A Virtual Driving Instructor that Generates Personalized Driving Lessons Based on Student Skill Level

J. Fredrik R. Bjørnland[*1], **Yrjar Gedde**[*1], **Johannes Rehm**[1,2], **Irina Reshodko**[1,2], **Odd Erik Gundersen**[1]

[1]Norwegian University of Science and Technology, Trondheim, Norway
[2] Way AS, Trondheim, Norway

Abstract

Currently, students acquire driving skills by practicing in actual traffic conditions and through direct interactions with an instructor. While one-on-one interactions could be tailored to a student's learning style and skill level, making them effective for learning, one-on-one interactions are also inefficient, potentially costly, and not standardized with limitations on which traffic situation can be safely taught. For these exact reasons Way AS has developed and commercially deployed a virtual driving instructor that educates students in high-fidelity simulators. In this paper, we present a module, the *Lesson generator*, that extends the virtual driving instructor to generate personalized lessons for individual students with the goal to practice in a focused and deliberately fashion the skills that need practice for the students to become proficient drivers. A case study is presented, and the path to deployment is discussed.

Introduction

Currently, students acquire driving skills by practicing in actual traffic conditions and through direct interactions with an instructor. While one-on-one interactions could be tailored to a student's learning style and skill level, making them effective for learning, one-on-one interactions are also inefficient, potentially costly, and not standardized with limitations on which traffic situation can be safely taught. The inefficiency stems from the dual dependency on the availability of an instructor and the accessibility to appropriate traffic situations. Driving in a bustling city, with its myriad of cars, pedestrians, and ongoing road works, cannot be simulated without access to such an environment. In certain areas, essential traffic elements like roundabouts or traffic lights might be absent. Moreover, practicing specific traffic situations in the real world could pose risks to the student, instructor, and other road users. Consequently, students often learn to drive without firsthand experience in managing hazardous situations, like animals on the road or pedestrians suddenly crossing. Finally, many traffic situations cannot be deliberately practiced and repeated over a short period of time so that handling these situations become second nature.

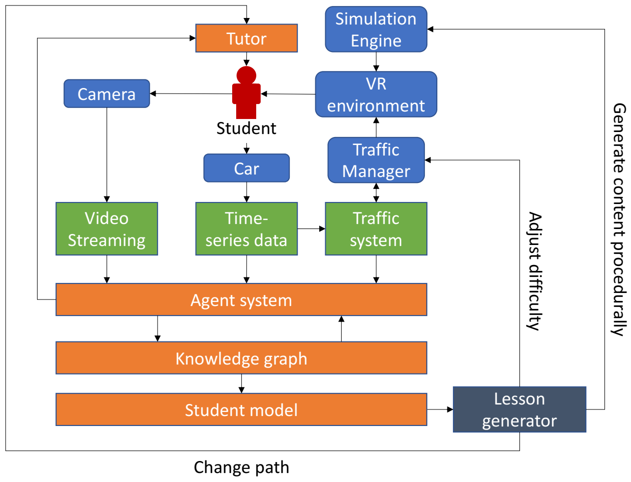
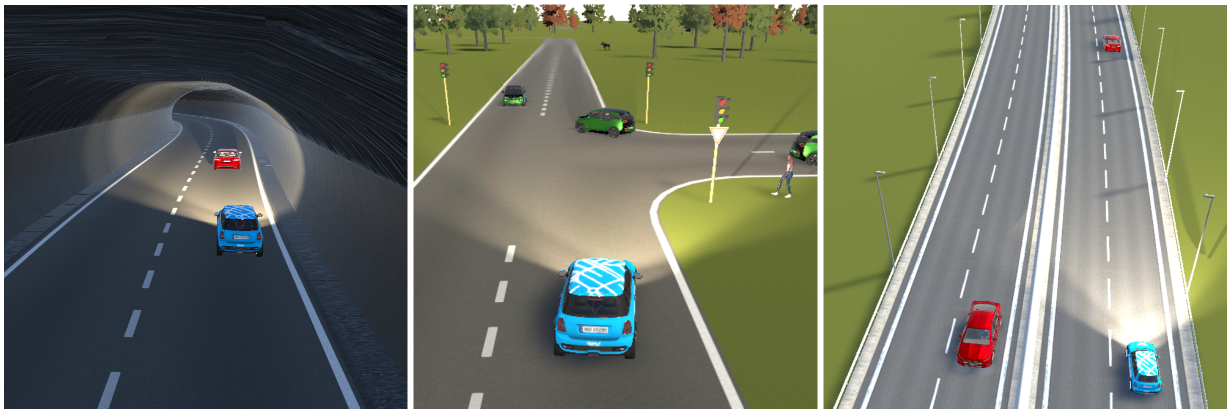


Figure 1: High-level architecture of the complete system.

