A Multifunctional VR-Simulator Platform for the Evaluation of Automotive User Interfaces

Tony Poitschke, Markus Ablaßmeier, Stefan Reifinger, and Gerhard Rigoll

Technische Universität München, Institute for Man-Machine-Communication, Theresienstrasse 90, 80333 Munich, Germany
{ablassmeier, poitschke, reifinger, rigoll}@tum.de

Abstract. This contribution presents an approach for a scientific driving simulator platform for the representation and evaluation of new driver assistance and information systems. We will give a short introduction to the used gaming engine and then present the developed components of our new driving simulation platform. For a most realistic driving behavior, we implemented an own physics-component basing on a real vehicle model. Further, the platform provides the opportunity to easily create new scenarios depending on the test setup.

Keywords: automotive, usability, driving simulation, evaluation, virtual reality.

1 Introduction

Nowadays, the vehicle is getting much more than only a mere transport medium. Particularly in cars of the premium segment and increasingly in middle class vehicles, familiar technical systems for comfort have been integrated. For example, for a long time cell phones and navigation systems have been very popular in cars and even TV, mp3-player, and internet are already in the equipment lists of some manufacturers. Also, this multiplicity of new multimedia applications must remain operated by the driver.

Therefore, evaluation of new driver information and assistance systems regarding their road safety, adequacy, acceptance, and simple operability, and learnability is of particular importance. For such usability studies two major techniques exist: evaluation under field conditions using a real car, and computer-assisted test series using a driving simulator. The advantage of a computer simulation is in the fact that test conditions almost remain constant for all test subjects and the results can be easily logged. Also, such test series accomplished in laboratory environments are exactly reproducible, which facilitates the scientific evaluation. In addition, not only costs can be minimized but also health risks for test subjects can be avoided. In order to meet to the constantly growing demands on such computer driving simulations we will present our proprietarily developed simulation platform in the following.

2 Motivation and Previous Work

For the evaluation of newly developed driver information and assistance systems as well as appropriate displaying and control concepts in the context of laboratory usability studies computer-based driving tasks are needed. Therefore, most diverse approaches can be used. These range from simple control task, simple driving simulations [1], lane keeping and/or switching tasks [2] up to immersive traffic simulations represented in 3D [3].

The market for scientifically used driving simulations is capacious (e.g., [4], [5]). A large disadvantage of these usually commercially distributed software platforms is in the fact that appropriate source code is usually not freely accessible. Therefore one cannot access different settings and parameters around to adapt these at the examined system and respective scenarios. As no appropriate extensions can be developed, the simulations cannot be simply integrated into own frameworks. A further disadvantage of these commercial platforms is caused by the usually very cost-intensive operation (e.g., acquisition and maintenance).

Very much basic research was already done at our institute within the domain of automotive usability engineering [6], whereby most diverse frameworks were developed, e.g. [7]. As our former driving simulator proposed in [1] could not any longer deal with the concerns of modern man-machine interfaces (MMI) in the vehicle the simulation platform presented in this contribution was developed.

The functional requirements gathered before the implementation covered amongst other the following points:

- convenient software modules
- very good visualization of the test scenarios
- increase of the subject's immersion
- use of off-the-shelf hardware (e.g., gaming steering wheel)
- arbitrarily scalable test setup (e.g., desktop evaluation vs. multiple screen projection of the driving simulation)
- support of standardized solutions from other driving simulations to a large extent (e.g., analysis of lane keeping in LaneChange [2])
- arbitrarily expandable functional range (e.g. integration of driver assistance systems such as ABS)

On the basis of these points we finally decided for the use and extension of the extremely high performance computer game engine Unreal Engine 2.0, which is used for example in the computer game Unreal Tournament 2004 [8]. This offers an editor and the possibility for the extension and/or modification beside the impressing graphical engine.

