Adaptive Tutoring on a Virtual Reality Driving Simulator

Sandro Ropelato
ETH Zurich
sandro.ropelato@gmail.com

Fabio Zünd
ETH Zurich
fabio.zund@inf.ethz.ch

Stéphane Magnenat
ETH Zurich
stephane@magnenat.net

Marino Menozzi
ETH Zurich
mmenozzi@ethz.ch

Robert W. Sumner
ETH Zurich / Disney Research Zurich
robert.sumner@inf.ethz.ch

ABSTRACT

We propose a system for a Virtual Reality (VR) driving simulator including an Intelligent Tutoring System (ITS) to train the user's driving skills. The VR driving simulator comprises a detailed model of a city, Artificial Intelligence (AI) traffic, and a physical driving engine, interacting with the driver. In a physical mockup of a car cockpit, the driver operates the vehicle through the virtual environment by controlling a steering wheel, pedals, and a gear lever. Using a Head-Mounted Display (HMD), the driver observes the scene from within the car. The realism of the simulation is enhanced by a 6 Degrees of Freedom (DOF) motion platform, capable of simulating forces experienced when accelerating, braking, or turning the car. Based on a pre-defined list of driving-related activities, the ITS permanently assesses the quality of driving during the simulation and suggests an optimal path through the city to the driver in order to improve the driving skills. A user study revealed that most drivers experience presence in the virtual world and are proficient in operating the car.

KEYWORDS

virtual reality, driving simulation, intelligent tutoring system

ACM Reference Format:
Sandro Ropelato, Fabio Zünd, Stéphane Magnenat, Marino Menozzi, and Robert W. Sumner. 2017. Adaptive Tutoring on a Virtual Reality Driving Simulator. In *Proceedings of 1st Workshop on Artificial Intelligence Meets Virtual and Augmented Worlds (AIVRAR), BITEC, Bangkok, Thailand, November 2017 (AIVRAR'17),* 6 pages.

1 INTRODUCTION

Learning how to drive a car involves many hours of training. Combinations of complex activities require the driver's full attention, and even experienced motorists make mistakes or show wrong reactions when faced with unexpected events. Having a simulated setup to improve car driving skills can be useful to both novice and experienced drivers, as a variety of scenarios that might occur on real roads can be exercised in a safe environment. In order to imitate real-life driving scenarios, an immersive Virtual Reality (VR) environment is required.

Training softwares running in a workplace-like environment, with a single screen and a keyboard, can be used to simulate an interactive car ride. However, presence and immersion in such setups are strongly limited. A keyboard does not resemble the instruments used to control a car, and a regular monitor fails to offer a sufficiently large field of view to experience movement. In addition, there is no physical feedback of acceleration and no intuitive way of looking around in the virtual environment.

Computer programs have been used to assist in training skills and have shown improvement in the learning progress when adapting to the individual learner. We combine a VR headset with a 6 Degrees of Freedom (DOF) motion system to improve the presence and immersion of the driving simulation, and include an Intelligent Tutoring System (ITS) to adapt the training to the individual user. In our proposed system, the ITS suggests an optimal sequence of exercises, such as stable driving, turning, and reaction, to the user. These exercises are spatially distributed in the virtual city. As such, the ITS is well-suited to be integrated into the driving environment as a Satellite Navigation System (satnav), which leads the user from one exercise to the next. Hence, the ITS does not interfere with the driving immersion and the user can follow the instructions provided by the satnav without being distracted from driving.

2 RELATED WORK

Virtual Reality Simulations. The Railway Technical Research Institute (RTRI) has developed a VR safety simulation system where users can train how to respond to critical situations [9]. In a virtual environment, various types of problems can be simulated. Users of the system are required to cooperate with each other and resolve problems in order to prevent further complications and restore services as soon as possible. The goal of this system is to offer a safe environment where unpredictable or dangerous incidents can be handled by public transport staff. The knowledge and experience gained in training situations in the virtual environment can be projected onto real-life scenarios and improve people's performance in problem solving.