For these exact reasons, intelligent tutoring systems supporting drivers in a simulator environment have been proposed to help students learning to drive. Examples include CarCoach (Arroyo, Sullivan, and Selker 2006), the virtual driving instructor (VDI) deployed in Green Dino simulators (Weevers et al. 2003), which can be interpreted as serious games for which Greitzer, Kuchar, and Huston (2007) propose five design guideline principles: 1) *Stimulate semantic knowledge* by relating material to the learner's experiences and existing semantic knowledge structures, 2) *manage the the learner's cognitive load* by organize the material into smaller chunks and make the material gradually more complex, 3) *Immerse the learner in problem-centered activities* by providing opportunities to work on meaningful and realistic tasks immediately, 4) *emphasize interactive experiences* by requiring the manipulation of objects to build lasting memories, and 5) *engage the learner* through keeping learners within a narrow range of difficulty where the material is challenging but not overwhelming.

Procedural content generation has been used to generated learning environments in several different domains, including level design for games (Calimeri et al. 2018) such as track generation for racing games (Togelius, De Nardi, and Lucas 2007) and level generation for Super Mario games

[*]These authors contributed equally.



(a) Tunnel. (b) T-Intersection with traffic lights. (c) Highway.

Figure 2: Examples of tiles generated for a student of high proficiency with some weaknesses.

(Shi and Chen 2016), but also personalized map generation (Raffe et al. 2014), road networks in games (Teng and Bidarra 2017), traffic scenario generation for testing of autonomous vehicles (Li et al. 2021), scenario generation for emergency rescue training games (Hullett and Mateas 2009), and serious game for conflict resolution (Grappiolo et al. 2011). There is also research focusing on personalized learning. Notable examples include (Chen, Lee, and Chen 2005) that utilizes item response theory for course material recommendation, (Huang, Huang, and Chen 2007) that utilizes case-based reasoning and genetic algorithms also for recommending course materials and (Hwang et al. 2010) that proposes a heuristic algorithm for context-aware ubiquitous learning applied to natural science butterfly-ecology. Most relevant for this work is (Ropelato et al. 2018) who introduce an adaptive tutoring system for a virtual reality driving simulator. The intelligent tutor monitors skills such as stable driving on both straight and curved roads, turning, complete stops, constant speed and reaction, and selects the next activity based on performance on the past activity in a predefined map, so that the student has to drive all the way to the part of the map that will train the desired skill.

We present a module, the *Lesson generator*, that extends the VDI developed by Way AS to generate personalized lessons for individual students with the goal to practice in a focused and deliberately fashion. The VDI provides immediate feedback, time for problem-solving and evaluation, and opportunities for repeated performance to refine behavior in line with deliberate practice (Anders Ericsson 2008). According to the classification of Togelius et al. (2011), the Lesson generator generates necessary content offline that the student has to interact with based on a parameter vector that describes the student skill level in a constructive way based on heuristic rules with some stochasticity.

System Architecture

An overview of the system architecture is given in Figure 1. For further details see (Sandberg et al. 2020; Rehm, Reshodko, and Gundersen 2023).

High-fidelity simulator: The simulator is an actual car placed in a room where the virtual world modeled in Unity (Unity Technologies 2022) is projected onto the walls so that looking in the mirrors will be realistic. This setup, including actuators that make the car move, increases the realism and the feeling of driving a real car.

Shared data: The *video streaming database* stores the video of the driver, so that the viewing direction of the driver can be analyzed. Data from the car are transferred using the CAN bus protocol and stored in the time-series database. The *traffic system* keeps track of all dynamic elements in the simulation including their location in the map and relational information such as the lanes cars are in.

Knowledge graph: The knowledge graph represents and organizes all relevant information of the continuously developing traffic situation. It is built on a property graph model (Hogan et al. 2021) that allows both nodes and relationships to be characterized by attributes.

Agent system: The agent system is a multi-agent system following the principles of subsumption architecture (Brooks 1991). Agents are responsible for building and maintaining the knowledge graph through fuzzy logic, rule-based reasoning and deep learning. Each agent has a concrete task. One agent could monitor whether mirrors or the blind zones are checked by analyzing the video stream while another would be checking whether this was done in the right sequence.

Student model: The student model is comprised of a static representation of skills in a Bayesian Network structure along with beliefs of a students skill level that is estimated from observing actions made in the virtual environment.