3 Implementation

The following chapter will describe the used gaming engine and its components. Afterwards we will introduce our developed components (e.g., driving dynamics) and present our extensions (e.g., context server).

3.1 Software for our Driving Simulation

The simulator platform introduced in this contribution is based on the computer game Unreal Tournament 2004 (UT2004), and it's Unreal Engine in the version 2.0 [8].

This already caused a sensation in its first version of 1998, particular because of the outstanding displaying quality. By its modular structure, not only advancement by its creators, but also a modification of contents by players was possible. Further this engine permits the representation of (driving) scenarios in a three-dimensional way if special stereo beamers are used. This substantially increases the test person's immersion. The engine covers the following fundamental parts: graphics engine, sound engine, physics engines, and the Unreal Script interpreter.

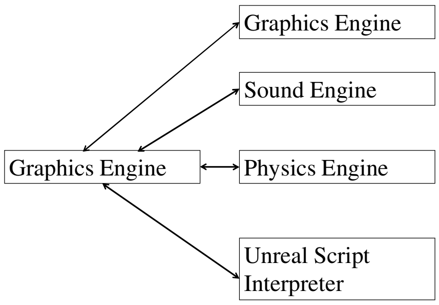


Fig. 1. UnrealEngine core and its core units

The *Graphics-Engine* is responsible for the representation of the scene on the output medium (e.g., monitor, beamer). Here, the entire states, variables, geometry data, and many other data are transformed into visually perceptible information. This process is accelerated by modern graphics cards that sufficiently computing time is available for other tasks. This engine manages all scene objects and only visualizes the objects in the player's field of view. Typical virtual environments consist of 150-300 visible objects and approx. 50,000 to 150,000 triangles. These values are only limited by the available computing power.

All signals for the acoustic output and feedback are processed in the *Sound-Engine*. Playing of sound effects is possible like kicks or jarring tires. The audio source is positioned in the 3D-scene using the unreal editor and the sound engine computes the correct sound from the player's position. For this, the tone can be put out in Dolby Digital Surround and the EAX 3.0 technology by Creative Labs is supported.

In the reality, movements and reactions between objects are based on the laws of physics. For example if a ball or another object experiences an accelerating impulse, it begins to roll, and becomes ever slower by the rolling friction. For simulating such forces Unreal's *Physics-Engine* uses the Mathengine also called Karma. Unreal also uses this for the simulation of vehicle physics.

UnrealScript is an object-oriented script language similar to C++. It permits the extension and modification to the UnrealEngine. The *UnrealScript Interpreter*, comparably with the Java Virtual Machine, reads and processes files, which were previously generated with the unreal command line compilers from the source code.

The infrastructure is based on a client/server *Network Structure*. This is optimized on network games with several players. Therefore, an exchange of the conditions and positions of the objects must take place among the clients. Since these change rapidly while gaming Unreal uses the User Datagram Protocol (UDP) for data exchange.

For further information concerning the Unreal Engine or programming the Unreal Engine please see [9].

3.2 Overview

The entire system is developed as centralized client/server architecture. The dispatcher is located in the center of this architecture. It broadcasts all incoming messages to all connected modules. Using this architecture, both the steering wheel and the simulation platform communicate with one another as well as different evaluated systems. Further, the module which evaluates the situative environmental context (see chapter 3.5) uses this interface for its communication. Thus, a consistent data exchange between the individual systems similarly to a bus system in the vehicle is to be guaranteed.

The extensions and modifications of the engine described in the following were implemented in an own mutator. These are only one approach to realize own modifications in Unreal. Generally, this kind of code is to make only slight changes at current play contents. Besides, not only the programmer but also the user should possess the possibility of configuring the mutator up to a certain degree. A further extremely important characteristic, which should be always considered with the implementation, is the compatibility to other mutators. In unreal a basic class `mutator` already exists, which possesses all necessary basic functions. From the basis class new modules can be derived and arranged individually. In principle, only the server executes mutators. Exceptions are represented by the HUD mutators (Head-UP Display) that are used in our approach and only exist at the client side. These mutators substantially serve for the representation of important gaming information for the client. Using a HUD mutator the programmer can freely design the graphical output. If the user wishes the integration of such a module, the loading procedure of a scenario is initiated with special parameters. Thus, the game engine knows which mutators should be merged [9].

