5.4 could impose certain limitations. A more balanced approach could be achieved by applying machine learning techniques or feedback loops to progressively adjust the configurations based on the outcomes of each driving lesson, thereby optimizing the scenario representation over time.

As briefly mentioned in section 5.2, the focus in this thesis is on the generation of *necessary content*. In the current system, the focus on the environment and the aesthetics of the generated content is minimal due to limited time. This limits the system from being able to represent traffic scenarios such as night driving, urban driving, and driving in weather conditions such as fog or heavy rain.

Finally, when examining the horizontal bar chart depicted in figure 6.14, there are consistently some base tiles favored for each of the four scenarios. Regarding the unevenly distributed skill levels from scenarios *S2 - Low Proficiency with Strengths* and especially *S3 - High Proficiency with Weaknesses*, observations **BT3** and **BT2** could make sense. However, the evenly distributed *S1 - Low Proficiency* and *S4 - High Proficiency* should potentially exhibit a more balanced selection than what was discovered in observation **BT4**. The matter is not as pressing in *S1 - Low Proficiency*, as indicated by the relatively high diversity detected in observation **BT1**, but it still remains noteworthy. A possible explanation for this can be found in the learning efficiency table 6.6. Given that some tiles train a significantly larger number of skills than others, they naturally have a higher cumulative probability of being chosen. Nonetheless, it can be argued that this outcome aligns with real-world dynamics, where such property also holds true.

RQ3 *How can we ensure an appropriate level of difficulty for each specific traffic situation?*

The system strives to achieve appropriate difficulty by assigning specific difficulties to each tile as illustrated in 5.4 while also dynamically adjusting the overall difficulty using distractors. The line chart depicted in figure 6.12 demonstrates that the average difficulty of the simulation adapts in response to the skill levels. This observation **DA1** indicates that the process of generating lessons effectively tailors the tasks to the student's proficiency level, providing more complex and challenging assignments as needed.

However, the model encounters issues when tasked to generate sufficiently easy content for *S1 - Low Proficiency* as described by observation **DA2**. This can be due to a lack of easy enough tiles or simply inaccurate tile initialization described in table 5.4. Additionally, observation **DA3** indicates that *S3 - High Proficiency with Weaknesses* ideally should exhibit significantly lower average difficulty due to its focus on training skills in which the student struggles. One potential explanation for this discrepancy could be that the distractors are determined based on the average difficulty rather than the skill level associated with the selected skill, thereby raising the difficulty for the entire lesson.

The use of these distractors introduced an additional layer of complexity in difficulty assessment, as the presence of distractors could significantly alter the perceived difficulty of a particular traffic situation. The model could potentially be enhanced by incorporating a more sophisticated system for managing distractor-related difficulties, such as an adaptive mechanism that adjusts the frequency and complexity of distractors based on the student's proficiency and performance.

Furthermore, determining the appropriate difficulty level for each traffic situation poses inherent challenges due to the difficulty assessment's highly subjective and context-dependent nature. In the current model, difficulty levels were initially defined based on empirical experience, and further improvements could be made through testing and iterative refinement.

7.2 Model Evaluation

When creating a system that is capable of personalizing virtual driving lessons to optimize learning, several design decisions had to be made. In this section, we will evaluate the decisions made, which include the system requirements, architecture, and various components of the proposed model. The evaluation aims to assess the effectiveness and suitability of these decisions in addressing the research questions and achieving the desired outcomes.

The proposed architecture, based on the Experience Driven Content Generation framework, provides a comprehensive approach to address the system requirements described in section 5.1 and achieve the desired outcomes. The four components of the architecture, namely the Player Experience Model, Content Representation, Content Quality, and Content Generation, form an interconnected system that ensures personalized learning experiences for driving students.

Player Experience Model

The Player Experience Model, utilizing gameplay-based Player Experience Modeling (PEM), captures the skill levels of the student and provides a comprehensive representation of their driving abilities, laying the foundation for the system to answer system requirement **R3**. By considering both specific skills and the general skill level, the model accurately assesses the student's proficiency. By considering both specific skills and the general skill level, the model captures a comprehensive evaluation of the student's driving abilities. In addition to modeling the skill levels of the students, other metrics such as engagement, focus, and learning style could also be modeled and used as a basis for the generation of content, as seen in [38], [14] and [25]. This could lead to higher engagement and an improved learning experience.