In a different simulation, Augusto et al. [1] show how VR can be used to train security staff in securing and protecting nuclear facilities. Based on construction plans of a nuclear power plant, a virtual environment is created. In a game-like training mode, security staff watch the facilities while the system simulates an infiltration attempt where intruders try to access restricted areas. The authors propose a way to improve physical security of nuclear facilities in two ways, by offering a method to analyze the facility's infrastructure and by enabling security personnel to actively practice operations in VR.

Both of the aforementioned projects aim at improving real-life performance in difficult scenarios by providing training in virtual environments, from which users learn how to handle real-life situations. In both examples, however, the exercises in the simulated scenarios are manually created and not adapted to the specific skills of the user. Carefully matching the tasks and their order to the needs of the user requires manual input.

Learning Progress. Previous work [3] has shown that the order in which exercises are solved can have a major influence on the learning progress. The optimal sequence depends on the subject solving the exercise and varies between individuals. The authors proposed a method to estimate the learning progress in each exercise and generate a sequence tailored to each user. When testing their algorithm, Zone of Proximal Development and Empirical Success (ZPDES), on primary school pupils solving basic math exercises, they showed that the system-generated order of activities yields a better overall learning progress than one defined by experts.

In our work, we combine the use of a VR training environment with the approach of dynamically adapting the sequence of trained activities using the ZPDES algorithm.

3 HARDWARE AND SOFTWARE ENVIRONMENT

Applying a tutoring system to car driving requires an environment where different skills can be trained. For this purpose, we created a VR driving platform to simulate an interactive car ride through a city. We used Unity, a 3D game engine, to combine visuals, a physics simulation, interaction with input devices, and a motion system. The following section presents an overview of all components used in the simulation, and describes how they interact with each other.

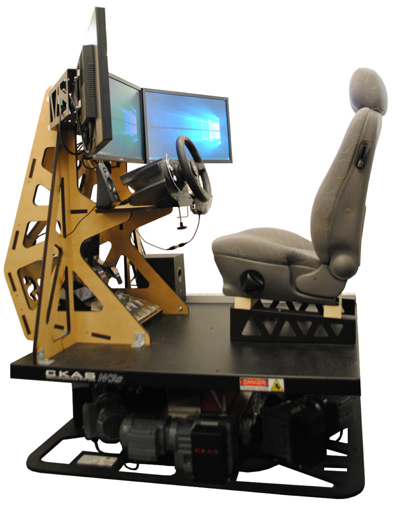


Figure 1: Cockpit mockup on the motion platform.

3.1 Hardware

Motion System. To exert physical forces on the driver, we employ a Thruxim Pro 6 DOF motion simulator by CKAS Mechatronics Pty Ltd. It supports linear displacement of the driving platform along the X, Y and Z axes, as well as rotation in all three directions. This allows simulating linear acceleration by moving and tilting the platform in the corresponding direction. For example, when accelerating in a real car, the driver is being pushed back into the seat. On the simulator, this can be imitated by rotating the platform around the horizontal axis. Lateral forces that occur while turning a car can be simulated by tilting the platform around the longitudinal axis. When the simulated acceleration is constant or changes very slowly, this creates an illusion of linear acceleration without an actual linear movement. In car driving, however, there are strong changes in acceleration. For instance, when driving at a constant speed and then immediately braking, the acceleration along the forward axis changes in almost no time from 0 G to as much as 1 G [8]. When the rotation of the platform changes too quickly, the motion is perceived as a rotation around the center of the platform rather than a change in velocity, thus destroying the illusion of linear acceleration. To avoid this, the tilting is supported by an actual linear movement along the corresponding axis. Especially at higher acceleration change rates, this can reduce the perception of a rotational movement [2].

Cockpit Mockup. The platform is equipped with a driving seat taken from an old Ford Ka and a wooden frame for mounting the steering wheel, the pedals, and three screens. Three 27-inch monitors have been arranged to offer a feld of view of up to 120 degees, depending on the driver's position. A Thrustmaster T500RS steering wheel and pedal set along with an 8-gear shifter imitate input devices present in real cars. Force feedback can be applied to the steering wheel. Fig. 1 shows the cockpit mockup mounted on the motion platform.