3.3 Driving Dynamics

Since the engine is optimized on fun and gaming many changes had to be realized. Particularly for investigations a longitudinal and a transverse dynamics module were developed to extend the Karma physics engine towards a most realistic driving behavior.

Therefore, we integrated a gaming steering wheel via a TCP/IP socket interface. The control is realized by an own module, which was implemented in C++ in order to carry out only a small delay during the subsequent treatment of the control data (accelerator position/steering angle). The interface “DirectInput” from DirectX was used in order to query the states of the attached controller. The enormous advantage of this interface is the independence of the programming of the used joystick hardware. This is usually a joystick with steering wheel, gas and brake pedal as well as a button array, which is usable such as an indicator switch and for the control of in-car systems (e.g., multifunctional steering wheel).

3.3.1 Longitudinal Dynamics

The simulated longitudinal dynamics are themselves based on realistic values of a BMW 3 series (BMW 318i, [10]). Hence, acceleration and braking values are computed basing on real engine characteristics (torque and engine output dependent on the current number of revolutions) basing on the respective pedal position obtained from the joystick module. Values for rolling friction F_R, and air resistance F_A are simulated as well. The driving force F_D produced by the engine is calculated using several formulas. In general, it is calculated from

$$F_D = \frac{M_D}{r_{dyn}} \tag{1}$$

where M_D is the driving torque at the wheels and r_{dyn} the dynamic radius (of the wheels). Dynamic, because it is dependent on the additional load, the tire pressure and the driving speed. The driving torque M_D is dependent on several parameters (engine torque M_E (according to current number of revolutions given by the respective pedal position), transmission i_D, and efficiency μ_D of the drive train, transmission i_S, and efficiency μ_S of the drive shaft) and computed using formula 2

$$\mathrm{M_D} = M_E \times i_D \times \mu_\mathrm{D} \times i_S \times \mu_\mathrm{S} \tag{2}$$

These values were set up according to the output curve and the data sheet of the BMW 318i.

The rolling friction F_R works against the driving power and results from the friction of the tires on the underground and is computed from

$$\mathrm{F_R} = F_W \times \mu \tag{3}$$

The weight F_W, given by the product of mass and acceleration due to gravity (9,81m/s²), is multiplied by a coefficient of friction and results in the rolling friction.

Further, air resistance F_A is an unwanted but nevertheless unavoidable factor within the vehicle dynamics. The formula

$$F_A = \frac{A \times \rho \times v^2 \times c_W}{2} \tag{4}$$

uses the largest profile area A of the vehicle, the atmospheric pressure ρ, the wind and driving velocity and the air drag coefficient (c_W). For the atmospheric pressure the value 1,29 kg/m³ is taken.

Our current approach determines the resulting acceleration strength of the automobile from the driving force F_D. The rolling friction of the tires F_R and the air resistance F_A are taken off from this and we obtain the propelling force F_P. Hence, the corresponding acceleration can be obtained by dividing F_P by the passenger car mass. The product of the elapsed time since the last speed update and acceleration results in the change of speed.

3.3.2 Transverse Dynamics

The transverse dynamics are implemented by means of a dynamic steering angle transformation. Therefore, it is necessary to compute the turning radius dependent on the current steering angle and driving speed. The active steering element has already been used in mass production for several years. It adapts the transmission from steering wheel angle to a corresponding steering angle at the speed of the vehicle. E.g., it is possible to realize a large steering angle with a relatively small turn of the steering wheel at low and medium speeds (among other things also when parking).