Content Representation

The Content Representation component, incorporating tiles, chunks, and lessons, allows for the dynamic creation of game worlds that provide diverse and interconnected traffic situations. The use of chunks facilitates the management of cognitive load and gradual complexity in the learning process. Additionally, the representation of tiles with semantics related to skills, difficulty, and distractors ensures the generation of relevant and engaging content. It provides the necessary flexibility to incorporate all existing traffic situations, adhering to the system requirement **R2**. In the existing representation, each distinct traffic scenario is created as a separate tile. For instance, a t-intersection and a t-intersection with a yielding sign are considered two distinct tiles. A more dynamic approach would be to use the fundamental structure of the tile, such as a t-intersection, as the base and then incorporate different objects onto it based on the semantics of the tile. This alternative method allows for content generation in a more flexible and efficient manner, aligning better with the fulfillment of system requirement **R1**. By adding dynamic and static objects to the fundamental structure of a traffic situation, it would also be easier to adjust the difficulty of the tile to the level of the student, compared to the distractor-induced difficulty implemented to comply with system requirement **R6**.

Content Quality

The incorporation of Content Quality through utility functions and skill mastery assessment ensures the optimization of learning outcomes and promotes replayability. By considering the compatibility between skill levels, tile difficulties, and skill frequencies, the model selects tiles that match the student's skill level and avoids excessive repetition. This enhances the effectiveness of the learning process and maximizes the educational value of the generated content. As mentioned previously, the evaluation functions introduced require several manually tuned weights. These manually tuned weights could introduce unwanted biases.

Content Generation

Content Generation, guided by the defined utility functions and algorithms, combines the available tiles and chunks in a personalized manner. The algorithm for generating lessons strategically selects tiles based on the student's skill level and skill frequencies. By employing this process, the generated content is carefully tailored to match the student's abilities, offering practical skill training that is both engaging and appropriately challenging. Still, the generation lacks somewhat in terms of ordering the chunks to ensure an incremental learning experience. The model incorporates interleaving when selecting base tiles, but an optimal relationship between the different tiles is not explored. Furthermore, due to skill level inference after the lesson ends, the current model utilizes offline generation. However, an online approach, where the lessons could be dynamically adjusted based on the student's performance during the lesson, would have been more in tune with system requirement **R5**. With the online generation of chunks, content could be generated based on the newest skill levels that account for how the student performed in the previous traffic situations, leading to a faster adaptation to the student's skill level.

To promote variation, the system utilizes *stochastic* generation. By incorporating randomness, it introduces diversity and unpredictability into the generated outputs. This variability allows for the exploration of different possibilities and the generation of unique, customized results. While stochastic generation can enhance diversity and unpredictability in the generated outputs, it can also result in a loss of control. The randomness introduced may lead to unexpected or undesired outcomes that may not align with the intended objectives or constraints of the system. This loss of control can pose challenges in ensuring the generated content meets specific requirements or aligns with predefined guidelines. Therefore, careful consideration and fine-tuning of the stochastic generation process are necessary to strike a balance between desired variation and maintaining control over the generated content.

While the proposed system relying on constructive generation successfully generates personalized driving lessons, the exploration of other methodologies, such as search-based generation, could potentially improve the system's efficiency and effectiveness. For example, the use of evolutionary algorithms and particle swarm optimization could introduce a higher degree of variability and potential for optimization, as seen in [31], [11], [14] and [21].

Pathfinder

In section 5.3, an optimal path is calculated to ensure that the student approaches the lesson in an efficient manner. The inclusion of this path serves the purpose of guiding the student to the base tile within a chunk and through the other chunks of the lesson, thereby attempting the fulfillment of system requirement **R4**. The path succeeds in guaranteeing that the student encounters desirable traffic situations. Still, the order in which the base tiles are visited depends on distance rather than optimized learning, subsequently lacking somewhat compared to the desired outcome. Additionally, the fact that the student has to drive through non-optimal tiles within a chunk could be an indicator that the content generation is suboptimal and that the pathfinder is a required tool to compensate for that limitation. Nevertheless, in this context, the pathfinder still serves an essential function in ensuring that the student has satisfactory progress through the lesson. Furthermore, even though not tested in this research, the pathfinder algorithm facilitates invoking the path generator during run-time with different base tiles, thereby enabling online alteration of desirable traffic situations.