Head-Mounted Display. The virtual environment is either presented on the three screens or through an HTC Vive Head-Mounted Display (HMD). The HMD is tracked using two base stations installed on the ceiling above the simulation platform. When moving or turning the head, the camera is moved accordingly in the virtual scene so that the driver is able to examine objects in the cockpit from different angles. Movement of the motion platform is subtracted from the HMD's position and rotation so that the driver's perspective remains relative to the cockpit when linear acceleration is simulated. The Vive's display has a resolution of 2160×1200 pixels (1080×1200 per eye) and its optics offer a field of view of up to 110 degrees. The display refresh rate is 90 Hz which allows for low-latency updates when moving and rotating the head [10]. The tracking system records the HMD's position with a maximum tolerance of 2 mm [6]. For our application, this is precise enough so that the user does not detect any jitter. We considered the total weight of 550 g [7] to be acceptable in that, even after test runs on the simulator above 30 minutes, no driver reported discomfort from the headset's weight.

Computer. A gaming computer with a 4 GHz Intel Core i7-6700K processor, 32 GB memory, and two Nvidia GeForce 1080 graphics cards, with 8 GB graphics memory each, drive the simulation.

3.2 Physics Simulation

The way a car behaves is influenced by various parameters such as mass, engine power, tire friction, and suspension. Some of these can be easily modeled with Unity's built-in physics engine. For others, we had to build new models based on specific characteristics of the car.

Suspension and Tires. On an abstract level, a vehicle such as a car consists of a rigid body with a number of wheels attached. In this case, there are four wheel, two of which are driven by an engine. In Unity, each wheel is configured to push the body of the car upwards. As on a real car, the wheels are not directly connected to the body but use a simplified suspension, simulating a spring and a damper. The amount of force applied by the wheel colliders depends on the configuration of the suspension model.

Along with the suspension properties, the tire friction is defined for each wheel. Unity uses a two-spline curve to specify the force exerted on the contact point between the wheel and the road as a function of tire slip, the magnitude of the motion vector between a tire's contact point and the road. The tire slip is zero when the wheel has full traction and increases when the tire slides on the road, e.g. in an emergency brake. The wheel friction spline is defined by two points, (expremum slip, extremum force) and (asymptote slip, asymptote force). Given a value for tire slip, the force on the wheel is taken from evaluating the spline. Since this is a vague approximation of a tire's behavior, the values do not correspond to any specification but have been evaluated by testing the slipping behavior when accelerating, braking, or turning the car at high speeds.

Engine and Transmission. The simulated car has a combustion engine, which means that the torque produced is a non-linear function of the engine speed. It is usually specified in revolutions per minute (RPM). In other words, when pushing down the accelerator pedal, the force that is being output by the engine depends on how fast the engine is already going. For this simulation, we used the specification of a Fiat 500's engine [12]. The engine keeps its speed at a predefined RPM value when the car is stationary or driving very slowly. This is done by gradually increasing the throttle until the idling speed is reached. The engine stops when its speed drops below a minimum RPM value.

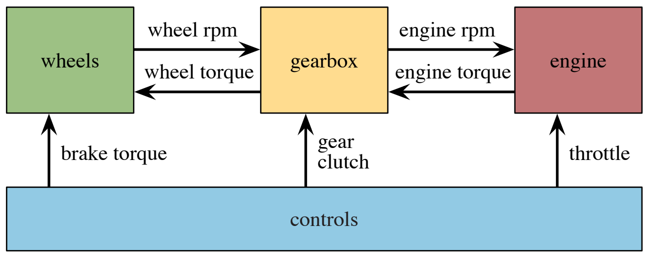


Figure 2: Components of the transmission model. The driver's input controls how fast the engine should accelerate, how much brake force is applied, how far the clutch is engaged, and which gear is selected.

The wheels are not directly driven by the engine. They are connected to the gearbox, translating the engine's speed to a different output speed defined by the gear's transmission ratio. As shown in Fig. 2, engine, gearbox, and wheels are connected to each other and are controlled by the driver's input.

3.3 3D Content Generation

Creating an appealing visual design substantially contributes to the realism of a virtual reality application. 3D models of cars, a detailed model of the car cockpit, and a set of traffic signals have been manually created to add to a life-like car driving experience. A city generator has been used to generate 3D models of buildings and a street layout upon which roads are dynamically constructed in Unity.