Tutor: The tutor is an agent responsible for giving feedback to the student on the driving. The feedback includes what is done well but also what can be improved and how. More feedback is given to less experienced students and less is given to more capable ones. Also, the tutor varies its feedback, and feedback modes include audio, visual in an in-car-dashboard, and visual cues in the virtual environment.

Lesson generator: The lesson generator is the main focus of this paper. It is a component of the virtual instructor that generates personalized lessons for individual students.

The VDI is designed to follow the design guideline principles proposed by Greitzer, Kuchar, and Huston (2007). It stimulates semantic knowledge by relating material to the experiences and observed mastery of the students, seeks to manage the the learner's cognitive load by making the material gradually more complex, immerses the learner in problem-centered activities and emphasizes interactive by letting the student drive in a personalized traffic environment, and engages by keeping learning within a narrow range of difficulty by adjusting the lesson to the student mastery of the driving skills and current performance.

Problem Description

The problem we seek to solve in this paper is to automatically tailor lessons for students driving in a high-fidelity simulator to increase the students' learning rate and reduce the time required to learn skills well. The end goal is to deliberately generate lessons that practice the skills that individual students need to improve on the most to achieve the skill level required to pass the driving test. Lessons should be generated based on the observed skill level of the students and adjusted according to observed performance. In this work, we do not intend to adjust the teaching style of the virtual driving instructor nor adapt the lesson or the feedback according to the student's learning style, although this could be the focus of future work on how to improve the tutor.

Hence, a *lesson* $l \in \mathcal{L}$ is comprised of a *map* $m \in \mathcal{M}$ of a virtual environment that contains static traffic elements, such as roads, signs, and traffic lights, *dynamic elements* $d \in \mathcal{D}$ such as cars and pedestrians, *events* $e \in \mathcal{E}$ that capture predefined dynamic traffic scenarios, such as yielding and overtake scenarios, and a *path* $p \in \mathcal{P}$ through the map that ensures that the student get to practice situations that the lesson intended the student to practice.

A *lesson generator* produces a personalized lesson l_s for a student in accordance to the skill level of a student $s \in \mathcal{S}$. Some skills are trained by only driving through a static map, such as lane positioning, while others require dynamic elements, such as an overtake scenario where a vehicle that moves slowly must be passed. Some static elements are more complex than others, as for example an intersection compared to a straight road segment. The complexity of a situation does not only depend on the complexity of the static elements, it also depends on how many dynamic elements the situation contains. Hence, the complexity $c \in \mathcal{R}$ of a lesson $l_{s,c}$ can be adjusted by increasing or decreasing the number of dynamic elements in the lesson. It can also be adjusted online during the lesson according to the observed performance of the student.

Representing Personalized Driving Lessons

A personalized lesson contains a map with both static and dynamic content in addition to a path through the map that ensures that the student experiences the content:

Map: A map is a set of tiles and chunks and the location of these that constitutes the virtual traffic environment that the student is to drive through. It lays out all the static objects that represent the virtual environment. Unique maps are generated for each student to train the skills that the student should practice on to become a better driver.

Tiles: Tiles are smaller pieces of the map that contain individual road segments populated with other static objects, such as crosswalks, tunnels, highways and traffic lights. Road segments are static objects, and they can be curved or straight segments, but more often they contain intersections of various types, such as T-intersections and roundabouts.

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, such as training of an overtaking maneuver which necessitates a slow-moving vehicle in front, complemented by an appropriately structured road conducive to overtaking.

Chunks: 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. Chunks are either collections of tiles of the same type, such as X-intersections or a collection of predefined tiles comprising a long tunnel.

Dynamic elements: Dynamic elements are elements that move in the simulation environment and add to the realism of the driving experiences by introducing unpredictability. The number of dynamic elements can be adjusted to change the complexity of the learning environment, and thereby adjust the level of difficulty. Dynamic elements include but are not restricted to vehicles, pedestrians, animals and bikes.

Path: Paths are routes through the map that ensure that all skills that the map is generated to train actually is trained. They are generated so that all chunks are visited without necessarily visiting all tiles. The shortest route between two chunks are identified by the path planning algorithm.

The lesson generator supports deliberate practice through generating lessons based on the skill level and by changing the path through the map dynamically based on the observed performance on particular skills.

Student Skill Level

The student skill level is modeled as a Bayesian network, consisting of two types of variables: directly measurable evidence, called performance variables, and latent skill mastery variables. The skill mastery variables capture causal relationships between distinct driving skills, such as overtake, mirror checking, and blinking comprising a *skill network*, which is modeled in collaboration with professional driving instructors. Complex skills are composed of more basic skills. For example, the overtake skill is composed of several skills including mirror checking and blinking. While blinking is an atomic skill, mirror checking is a composite skill as well and includes checking the rear-view mirror, sideways mirrors and the blind zones in a pre-defined sequence.