In summary, the decisions made in the Model chapter demonstrate a thoughtful and comprehensive approach to addressing the research questions and achieving the desired outcomes. The proposed system requirements, architecture, and components provide a solid foundation for building a model that delivers personalized and compelling learning experiences in virtual driving simulators.

7.3 Validity Evaluation

The experiment conducted aimed to investigate the suitability of automatic personalized content generation as an alternative to static lesson generation in the context of virtual driving simulators. To verify the validity of the research, the experiment is reviewed in light of the four validity threats introduced by Wohlin et al. [41], namely conclusive validity, internal validity, construct validity, and external validity.

7.3.1 Conclusive Validity

Conclusive validity pertains to the extent to which the causal relationships between experiments and observations are warranted. In the context of this project, conclusive validity is supported by the valuable insights obtained from the analysis of the generated plots. These plots, derived from a large number of generated lessons, provide a robust foundation for drawing conclusions and identifying patterns and trends in the data.

However, it is essential to recognize that conclusions drawn from the experiment are limited to the specified scenarios. The algorithm's results could yield other observations when tested on different scenarios or on real-life subjects. In addition to a narrow test base, the scenarios might not represent real-life students in a realistic manner. Seeing as the scenarios are developed to test constructed edge cases, this might be unrepresentative of the skill levels that the skill model actually provides. For instance, low proficiency is in this thesis characterized by skill levels in the range of 0.14 to 0.35, but this could be an unrealistic depiction of low proficiency. Consequently, caution should be exercised when generalizing the findings to broader populations.

7.3.2 Internal Validity

Internal validity, although not the most relevant for this project, refers to whether the outcomes in an experiment can be attributed to the specific interventions or modifications implemented. Most threats to internal validity relate to uncontrollable interference or disturbance when gathering results. The experiment conducted in this research relied on running an identical algorithm on predefined scenarios, thereby eliminating most of the threats to internal validity. However, overfitting by excessively tailoring the model to suit the designed scenarios is a possible limitation that needs to be validated.

7.3.3 Construct Validity

Construct validity refers to the extent to which the concepts, theories, or constructs used in the experiment accurately represent the underlying phenomena being studied. In the context of this experiment, several potential challenges could impact the construct validity. Firstly, it is demanding to verify appropriate difficulty without testing on real-world subjects. In addition, it is challenging to conclude that constitutes a greater perceived difficulty for a given individual, as some students might find certain elements more difficult than others. Furthermore, setting the correct initialization for tile difficulty, learning efficiency, and distractor levels should be done more intelligently and subsequently authenticated. In summation, to enhance the construct validity of the research, a more exhaustive approach is required.

7.3.4 External Validity

External validity addresses the concept of generalizing the causal relationships to real-world populations. Considering the experiment is conducted in collaboration with WAY, utilizing their existing skill model and creating the algorithm in Unity compatible with the existing solution, generalization to the real world should be a relatively unchallenging obstacle. However, as mentioned earlier, the causal relationships would benefit from a more exhaustive experiment before being introduced in a commercial environment.

Chapter 8

Conclusion and Future Work

This chapter provides the reader with a conclusion and suggestions for future work. The conclusion is structured around the research questions and the project's validity, whereas the section related to future work proposes some possibilities for further research identified throughout the thesis.

8.1 Conclusion

This master thesis proposes an automatic personalized content generation model compatible with WAY's unity-based simulator. The model adapts content based on an individual student's skill levels, aiming to optimize learning by providing students with appropriate difficulty and tailored traffic situations. Implementing this model, we wanted to investigate the research objective and subsequent research questions described in section 1.2.

In chapter 7, the research questions are revisited and evaluated in light of the model design and key observations from the results chapter. Regarding **RQ1**, the results provide strong indications of successful personalization of driving lessons. There are a couple of irregularities in the observations relating to uneven training for *S4 - High Proficiency* and an excessive amount of intersection training. Still, both have straightforward explanations and do not undermine the remaining promise shown. However, when addressing the optimization of learning, the experiment is inconclusive, as this would require thorough tests conducted with real-world subjects.