We used this approach to integrate a low cost gaming steering wheel in our driving simulator. To convert the design-conditioned direct behavior of these steering wheels into a realistic manner its output values are differently lowered in dependence on the driven speed of the vehicle. Afterwards, these are converted into the corresponding steering angle for the driving simulator. Thus, a maximum steering angle of 10,5° results in the case of a speed of 0 - 40 km/h and a maximum steering angle of 3,5° starting from a speed of 80 km/h. Within the range of 40 - 80 km/h the maximum steering angle is linearly falling. The resulting maximum steering angles of 10,5° and/or 3,5° do not correspond any longer to the maximum steering angles with real vehicles. However, these values turned out to be acceptable and appropriately as they worked satisfactorily in the driving simulation.

In order to avoid inflated velocities in close curves, an implementation of the centrifugal force is needed. It is a force of inertia, which can be found in real traffic situations. It works against the centripetal force, which affects a vehicle with driving along curves and holds it in the trace. With the formula

$$F_{centri} = m_{car} \times \frac{v^2}{r} \tag{5}$$

the centrifugal energy can be calculated from the mass of the car, the square of the current speed v and the steering radius r. If the centrifugal force exceeds the value of the friction force, the vehicle and its corresponding driving trajectory get affected and shifted toward the centrifugal force. The direction of the centrifugal force results from the normal vector of the stamped steering radius. The new position of the vehicle is then computed and updated from the old position added with the scaled direction vector again. The feedback unit of the gaming steering wheel is used to deliver effects caused by the centrifugal force. Also different dynamic steering forces and feedback are generated dependent on the driven speed. Likewise, the feedback can be adapted to different roadway surfaces.

3.4 Surrounding Traffic

For a realistic simulation of various traffic scenarios a dynamic model for surrounding traffic was implemented. Primary purpose of the AI-controlled vehicles is the emulation of desired traffic conditions for usability studies. Therefore, an accurate reproducibility of the scenes and real time capabilities have to be ensured. The control of surrounding traffic was implemented as multi-agent system. Each simulated participant represents an intelligent agent. These take part in the scenario and cause desired situations, autonomously moved and coordinated by the mutator. The surrounding traffic vehicles are able to act with a certain amount of circumspection, e.g., by attending traffic signs, priority to right regulation, or a foresighted attitude.

Therefore, all vehicles are enabled to detect others and to prevent collisions. The initiated action to avoid a collision essentially depends on the currently driven on type of road and current traffic conditions. Therefore, first the ideal velocity and the actually measured speed are determined in each computation step. Further, each vehicle analyzes a spherical zone, which detects all vehicle objects in it. Additionally, two even sectors are stretched within this range in front of the vehicle. A rectangular area, which is directly in front of the vehicle is considered as critical zone. If another car enters the area the ideal velocity is decreased and full application of the brake is triggered. Once, no more obstacles are in the critical zone the vehicle is accelerated at the ideal velocity. However, the second range represents the warning zone, which has likewise dynamics in dependence of the speed. The entrance of other vehicles into this zone releases an overtaking attempt. Firstly, the neighboring roadway is examined with the shoulder check function. If the current situation does not permit an overtaking maneuver, then the ideal velocity is reduced dependent on the current distance to the ahead-driving vehicle.

The purposeful induction of different actions, e.g., sudden track switching or abrupt deceleration is implemented using trigger objects placed in the roadway. Dependent on the configuration of the respective trigger different actions of the vehicle are forced. Beside the changes of direction or being with certain road types here also abrupt decelerations and lane switching maneuvers can be induced.

3.5 Context Server

To use the different scenarios for the evaluation of new driver information and assistance systems a context management system was likewise integrated into the simulation platform. This permits to store and administrate the scenario's environmental data, e.g., coordinates of points of interest and roads. These are automatically extracted from the t3d-files (see chapter 4) describing a scenario (or a level). Also, further context data of the tested in-car systems are stored in the central data base. Thus, different systems of an experimental setup can request data from this data base, e.g., computation of navigation routes or query data from a message stack (e.g. error messages in on-board computer). It is possible to simulate close-to-reality conditions as in a modern vehicle. Depending upon setting of tasks, new displaying and interaction concepts can be evaluated comprehensively. The context manager communicates with the corresponding MMIs using the above mentioned TCP/IP network structure.