Skill mastery nodes are characterized by three potential states: mastered, learning, and struggling. These mastery nodes have the flexibility to have an arbitrary number of parents and child skill mastery nodes, as well as associated evi-

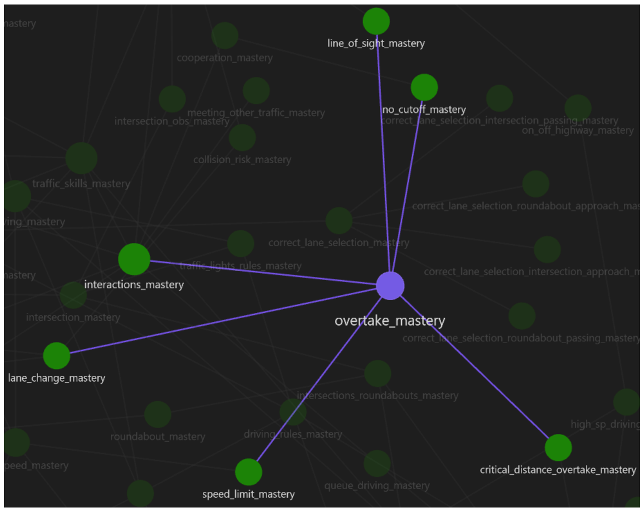

line_of_sight_mastery
cooperation_mastery
no_cutoff_mastery
meeting_other_traffic_mastery
intersection_obs_mastery
correct_lane_selection_intersection_passing_mastery
on_off_highway_mastery
collision_risk_mastery
traffic_skills_mastery
correct_lane_selection_roundabout_approach_mas
correct_lane_selection_mastery
traffic_lights_rules_mastery
correct_lane_selection_intersection_approach_m
interactions_mastery
intersection_mastery
overtake_mastery
correct_lane_selection_roundabout_passing_mastery
lane_change_mastery
intersections_roundabouts_mastery
high_sp_driving
roundabout_mastery
driving_rules_mastery
critical_distance_overtake_mastery
speed_limit_mastery
queue_driving_mastery

Figure 3: Skill network emphasizing overtake mastery and its parent nodes.

dence. Typically, skill mastery nodes remain unobserved, allowing probabilistic inference. Agents generates the directly measurable evidence for the most basic skills, which is kept up-to-date in real-time in the knowledge graph.

The conditional probability distribution for parental-child skill mastery defines the inter-dependencies among the mastery nodes within the network described in equation 1:

$$P(Mastery_s|Parent_1, Parent_2, ..., Parent_m), \quad (1)$$

where $Parent_i \in \{Mastered, Learning, Struggling\}$. Figure 3 illustrates a small portion of the skill network representing *overtake mastery* and its parent mastery nodes *lines of sight*, *no cutoff*, *critical distance overtake*, *speed limit*, *lane change* and *interactions*. The conditional probability distribution can be established through heuristic approaches or be acquired through the analysis of a dataset generated by knowledgeable driving instructors. The latter is the case for the network used in this research.

The performance variables are measured through direct observations of how a student performs in particular traffic situations during driving sessions in the simulator. The direct observations are done by agents that have tasks that to a large degree mirror the skills in the skill network. The agents register their findings in the knowledge graph so that other agents can find the information there and reason with this information for their own tasks. The goal is to predict the probability distribution of the latent variables based on the observed evidence through the model's inter-dependencies. In this way, the students' performance on particular skills can be assessed and their progress monitored over time.

Generating Personalized Driving Lessons

The map generated for a lesson is designed to provide variety to keep the learner engaged. The lesson automatic design considers the interleaving of different skills, in line with research showing that interleaved practice can lead to better learning outcomes than blocked practice (Dunlosky et al. 2013). Combining these principles enables the generation of lessons that are engaging, effective, and personalized for each learner's skill level. See Figure 4 for an example.