Car and Cockpit Model. We created a 3D model of a Fiat 500 and integrated it with Unity's built-in shaders and support for performance-saving Level of Detail (LOD) rendering. A detailed model of the car's interior has been designed to resemble the real cockpit. It contains a speedometer, a tachometer, mirrors, control LEDs for the indicators, and a satnav, which displays directions provided by the ITS. The mirrors correctly display the scene behind the car. This has been realized by placing three cameras in front of the mirrors, each rendering the virtual environment as seen through the respective mirror. The camera's rendered output is stored in a render texture and displayed on the mirror. Fig. 3 shows the view presented to the driver when sitting inside the car's cockpit.

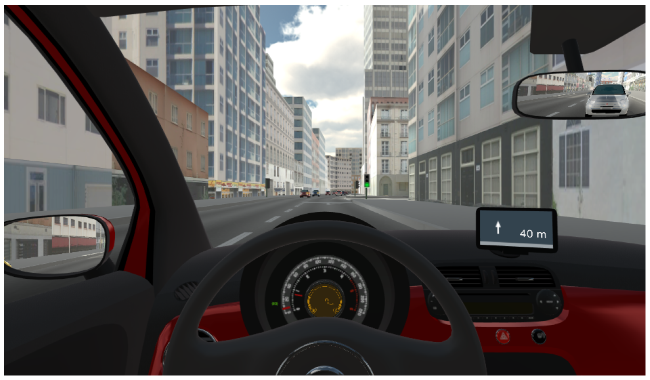


Figure 3: Interior view of the car. The mirrors reflect the environment behind the car. The satnav displays directions and distance to the next junction.

City Generation. Manually creating a large-scale virtual environment, such as a city with road junctions, buildings, and traffic signals, is a time-consuming task. For the generation of the simulated city we used CityEngine, a city generation software developed by Esri. It features rule-based geometry generation and offers highly-customizable models of buildings and roads. In a first step, we defined a road network graph containing nodes that are connected by road segments. Each road segment holds information about the number of lanes, the lane width. as well as the sidewalk width. Additional user-defined properties are added to define the maximum allowed speed on each lane and a flag indicating which lane goes in which direction. Once the road network has been defined, CityEngine subdivides areas enclosed by streets into footprints of buildings. It generates the building models and road geometry, including sidewalks. All generated models can be exported into a variety of 3D formats, including FBX, which are then imported into Unity.

For the driving simulator, we required more control over the exact shape and textures of the roads, especially at junctions. We therefore decided to not export the street geometry but to dynamically generate roads in Unity. A Python interface provides access to the coordinates and other attributes of all nodes and segments defined in the road network. With an export script, we write all information that is required to construct the roads into an XML file. An import script in Unity reads the exported file. In a first step, the road graph is created from the information contained in the node and segment tags. Then, the road geometry is created along the shape of the road segment and overlaid with asphalt textures containing road markings. Where two road segments join, the geometry is aligned so that there is no gap. In addition to the road geometry, sidewalks are generated along both borders of the road. As on real streets, the sidewalks are rounded on corners to enable proper turning at junctions. Fig. 4 shows a junction generated based on a CityEngine export. The complete city is shown in Fig. 5.

Performance. The city we created contains nearly 9 kilometers of roads, 519 buildings, and 40 cars that drive around the streets. Using the following optimization techniques, we achieved a framrate constantly above 60 Frames per Second (FPS). The number of draw

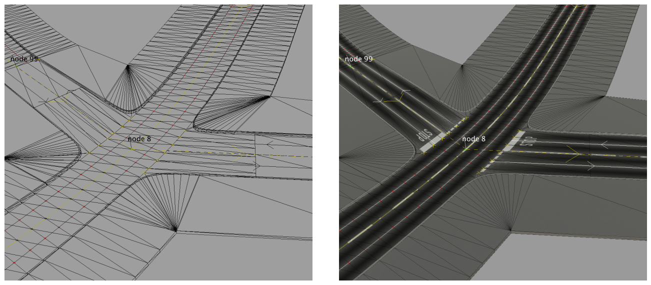


Figure 4: Road geometry with and without textures applied. The yellow lines mark the center of the road segments connecting the nodes. The lanes are symbolized with a white line and the direction of each lane is indicated by the arrows.