3.6 External Control

In order to ensure the same (reproducible) test conditions for all subjects, we integrated a remote control using the UsaWiz framework to control the simulation platform [7]. Thus, the simulation can be stopped and resumed at the same place again (e.g., intermediate questioning between test parts). Further the simulation can be resumed at arbitrary starting points (e.g., in order to fulfill the same task with same basic conditions again). Further, most diverse settings of speed limits, displayed assignments, control of surrounding traffic, lightning effects, sound reproduction, etc. are controlled using this interface.

3.7 Analysis of Test Data

During the test drives the following data are logged to a file: timestamp, currently driven speed, distance to the car in front, deviation from the ideal lane, the stamped steering angle, positioning data (sampling rate 100 Hz), and appropriate markings for the start and the end of individual test sectors and tasks.

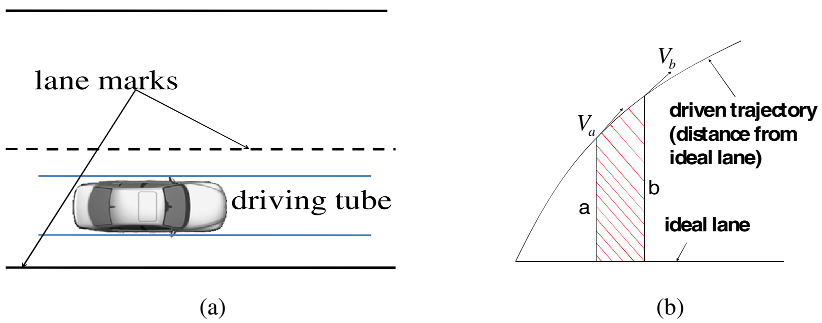


Fig. 2. (a) Driving lane with corresponding lane marks and the defined driving tube. (b) illustrates the computation of the area between ideal and driven lane during the time slot given by a and b.

For the evaluation of this logged data we are using a method similar to the scheme proposed in [1]. As the area between the ideal lane and the driven lane indicates the quality of keeping the lane, the different deviation events are integrated. Therefore, we defined a driving tube (width and direction of uncritical lane, see Fig 2a). If the test person crosses its borders, a deviation event is generated.

Also, a more significant deviation event is counted if the driver even crosses the lane marks. In order to differentiate and evaluate the individual deviation events further local and global areas were defined. Thereby, the local surface A_{local} describes an individual deviation from crossing the tube's boarder up to the reentry. The global area A_{global} describes the entire value of the deviation during the fulfillment of an entire test part or an entire task. Thus, we get the deviation value for a local area from (see Fig. 2b)

$$A_{local,i} = \frac{1}{2}\times(a+b)\times\sqrt{(\frac{1}{2}\times(v_a+v_b)\times\Delta t)^2-(b-a)^2} \tag{6}$$

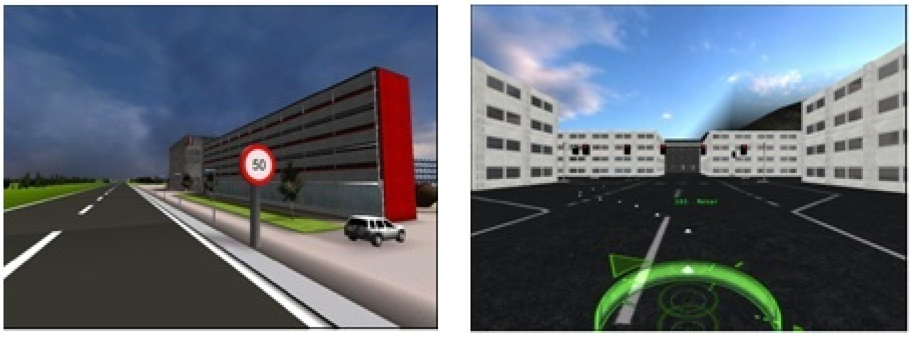
and the global overall area A_{global} as