Procedural Generation of the Map

The *Lesson generator*'s map design algorithm revolves around two different types of preferences: 1) the preference of training the skills for which a student has the lowest proficiency and 2) the preference of the tiles that trains this exact skill. To engage the student, lessons are designed to contain varied maps training a broader set of skills than only those for which the student have the lowest proficiency. Therefore, the process of generating maps for particular students with particular skill levels is not an optimization process. Instead the maps are generated stochastically in a process where map tiles that train the low proficiency skills have a higher probability of being drawn than those the student are more proficient in. The probability of drawing a tile is calculated as follows:

$$P_t = \frac{U_t}{\sum_{t \in T} U_t}, \quad (2)$$

where U_t is the utility of a tile. U_t defines how well the skill trained by the tile t match with the skills that a student needs to train the most:

$$U_t = \beta_t + \frac{\sum_{s \in S_t} L_s \times U_s}{n_t}, \quad (3)$$

where $s \in S_t$ represents all skills that the tile trains, L_s is the learning efficiency of a tile, meaning to which degree a tile trains a skill, n_t represents the number of skills a tile trains, β_t represents how well the difficulty of a tile aligns with the skill that the student's mastery of the skill mostly trained by a tile, and U_s is the utility of a skill. U_s captures the degree to which a skill s should be selected for training given how well a student master a skill and which skills that already have been selected for training:

$$U_s = (1 - M_s)^u - F_s v, \quad (4)$$

where M_s represent how well a student master skill s (0 means that the student does not master the skill while 1 means complete mastery, and the actual value comes from the student model), u is a parameter controlling the strength for selecting a skill, F_s represents the frequency of a skill being trained in the map generated so far, and v represents the weight with which a skill frequently chosen should be punished (increasing v promotes more diversity in skill representation). The probability of drawing a skill, P_s, is computed similarly as for drawing tiles in equation 2. Tiles are added to the map iteratively until the duration of driving through the map d_m is estimated to be longer than the duration of a driving lesson d_l.The process is sketched out in Algorithm 1. All parameters are determined experimentally.

Adding Dynamics to the Map

While tile-based events that target training of specific skills, such as overtake, are added as part of the procedural generation of the map, dynamic elements, such as vehicles, are added after the map is generated. The tile-based events are

Algorithm 1: Generating maps

Input: Skill level S_i of student i
Input: Minimum duration of a lesson d_l
Output: A map m_i that is designed for student i at the student's current skill level S_i.

```
1: Let $d_m = 0$.
2: Let $U_s = \{\}$.
3: Let $U_t = \{\}$.
4: while $d_m < d_l$ do
5:     % Calculate utility for all skills
6:     for $s \in S_i$ do
7:         $u_s \leftarrow$ calculate skill utility using equation 4
8:         $U_s \leftarrow u_s$
9:     end for
10:    % Draw skill $s$ to train from $U_s$
11:    $s \leftarrow$ draw from $U_s$ using equation 2
12:    % Calculate utility for all tiles $T_s$ training skill $s$
13:    for $t \in T_s$ do
14:        $u_t \leftarrow$ calculate tile utility using equation 3
15:        $U_t \leftarrow u_t$
16:    end for
17:    % Draw tile $t$ to generate chunk from
18:    $t \leftarrow$ draw from $U_t$ using equation 2
19:    $C_t \leftarrow$ generate chunk from $t$
20:    % Add chunk to map $m_i$
21:    $m_i \leftarrow C_t$
22:    $d_m \leftarrow$ estimate duration of $m_i$
23: end while
24: return $m_i$
```

not only located on specific tiles, elements part of the event have predefined actions and start to interact with the world when the student is approaching the specific event. Several skills require particular events to happen at the correct time to provide students with the necessary situation for skill development. For example, training of overtaking maneuvers necessitates a slow-moving vehicle in front of the student so that the student has to pass it. The predefined setup enforced by an event differs from how dynamic elements are added to the map. Dynamic elements are added to introduce complexity into the learning environment, and thereby increasing the level of difficulty. They aim is to emulate the unpredictability of real-world driving and can be added into any tile. The skill level is used to decide the appropriate difficulty for that student and subsequently set the amount of traffic and pedestrians in the virtual environment.

Static and dynamic content and the path are currently added in the offline generation of a lesson. However, both dynamic elements and the path can be adjusted online during the lesson in response to the student performance. For example, the number of dynamic elements in the virtual world can be reduced if the students fail at tasks where the dynamic elements are extraneous stressors to the skill being learned (Rudland, Golding, and Wilkinson 2020). Also, the path can be changed to increase training of a skill that the student need to practice more or to reduce the practice of a skill the student excels at. The path can also change if the student is driving through the lesson faster than estimated so that the student fills the time allotted to her.