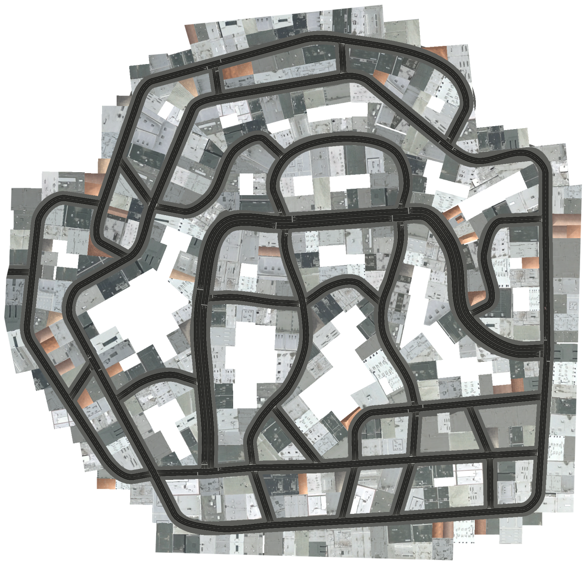


Figure 5: Imported city with 8896 m of roads, sidewalks, and 519 buildings.

calls could be drastically reduced with Unity's built-in occlusion culling. The physics calculation for other cars is only activated when they are closer than 60 meters to the driver's car. Communicating with the motion platform and calculating the shortest path to the next target are handled in separate threads to not block the main thread.

The 90 FPS refresh rate of the HTC Vive caps the frame rate of the simulation.

4 AI AND ADAPTIVE LEARNING

With a working environment of a driving scenario in place, we extended the simulation with a traffic simulation of Artificial Intelligence (AI)-controlled cars and implemented an adaptive learning system to train the driver.

4.1 AI Cars

In order to simulate lifelike behavior of other road users, the system must know where, when and how fast the computer generated cars drive. In this simulation, other cars are required to be able to follow a lane, automatically accelerate and brake, respect each other's right of way, and indicate where they go using their turn signal. With precise information about the position of each lane, it is easy to have other cars just follow the road. In order to allow AI cars to correctly handle turning at junctions, we extended the lane information by junction segments, connecting incoming and outgoing lanes at each intersection. Each junction segment is then assigned a unique priority, specifying which car can go first. While driving along the lanes and turning at junctions, all computer controlled cars obey the following rules:

Do not exceed the allowed speed. When driving, the cars accelerate up to a speed of 50 km/h, the allowed speed in the city.

Keep enough space to the car ahead. The gap between the cars is always big enough to safely come to a stop when the driver in front suddenly brakes. It is calculated from the car's current velocity and the configured braking acceleration.

Respect right of way. Before crossing a junction, yield to other road users that have the right of way.

Complete stop. Respect the same rules as on natural intersections, but come to a complete stop before driving onto the junction.

Stop at red lights. When on a red light, stop behind the signalization. After the light turns green, drive onto the junction but respect other vehicles who have the right of way (e.g. when turning left, yield to oncoming traffic).

4.2 Intelligent Tutoring System

The ability to drive a car involves skills in a set of activities. Clement et al. [3] show that the order in which activities are trained has an impact on the overall learning progress. Their proposed algorithm, ZPDES, optimizes the activity sequence based on continuous evaluation of a driver's skills. Activities are organized by exercise type and difficulty level. The Zone of Proximal Development (ZPD) defines a subset of activities that are expected to improve the user's skills when being trained. ZPDES updates this set based on the evaluation of each performed activity and selects the next activity with the highest expected learning progress.

In this section, we will show how we created an ITS by adapting the the ZPDES algorithm to the task of car driving, how a driver's skill in various activities is continuously tested, and how the activities chosen by the ITS are presented.

Activities. During a discussion with a professional driving instructor, we assembled a list of abilities that define a good driver. While we agreed that automatically deciding whether a person is proficient in car driving is not possible in an artificial environment, we were able to define a subset of these abilities that make sense to be trained on a driving simulator as they can be tested under controlled conditions and evaluated by the system:

Stable driving (on straight roads). The driver maintains a stable track with only little variance in the distance to the center of a straight lane.
Evaluation: When on a lane segment, the distance between the car's position and the closest position on the lane segment is recorded every meter. The recording starts a certain distance d after the start node of the segment and ends d before the end node to ignore deviation from the lane caused by turning. After enough samples have been recorded, a score between 0 and 1

is given based on the variance of the samples (lower variance yielding a higher score).