$$A_{global} = \sum A_{local,i} \tag{7}$$

For the evaluation of test trips, a graphic tool was developed. This permits the test manager to analyze and evaluate the logged driving data. Therefore driving data as well as chronology of single control steps can be represented according to the time stamp. Thus for example correlations between a critical lane deviation and control steps can be analyzed.

4 Art Pipeline

The first step in generating new scenarios is the modeling of new contents (e.g. roads and houses). Also, these can be taken from already build scenarios.



(a) (b)

Fig. 3. Examples from two scenarios. (a) illustrates the campus of TU Munich in Garching near Munich. (b) illustrates a junction scenario with integrated Head-Up Display.

The modeling is accomplished using programs, which already offer an export filter for Unreal Tournament (e.g. Maya with a suited plug in). In addition, the objects are directly transferred into the UnrealEd, where they are used as StaticMeshes. Afterwards, the desired objects are placed within the 3D world and modeled to a scenario. These scenarios are arbitrarily expandable and adaptable at appropriate test setups (e.g. motorway scenarios, city traffic, cross-country) for the investigation and evaluation of different assistance and information systems. Figure 3 displays two sample views of our scenarios. Additionally, the UnrealEd provides the t3d file needed by the context server for the analysis of the environmental context.

5 Conclusions and Future Work

In this contribution we presented an approach for a low-cost implementation of a driving simulation using a gaming engine. In order to avoid gaming oriented vehicle handling we implemented an own module for driving dynamics. Basing on the modeled and analyzed 3D scenarios we implemented a context module. In order to evaluate logged driving data (e.g., driven speed, lane departure, etc.) we implemented a graphical evaluation tool.

Currently we are working on a synchronized multi-screen projection in a cave alignment to increase the immersion of the subjects. Also, we want to add further displays to our simulation vehicle to generate the view from the rear view mirrors. The sound engineering (e.g., engine sound, sound from the environment) needs some further improvement as well.

Acknowledgements

We'd like to thank the student workers who supported the work we presented in this contribution. Especially we'd like to thank Jürgen Gast, Stefan Moser, and Siu Lam Nghiem.

References

1. McGlaun, G., Althoff, F., Schuller, B., Lang, M.: A New Technique for Adjusting Distraction Moments in Multitasking Non-field Usability Tests. In: CHI 2002. Int. Conference on Human Factors in Computing Systems, Minneapolis, MN, USA
2. Kuhn, F.: Methode zur Bewertung der Fahrerablenkung durch Fahrerinformations-Systeme. World Usability Day 2005 Stuttgart (2005)
3. Interactive Driving Simulator (InDriveS), RWTH Aachen, http://www.zlw-ima.rwth-aachen.de/forschung/projekte/indrives/index.html
4. Wegmann, K.-M.: GmbH & Co. KG, http://www.kmweg.de/index.php
5. Systemtechnik, M.: GmbH, http://www.muellersystec.de/
6. Neuss, R.: Usability Engineering als Ansatz zum Multimodalen Mensch-Maschine-Dialog. Dissertation 2001, Technische Universität München (2001)
7. Schuller, B., Althoff, F., McGlaun, G., Lang, M., Rigoll, G.: Towards Automation of Usability Studies. In: IEEE International Conference on Systems, Man and Cybernetics (2002)
8. Epic Games, http://www.epicgames.com/
9. Unreal Developer Network, http://udn.epicgames.com/Main/WebHome
10. BMW Group, http://www.bmwgroup.com