Concerning **RQ2**, the result proves the value of a semantic-based representation in this research context, although the distinction between dynamic and static chunks is suboptimal. Moreover, the representations' dependence on human input for configuration leaves room for improvement. However, despite some weaknesses, the representation successfully depicts all the traffic situations we saw fit to include, thereby answering the research question in a satisfactory manner.

The final research question **RQ3** pertains to ensuring appropriate difficulty levels relative to an individual's skill levels. The results demonstrate a clear correlation between the student's skill levels and the difficulty corresponding with the generated content. Nevertheless, the statistical analysis brings to light certain challenges concerning the impact of distractor-induced difficulty and the subjective configuration of tile difficulties and learning efficiency. In order to enhance the precision of the relationship, it is necessary to incorporate a wider variety of tiles and devise a more intelligent method for deciding configurations.

Upon evaluating the experiment's validity in section 7.3, the findings indicate satisfactory levels of conclusive, internal, and construct validity, despite elaboration on some possible threats for each of them. However, there are certain limitations pertaining to construct validity that are suboptimal. These limitations are interconnected with other challenges, highlighting the necessity for comprehensive real-world testing to drive further improvements.

8.2 Contributions

In this research, we have made some noteworthy contributions. Firstly, a literature review on procedural content generation and personalized learning in the context of virtual simulators gives a detailed outline of the research field. Furthermore, a semantic representation for depicting traffic situations is provided. This representation is mainly designed from the perspective of driving lessons but can be utilized in a wider context. Moreover, a model and algorithm for the automatic generation of personalized content is introduced. This model is compatible with WAY's Unity-based simulator and skill model, thereby shortening the deployment time. Included in the algorithm is also a method for reviewing the content in relation to a student's skill levels, which is possibly applicable to other domains. In addition, the thesis offers an analysis of the generated content and reviews results against the research questions presented. Lastly, proposals for future work based on this research are suggested.

8.3 Future Work

For future work, there are several interesting approaches to continue this research. Firstly, since the model is implemented in collaboration with WAY, hereby ensuring compatibility with existing technology, this facilitates a great opportunity to test the model on real-world subjects. This could aid in fine-tuning the configurations proposed for the presented model and potentially uncover additional insights. Moreover, a comprehensive test in collaboration with WAY could open for generalizing the content generation algorithm to real-world applications.

As touched upon in the discussion, the system only generates *necessary content*. In the future, implementing content generation that can adjust the environments from rural to urban, and implement different weather and time of day, could add another dimension to the realness of the lessons.

Furthermore, some alternative approaches can be reviewed. The literature in section 4 established a broad application of search-based algorithms for personalized procedural content generation. Such methods could be applicable to this research as well and possibly yield better results. Another intriguing perspective to consider in this research is the concept of online generation. Due to limitations in the timing of the skill inference model, this was not applicable in this particular research but would likely yield intriguing findings. This could be attempted by either further exploration of the papers presented in related work proposing online methods or by utilizing the functionality of the pathfinder algorithm to update the path during a lesson while the student receives live feedback on performances.

Finally, there is potential for further exploration in the representation of traffic scenarios. If a greater level of consistency across several tiles could be attained, it could open for the removal of static chunks, as well as the possibility of utilizing the fundamental road structure of a tile as the base and dynamically adjusting the remaining content to suit the student's needs.

Bibliography

[1] Way AS. *Way Traffic School - About us*. https://way.no/en/om-oss/. (Accessed on 12/09/2022).

[2] Ronald Baecker and William Buxton. 'Readings in human-computer interaction: A multidisciplinary approach'. In: (1987).

[3] Sander Bakkes et al. 'Challenge balancing for personalised game spaces'. In: *2014 IEEE Games Media Entertainment*. 2014, pp. 1–8. DOI: 10.1109/GEM.2014.7047971.

[4] Jan vom Brocke, Alan Hevner and Alexander Maedche. 'Introduction to Design Science Research'. In: Sept. 2020, pp. 1–13. ISBN: 978-3-030-46780-7. DOI: 10.1007/978-3-030-46781-4_1.