Stable driving (on curved roads). This activity has the same objective and uses the same technique of evaluation as the *Stable driving (on straight roads)* activity but is tested on curved roads for increased difficulty.

Turning. Execute all steps required to properly turn the car at a junction (check mirrors, look over shoulder, set indicator).
Evaluation: When approaching a turn, check head rotation and indicator state. A score of 1 is rewarded when all steps have been executed. Failing to set the indicator reduces the score by 0.5, not checking the mirror by 0.3, and a missing shoulder check by another 0.2.

Complete stop. Bring the vehicle to a complete stop at a stop sign.
Evaluation: When passing a junction from a road signaled with a stop, the vehicle must be completely stationary within a certain distance before the stop line. This activity is graded in a binary way, yielding either 1 or 0 points.

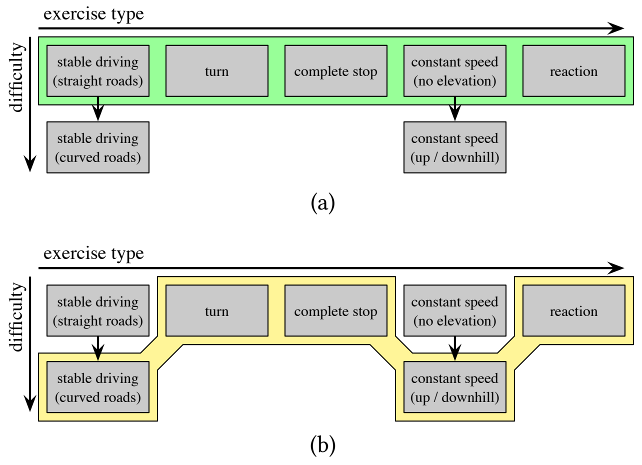
Constant speed (without elevation). The driver maintains constant speed throughout a lane segment on an even road.
Evaluation: In a fixed time interval t, the car's velocity is recorded. If the speed needs to be reduced due to traffic driving slower, the recorded sample is discarded. Similar to the stable driving activity, the variance is calculated to give a score between 0 and 1. If the end of the lane segment is reached without collecting enough samples, the activity is aborted and not scored in order to prevent incorrect evaluation.

Constant speed (on up or downhill roads). This activity has the same objective and uses the same technique of evaluation as the *Constant speed (without elevation)* activity but is tested on either ascending or descending roads.

Reaction. React to a vehicle unexpectadly crossing the driver's path.
Evaluation: A computer controlled car is positioned at a junction crossing the way of the driver's vehicle. The car then pulls out to provoke a collision if the driver does not react quickly. The score is calculated from the time between the other car starts moving and the moment the driver hits the brakes.

Adaptation of the ZPDES Algorithm. In a first step, the activities are structured into the activities graph, as depicted in Fig. 6, which orders them by exercise type and difficulty level. The two stable driving activities, as well as the constant speed activities, are connected as they are activities of the same type. The second version of each is considered more difficult, which is why they are positioned at a higher difficulty level in the graph. Amongst all other activities, the difficulty level is the same. When the simulation starts, all activities of the lowest difficulty level are included in the ZPD while the more difficult ones are excluded.

Selecting the Next Activity. Based on the result of previously solved activities, ZPDES suggests the next activity to be chosen. Each activity can be tested at various locations on the map. In order to complete as many activities as possible within a given time, the closest instance of the activity should be found. Given an activity and the current position of the driver, the distance of the shortest path to each activity instance is determined using Dijkstra's shortest path algorithm [4] and the one with the minimal distance is set as the target on the virtual satnav. Once the path to the chosen activity is known, all activity instances on the way can be tested, gaining additional information about the driver's performance.



exercise type
difficulty
stable driving (straight roads)
turn
complete stop
constant speed (no elevation)
reaction
stable driving (curved roads)
constant speed (up / downhill)
(a)
exercise type
difficulty
stable driving (straight roads)
turn
complete stop
constant speed (no elevation)
reaction
stable driving (curved roads)
constant speed (up / downhill)
(b)

Figure 6: Activities graph with initial ZPD (a) and updated ZPD where the more complex activities have been activated (b).