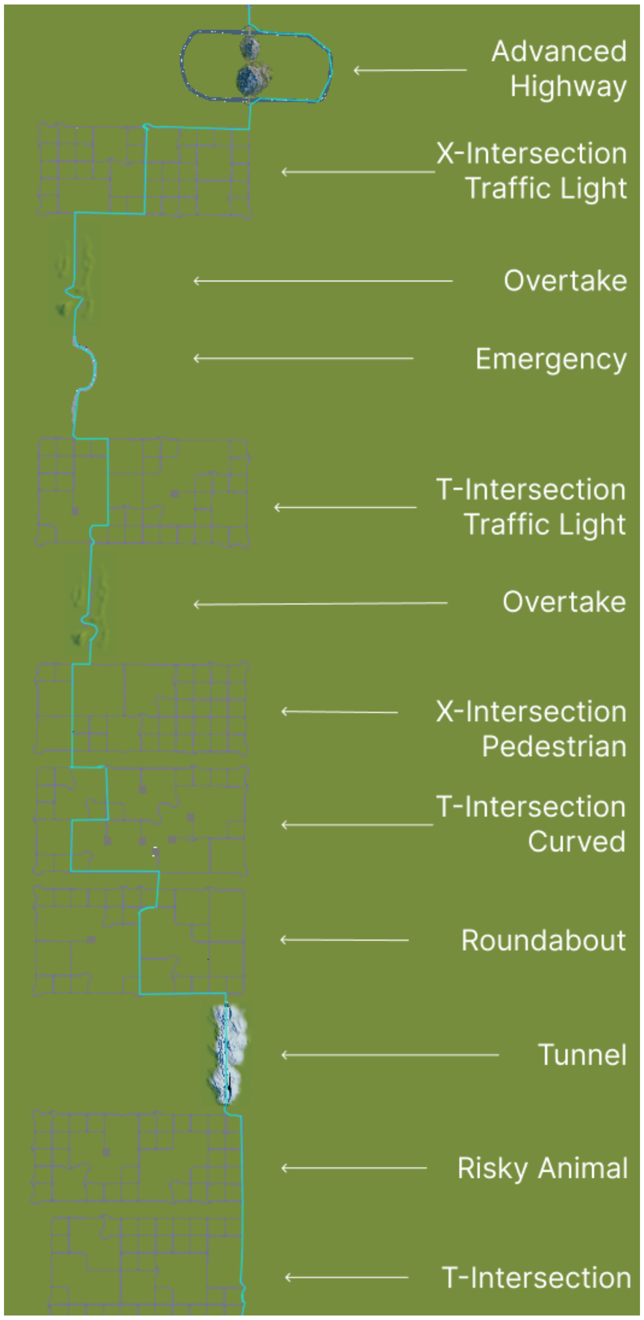
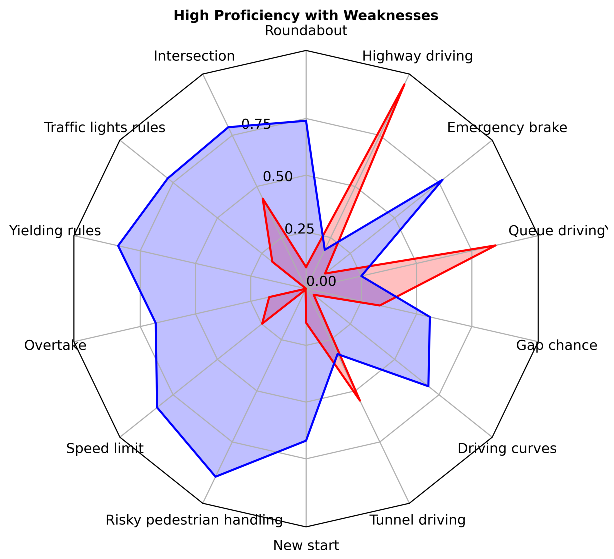


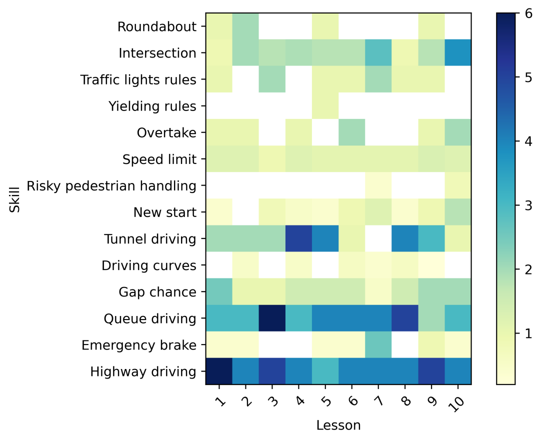
Figure 4: Example of automatically generated lesson.