[5] Francesco Calimeri et al. 'Answer Set Programming for Declarative Content Specification: A Scalable Partitioning-Based Approach'. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 11298 LNAI (2018). Cited by: 3, pp. 225–237. DOI: 10.1007/978-3-030-03840-3_17.

[6] Darryl Charles and Michaela Black. 'Dynamic player modeling: A framework for player-centered digital games'. In: *Proc. of the International Conference on Computer Games: Artificial Intelligence, Design and Education*. 2004, pp. 29–35.

[7] Chih-Ming Chen, Hahn-Ming Lee and Ya-Hui Chen. 'Personalized e-learning system using Item Response Theory'. In: *Computers Education* 44.3 (2005), pp. 237–255. ISSN: 0360-1315. DOI: https://doi.org/10.1016/j.compedu.2004.01.006.

[8] E.W. DIJKSTRA. 'A Note on Two Problems in Connexion with Graphs.' In: *Numerische Mathematik* 1 (1959), pp. 269–271.

[9] Dajana Dimovska et al. 'Towards procedural level generation for rehabilitation'. In: *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*. 2010, pp. 1–4.

[10] John Dunlosky et al. 'Improving students' learning with effective learning techniques: Promising directions from cognitive and educational psychology'. In: *Psychological Science in the Public Interest, Supplement* 14.1 (2013). Cited by: 1539, pp. 4–58. DOI: 10.1177/1529100612453266.

[11] Corrado Grappiolo et al. 'Towards player adaptivity in a serious game for conflict resolution'. In: Cited by: 31. 2011.

[12] Frank Greitzer, Olga Kuchar and Kristy Huston. ‘Cognitive science implications for enhancing training effectiveness in a serious gaming context’. In: *ACM Journal of Educational Resources in Computing* 7 (Nov. 2007). DOI: 10.1145/1281320.1281322.

[13] Linda Marie Harasim et al. *Learning networks: A field guide to teaching and learning online*. MIT press, 1995.

[14] Ken Hartsook et al. ‘Toward supporting stories with procedurally generated game worlds’. In: Cited by: 71. 2011.

[15] Mu-Jung Huang, Hwa-Shan Huang and Mu-Yen Chen. ‘Constructing a personalized e-learning system based on genetic algorithm and case-based reasoning approach’. In: *Expert Systems with Applications* 33.3 (2007), pp. 551–564. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2006.05.019.

[16] Kenneth Hullett and Michael Mateas. ‘Scenario generation for emergency rescue training games’. In: Cited by: 33. 2009.

[17] Gwo-Jen Hwang et al. ‘A Heuristic Algorithm for planning personalized learning paths for context-aware ubiquitous learning’. In: *Computers and Education* 54.2 (2010). Cited by: 158, pp. 404–415. DOI: 10.1016/j.compedu.2009.08.024.

[18] W.S. Jäger et al. ‘A Bayesian network approach for coastal risk analysis and decision making’. In: *Coastal Engineering* 134 (2018). Cited by: 56; All Open Access, Green Open Access, pp. 48–61. DOI: 10.1016/j.coastaleng.2017.05.004.

[19] Yannakakis Georgios N. Togelius Julian. ‘Experience-Driven Procedural Content Generation’. In: *IEEE TRANSACTIONS ON AFFECTIVE COMPUTING, VOL. 2* (2011). Cited by: 355.

[20] Fedwa Laamarti, Mohamad Eid and Abdulmotaleb El Saddik. ‘An overview of serious games’. In: *International Journal of Computer Games Technology* 2014 (2014).

[21] Ao Li et al. ‘SceGene: Bio-Inspired Traffic Scenario Generation for Autonomous Driving Testing’. In: *IEEE Transactions on Intelligent Transportation Systems* 23.9 (2022). Cited by: 1, pp. 14859–14874. DOI: 10.1109/TITS.2021.3134661.

[22] Peter H Lindsay and Donald A Norman. *Human information processing: An introduction to psychology*. Academic press, 2013.

[23] Ricardo Lopes and Rafael Bidarra. ‘A semantic generation framework for enabling adaptive game worlds’. In: Nov. 2011, p. 6. DOI: 10.1145/2071423.2071431.