5 EVALUATION

In order to evaluate the driving simulator, we conducted a user study. The goal was to determine if the overall quality is high enough, to validate if the simulator can be used for future experiments, and to collect user feedback for possible improvements.

Experiment Design. We invited a total of 17 participants from various departments, as well as people not related to our facility. All of them had a valid driving license and were familiar with the Swiss traffic rules. 5 (29.4%) of the participants were female and the average age was 29.5 years (SD: 8.3 years). After being instructed how to operate the vehicle, the participants were asked to wear the HMD and drive through the virtual city for 15 minutes, following the directions given by the satnav. A simulator sickness questionnaire [5] was filled in before and after the driving session. A presence questionnaire [11] was answered after the test run. The drivers were told to immediately abort the experiment as soon as they experience any kind of discomfort.

5.1 Results

4 participants (23.5%) aborted the run due to symptoms of simulator sickness. The rest managed to complete the 15 minutes run without experiencing major discomfort.

Presence. In a total of 22 questions, the participants indicated how strongly they experienced presence in the simulator by giving values between 1 (not at all) and 7 (completely). These questions were then evaluated and converted into a scoring scheme in 7 subscales as shown in Fig. 7. The overall presence is on a satisfactory level. Of the measured criteria, the quality of the *sound* effects yields the lowest score, which we figured we can improve in a future version. A relatively high score on the *possibility to act* subscale suggests that the simulated behavior of the car resembles a real-life car to a level that enables users to control the vehicle through the virtual scene.

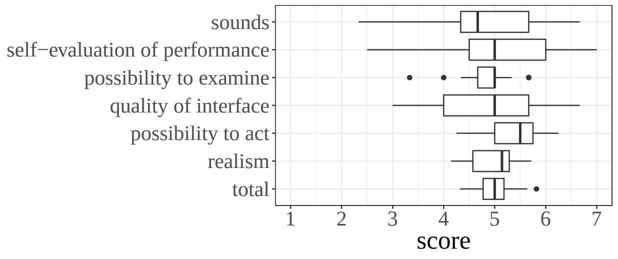


Figure 7: Evaluation of the presence questionnaire (N = 17). A score between 1 and 7 is calculated for each of the subscales. The median is marked with a bold black line. The boxes show the 25th and 75th percentile. The whiskers are limited at 1.5 IQR and outliers above or below are symbolized by black dots.

Simulator Sickness. The participants indicated how strong they experience discomfort by assigning values (0: none, 1: slight, 2: moderate, 3: severe) to 16 symptoms. The same set of questions was asked before and after the test run. Evaluating the simulator sickness questionnaire summarizes symptoms on three subscales: Nausea, oculomotor disturbance, disorientation, and a total score. Fig. 8 shows the results of the questionnaire before and after the test subjects participated in the driving activity. The most considerable increase in discomfort is reflected on the *nausea* subscale. This can be explained by the fact that, with our hardware setup, real acceleration cannot be experienced. The acceleration imitated by tilting the platform pretends that the user is moving but a discrepancy between the visual representation and the physically perceived movement remains, which for some people leads to feeling nauseous.

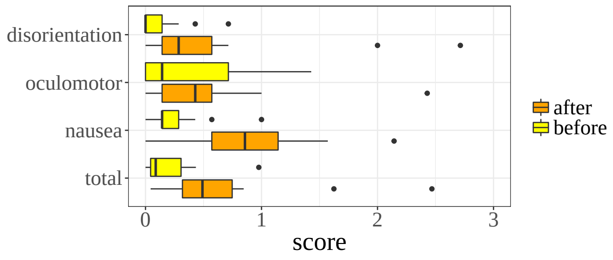


Figure 8: Evaluation of the simulator sickness questionnaire (N = 17). A score between 0 and 3 is calculated for each of the subscales before (yellow) and after driving (orange). The median is marked with a bold black line. The boxes show the 25th and 75th percentile. The whiskers are limited at 1.5 IQR and outliers above or below are symbolized by black dots.