Case Study

In order to evaluate the viability of the lesson generator systematically, we have done a case study of four different students with different characteristics. The case studies are done through simulation for two reasons: 1) to control the characteristics of the students and 2) to get early feedback before starting to test on actual students. A proper evalua-



(a) Radar diagram showing skills trained in lessons (red) and skill levels of a student (blue).



(b) Distribution of skills trained for ten consecutive lessons generated for the exact same student.

Figure 5: Case study of high proficiency student with the weaknesses tunnel, queue and highway driving.

tion requires an improved and diverse set of tiles, which is both costly and requires effort of sparse resources to design.

To control the study variables, the case study consists of four students with differing skill levels characterized as *low proficiency*, *low proficiency with strengths*, *high proficiency*, and *high proficiency with weaknesses*. Low proficiency skills have low values, while high proficiency skills have high values, so a low skill proficiency student with strengths generally has low skill levels but some that score higher. Furthermore, for each of the 22 predefined tile types, one example tile has been created for this test, including *straight road with crosswalk*, *curved road*, *T-intersection*, *tunnel*, *T-intersection with traffic lights* and *high-way*. Our experiment resulted in the expected differences in generated routes for the four profiles underpinning the viability of the method.

Figure 2 shows three tiles that were generated for a high proficiency student with some weaknesses. The tiles train complex skills such as T-intersections with traffic lights and highways. They are also populated with quite some dynamic elements, inclucing cars, a pedestrian, and an animal that can be glimpsed in the back of Figure 2b. Figure 5a shows a radar diagram of the skill level (blue) of the high proficiency student with some weaknesses. From the figure it is clear that the weaknesses are highway driving, queue driving and tunnel driving. The figure also shows that the driving lesson that has been generated trains these skills specifically (red). Figure 5b illustrates the distribution of skills trained per lesson when ten consecutive lessons are generated for the same student represented by the exact same skill network. It shows that the skills representing the weaknesses of the student are consistently trained the most. Figure 4 illustrates a complete lesson for the the high proficiency student. The twelve different chunks are described and the path going through the map is highlighted. Different skills, none with low complexity, are trained in this map, which is done to keep the student challenged and engaged.

Limitations and Path to Deployment

The research presented here is a pilot that has been conducted on synthetic students represented by carefully designed skill models, so the system has not been tested on human students yet. Also, the test only involved offline generation of content, which means that online dynamics have not yet been tested, although the current design fully supports it. Furthermore, the tiles are not of production quality and only one tile per tile type has been designed. Developing 3D content of high quality is costly and requires skilled personnel, and hence this has been delayed until the system was tested in an offline fashion. In the future, tiles could be imagined to be generated fully on the fly instead of generating the map offline.Also, we would like to personalize the teaching style and test and evaluate the feedback to given students to further improve the learning rate.

Procedurally generating personalized lessons based on the skill level of students is key to fully automate driver education training and removing the costly involvement of human driving instructors in the early stage of driver education (Rehm et al. 2024). The lesson generator will be deployed at the driving school Way AS in stages and not fully replace the current regime where students train particular skills per lesson. Both students and traffic educators will be involved to provide qualitative feedback on how lessons best can be generated to help students learn faster. In addition, controlled experiments will be conducted to monitoring progress quantitatively. Our experience is that a feedback loop involving students, teachers and developers should be established to ensure a high-quality solution.

Acknowledgments

This research was funded by Way AS and the Norwegian Research Council through the TRANSPORT program under the grant agreement 296640.

References

Anders Ericsson, K. 2008. Deliberate practice and acquisition of expert performance: a general overview. *Academic emergency medicine*, 15(11): 988–994.

Arroyo, E.; Sullivan, S.; and Selker, T. 2006. CarCoach: a polite and effective driving coach. In *CHI'06 Extended Abstracts on Human Factors in Computing Systems*, 357–362.

Brooks, R. A. 1991. Intelligence without representation. *Artificial intelligence*, 47(1-3): 139–159.

Calimeri, F.; Germano, S.; Ianni, G.; Pacenza, F.; Pezzimenti, A.; and Tucci, A. 2018. Answer set programming for declarative content specification: A scalable partitioning-based approach. In *International Conference of the Italian Association for Artificial Intelligence*, 225–237. Springer.

Chen, C.-M.; Lee, H.-M.; and Chen, Y.-H. 2005. Personalized e-learning system using item response theory. *Computers & Education*, 44(3): 237–255.

Dunlosky, J.; Rawson, K. A.; Marsh, E. J.; Nathan, M. J.; and Willingham, D. T. 2013. Improving students' learning with effective learning techniques: Promising directions from cognitive and educational psychology. *Psychological Science in the Public Interest, Supplement*, 14(1): 4 – 58. Cited by: 1539.