[24] Ricardo Lopes and Rafael Bidarra. ‘Adaptivity challenges in games and simulations: a survey’. In: *IEEE Transactions on Computational Intelligence and AI in Games* 3.2 (2011), pp. 85–99.

[25] Ricardo Lopes, Tim Tutenel and Rafael Bidarra. ‘Using Gameplay Semantics to Procedurally Generate Player-Matching Game Worlds’. In: PCG’12. Raleigh, NC, USA: Association for Computing Machinery, 2012, pp. 1–8. ISBN: 9781450314473. DOI: 10.1145/2538528.2538531.

[26] Regan L Mandryk and M Stella Atkins. 'A fuzzy physiological approach for continuously modeling emotion during interaction with play technologies'. In: *International journal of human-computer studies* 65.4 (2007), pp. 329–347.

[27] David Michael and Sande Chen. 'Serious Games: Games That Educate, Train, and Inform'. In: (Jan. 2006).

[28] George A. Miller. *The magical number seven, plus or minus two: Some limits on our capacity for processing information.* US: American Psychological Association, 1956.

[29] Harold Pashler et al. 'Learning styles concepts and evidence'. In: *Psychological Science in the Public Interest, Supplement* 9.3 (2008). Cited by: 1100, pp. 105–119. DOI: 10.1111/j.1539-6053.2009.01038.x.

[30] Neil Peirce, Owen Conlan and Vincent Wade. 'Adaptive educational games: Providing non-invasive personalised learning experiences'. In: Cited by: 134; All Open Access, Green Open Access. 2008, pp. 28–35. DOI: 10.1109/DIGITEL.2008.30.

[31] Rafael Guerra de Pontes, Herman Martins Gomes and Igor Santa Ritta Seabra. 'Particle swarm optimization for procedural content generation in an endless platform game'. In: *Entertainment Computing* (2022). Cited by: 1.

[32] William L. Raffe et al. 'Integrated Approach to Personalized Procedural Map Generation Using Evolutionary Algorithms'. In: *IEEE Transactions on Computational Intelligence and AI in Games* (2015). Cited by: 8.

[33] Sandro Ropelato et al. 'Adaptive Tutoring on a Virtual Reality Driving Simulator'. In: Nov. 2017.

[34] Noor Shaker, Julian Togelius and Mark J Nelson. *Procedural content generation in games.* Springer, 2016.

[35] Peizhi Shi and Ke Chen. 'Online level generation in Super Mario Bros via learning constructive primitives'. In: Cited by: 8. IEEE Computer Society, 2016.

[36] Gillian Smith. 'Procedural Content Generation, An Overview'. In: *Level Design* (2013). Cited by: 10.

[37] Edward Teng and Rafael Bidarra. 'A Semantic Approach to Patch-Based Procedural Generation of Urban Road Networks'. In: *Proceedings of the 12th International Conference on the Foundations of Digital Games.* FDG '17. Hyannis, Massachusetts: Association for Computing Machinery, 2017. ISBN: 9781450353199. DOI: 10.1145/3102071.3110569.

[38] Julian Togelius, Renzo De Nardi and Simon M. Lucas. 'Towards automatic personalised content creation for racing games'. In: Cited by: 228; All Open Access, Green Open Access. 2007, pp. 252–259. DOI: 10.1109/CIG.2007.368106.

[39] Julian Togelius et al. 'Search-Based Procedural Content Generation: A Taxonomy and Survey'. In: *IEEE Transactions on Computational Intelligence and AI in Games* 3.3 (2011), pp. 172–186. DOI: 10.1109/TCIAIG.2011.2148116.

[40] Claes Wohlin. 'Guidelines for snowballing in systematic literature studies and a replication in software engineering'. In: Cited by: 718; All Open Access, Green Open Access. 2014. DOI: 10.1145/2601248.2601268.

[41] Claes Wohlin et al. *Experimentation in software engineering.* Vol. 9783642290442. Cited by: 2798; All Open Access, Green Open Access. 2012, pp. 1–236. DOI: 10.1007/978-3-642-29044-2.

[42] Yuzhong Zhang, Guixuan Zhang and Xinyuan Huang. 'A Survey of Procedural Content Generation for Games'. In: Cited by: 1. Institute of Electrical and Electronics Engineers Inc., 2022.

NTNU

Norwegian University of
Science and Technology