Lessons Learned. We interpret the results of the presence questionnaire as an indication that the behavior of our simulation resembles a real-life car to a level that enables users to control the vehicle through the virtual scene without major difficulties. The simulator sickness questionnaire revealed that driving through the virtual environment leads to a slight increase in discomfort for test runs below 15 minutes. Future experiments should therefore be designed to last only for short periods of time. Applied to the objective of improving car driving skills in a virtual reality environment, this suggests a setup with a series of many shorter training sessions rather than few long ones.

6 CONCLUSION

We have shown how VR technologies can be applied to create an immersive car driving experience. We explained how physical properties influencing the driving characteristics of a car can be simulated, and presented in a way to model the behavior of a car's engine and transmission system. Our software connects to a 6 DOF motion system to simulate acceleration while driving, and queries input devices in the cockpit mockup, controlling the virtual car. AI cars drive through the city, follow the existing traffic rules, and interact with each other, as well as with the user's car. We presented five different types of driving-related activities that can be trained and automatically evaluated through an ITS. By adapting the ZPDES algorithm for car driving, we have shown how a personalized teaching sequence can be generated. Challenges involving limited computational performance, integration of motion hardware, and efficiently simulating city-wide traffic could be addressed. A user study revealed that most users experience a good level of presence in the virtual world and are proficient in operating the car on the VR driving simulator.

Future Work. The information gathered from the user study provides useful information when setting up further experiments. With a future user study, we aim to evaluate how strong the ITS' effect is on the learning progress. The content generation framework we provided can be used to create driving environments tailored to specific requirements posed by future research in VR car driving.

REFERENCES

[1] S. C. Augusto, A. C. A. Mol, P. C. Mol, and D. S. Sales. 2009. Using Virtual Reality in the Training of Security Staff and Evaluation of Physical Protection Barriers in Nuclear Facilities. *International Nuclear Atlantic Conference* (2009).
[2] D. R. Berger, J. Schulte-Pelkum, and H. H. Bülthoff. 2007. *Simulating believable forward accelerations on a Stewart motion platform.* Technical Report 159. Max Planck Institute for Biological Cybernetics.
[3] B. Clement, D. Roy, P.-Y. Oudeyer, and M. Lopes. 2015. Multi-Armed Bandits for Intelligent Tutoring Systems. *Journal of Educational Data Mining (JEDM)* 7, 2 (2015).
[4] E. W. Dijkstra. 1959. A note on two problems in connexion with graphs. *Numer. Math.* 1, 1 (1959), 269–271.
[5] R. S. Kennedy, N. E. Lane, K. S. Berbaum, and M. G. Lilienthal. 1993. Simulator Sickness Questionnaire: An Enhanced Method for Quantifying Simulator Sickness. *The International Journal of Aviation Psychology* 3, 3 (1993), 203–220.
[6] O. Kreylos. 2016. Analysis of Valve's 'Lighthouse' Tracking System Reveals Accuracy. Available online at https://www.roadtovr.com/analysis-of-valves-lighthouse-tracking-system-reveals-accuracy/. (2016).
[7] B. Lang. 2017. New HTC Vives Weigh 15Launch. Available online at https://www.roadtovr.com/htc-vive-weight-15-percent-lighter-than-original-headset-vs-oculus-rift-comparison/. (2017).
[8] K. Reif. 2014. *Fundamentals of Automotive and Engine Technology.* Springer Fachmedien Wiesbaden. 15–21 pages.
[9] T. Shibata and H. Fujihara. 2006. Development of Railway VR Safety Simulation System. *Quarterly Report of RTRI* 43, 2 (2006), 87–89.
[10] Digital Trends Staff. 2017. Spec Comparison: Does the Rift's Touch Update Make it a True Vive Competitor? Available online at https://www.digitaltrends.com/virtual-reality/oculus-rift-vs-htc-vive/. (2017).
[11] B. G. Witmer and M. J. Singer. 1998. Measuring Presence in Virtual Environments: A Presence Questionnaire. *U.S. Army Research Institute for the Behavioral and Social Sciences* 7, 3 (1998), 225–240.
[12] P. Zal. 2015. 2015 Fiat 500 1.2 engine Horsepower / Torque Curve. Available online at http://www.automobile-catalog.com/curve/2015/2182295/fiat_500_1_2.html. (2015).