Grappiolo, C.; Cheong, Y.-G.; Togelius, J.; Khaled, R.; and Yannakakis, G. N. 2011. Towards player adaptivity in a serious game for conflict resolution. In *2011 Third International Conference on Games and Virtual Worlds for Serious Applications*, 192–198. IEEE.

Greitzer, F.; Kuchar, O.; and Huston, K. 2007. Cognitive science implications for enhancing training effectiveness in a serious gaming context. *ACM Journal of Educational Resources in Computing*, 7.

Hogan, A.; Blomqvist, E.; Cochez, M.; D'amato, C.; Melo, G. D.; Gutierrez, C.; Kirrane, S.; Gayo, J. E. L.; Navigli, R.; Neumaier, S.; Ngomo, A.-C. N.; Polleres, A.; Rashid, S. M.; Rula, A.; Schmelzeisen, L.; Sequeda, J.; Staab, S.; and Zimmermann, A. 2021. Knowledge Graphs. *ACM Comput. Surv.*, 54(4).

Huang, M.-J.; Huang, H.-S.; and Chen, M.-Y. 2007. Constructing a personalized e-learning system based on genetic algorithm and case-based reasoning approach. *Expert Systems with applications*, 33(3): 551–564.

Hullett, K.; and Mateas, M. 2009. Scenario generation for emergency rescue training games. In *Proceedings of the 4th International Conference on Foundations of Digital Games*, 99–106.

Hwang, G.-J.; Kuo, F.-R.; Yin, P.-Y.; and Chuang, K.-H. 2010. A heuristic algorithm for planning personalized learning paths for context-aware ubiquitous learning. *Computers & Education*, 54(2): 404–415.

Li, A.; Chen, S.; Sun, L.; Zheng, N.; Tomizuka, M.; and Zhan, W. 2021. Scegene: Bio-inspired traffic scenario generation for autonomous driving testing. *IEEE Transactions on Intelligent Transportation Systems*, 23(9): 14859–14874.

Raffe, W. L.; Zambetta, F.; Li, X.; and Stanley, K. O. 2014. Integrated approach to personalized procedural map generation using evolutionary algorithms. *IEEE Transactions on Computational Intelligence and AI in Games*, 7(2): 139–155.

Rehm, J.; Reshodko, I.; Børresen, S. Z.; and Gundersen, O. E. 2024. The Virtual Driving Instructor: Multi-Agent System Collaborating via Knowledge Graph for Scalable Driver Education. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38.

Rehm, J.; Reshodko, I.; and Gundersen, O. E. 2023. A virtual driving instructor that assesses driving performance on par with human experts. Forthcoming.

Ropelato, S.; Zünd, F.; Magnenat, S.; Menozzi, M.; and Sumner, R. 2018. Adaptive tutoring on a virtual reality driving simulator. *International SERIES on information systems and management in creative emedia (CreMedia)*, 2017(2): 12–17.

Rudland, J. R.; Golding, C.; and Wilkinson, T. J. 2020. The stress paradox: how stress can be good for learning. *Medical education*, 54(1): 40–45.

Sandberg, M. K.; Rehm, J.; Mnoucek, M.; Reshodko, I.; and Gundersen, O. E. 2020. Explaining traffic situations–architecture of a virtual driving instructor. In *Intelligent Tutoring Systems: 16th International Conference, ITS 2020, Athens, Greece, June 8–12, 2020, Proceedings 16*, 115–124. Springer.

Shi, P.; and Chen, K. 2016. Online level generation in Super Mario Bros via learning constructive primitives. In *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, 1–8. IEEE.

Teng, E.; and Bidarra, R. 2017. 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*, 1–10.

Togelius, J.; De Nardi, R.; and Lucas, S. M. 2007. Towards automatic personalised content creation for racing games. In *2007 IEEE Symposium on Computational Intelligence and Games*, 252–259. IEEE.

Togelius, J.; Yannakakis, G. N.; Stanley, K. O.; and Browne, C. 2011. Search-based procedural content generation: A taxonomy and survey. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3): 172–186.

Unity Technologies. 2022. Unity. https://unity.com/. Version 2021.3.4.

Weevers, I.; Kuipers, J.; Brugman, A. O.; Zwiers, J.; Van Dijk, E. M.; and Nijholt, A. 2003. The virtual driving instructor creating awareness in a multiagent system. In *Advances in Artificial Intelligence: 16th Conference of the Canadian Society for Computational Studies of Intelligence, AI 2003, Halifax, Canada, June 11–13, 2003, Proceedings 16*, 596–602. Springer.