Assessing In-Vehicle Information Systems application in the car: a versatile tool and unified testing platform.

Nicolas Louveton†, Roderick McCall†, Tigran Avanesov†, Vincent Koenig†‡, Thomas Engel†,

†SnT, Université du Luxembourg
4, rue Alphonse Weicker
L-2721 Luxembourg

‡ EMACS, Université du Luxembourg
Route de Diekirch
L-7220 Walferdange, Luxembourg

<firstname>.<lastname>@uni.lu

ABSTRACT

In this paper we present the DriveLab IVIS testing platform which allows for the same experiments to be conducted both under simulator and real car conditions. Other key aspects of DriveLab is that it is highly modular (therefore allowing the exchange or integration of different components) and that it supports more than one driver. For example we show that the same IVIS devices and scenario can be used with two different 3D engines. The paper provides a technical overview and a brief example of use.

Categories and Subject Descriptors

H.1.2 [**Information Systems**]: User/Machine Systems—*Human information processing*; D.2.2 [**Software Engineering**]: Design Tools and Techniques—*User interfaces*

Keywords

Driving simulators, Mobile applications, IVIS, Usability

1. INTRODUCTION

Infotainment applications for cars are taking an increasing place in the in-vehicle space [4]. As they offer new sets of functionalities and come on top of other on-board systems, these In-Vehicle Information Systems (IVIS) are likely to place an increasing cognitive demand on the driver [3, 5]. This requires an approach for evaluating new in-car infotainment systems both under simulation and real world setting where scenarios can be tested and evaluated using the same design analysis tools and evaluation method. These pressing needs have been identified in the I-GEAR project (outlined later) and developed under a platform known as DriveLab which allows for new scenarios and devices to be easily added to a common 3D (for now desktop) and in-car simulation platform for IVIS evaluation. In the following paper we present a full description of the testing and simulation platform being developed within I-GEAR.

AutomotiveUI'13, October 28 - 30 2013, Eindhoven, Netherlands

Publication rights licensed to ACM. ACM 978-1-4503-2478-6/13/10...$15.00. http://dx.doi.org/10.1145/2516540.2516576

We begin by explaining the background to the project, then discussing, comparing our ideas to existing solutions and finally explaining the technical aspects of the simulation platform making it both unified and versatile.

2. I-GEAR PROJECT

In-car infortainment systems, urban mobility and community-based platforms such as WAZE© (`http://www.waze.com/`) are growing in popularity. This is in part due to the increasing problems of traffic congestion in many major cities across the world. In order to help overcome traffic problems, the I-GEAR project [9, 10] aims to develop a mobile application that provides incentives for commuters to undertake alternative mobility behaviours by making use of game-like concepts such as leaderboards and challenges. As this application is likely to be used in the car, its development requires an extensive evaluation of its usability; this will be done first in a simulated environment and then on the road.

3. REQUIREMENTS FOR THE PLATFORM

Car driving simulators can take many different shapes ranging from low-cost and compact desktop set-ups to full fixed- or motion-based car cab simulators. In the same way the complexity and realism of the 3D environment, physics and traffic simulation can vary greatly. While high-profile driving simulators can improve immersion and fidelity, they are also expensive and space-consuming, making them not necessarily suitable for early and rapid prototyping of IVIS.

In turn, low-cost simulators may impair the perception of ego-motion, speeds and distances, yielding under-estimated speed and inter-vehicular judgements [7]. However, while high-fidelity simulation is required for complex driving scenarios (involving weather conditions, traffic characteristics etc.) and for driver's perception studies, their low-cost counterparts may be used efficiently for assessing dashboard ergonomics and simple driving tasks [7].

Indeed low-cost simulators, in spite of their lack of physical fidelity (no driving cab, narrow visual field and lack of motion) and virtual reality immersion (more or less realistic simulated environment) were proven to be relevant at early stage of IVIS developments [6]. In this comparative study, authors demonstrated a less safe behaviour in regard to inter-vehicular distances in a low-cost set-up compared to the high-profile one. However, results related to speed control and secondary task (interaction with the IVIS) completion time were found to be consistent across simulator

types as no significant effect was evidenced.

Given the early stage of the development of our application we decided to develop DriveLab, a compact yet scalable platform for testing in-car applications. Although, our platform is currently based around a desktop simulator, DriveLab implements a modular architecture (detailed in Section 4) that supports the integration of new components including more complex 3D/Physics engine, traffic simulators and real hardware/physical devices. Moreover, to the best of our knowledge there is no single platform which can be used to design a user study and run it within both a 3D simulation environment and a real car with minimal modification. This is where our approach differs as the car or 3D environment are viewed merely as a plug-in (with a range of limitations) from which data can be sent and received (see also section 5). As noted later this modular approach therefore makes it easier to construct studies, conduct tests and compare data across simulated and real environments. In the long run the goal of our platform is to serve both academic and industry research. Although our platform is not a design tool as such, our system is open to be connected to any interactive prototype as long as this prototype can send and receive data to/from DriveLab.

4. MODULAR ARCHITECTURE

A key requirement for our platform is to be highly modular and customisable. This means that it has to support e.g. new graphics engines as they become available, allowing us to change, add and remove hardware components such as tablet PCs and finally integrating external applications (e.g., mobile applications, monitoring software, eye tracking software). This has been achieved by implementing a plug-in architecture around the core control and communication part. Each plug-in provides specific features and a common programming interface that allows their connection to the main platform. The core application maintains an up-to-date model of the simulation states (cars telemetry and traffic data) and performs necessary communication between the different plug-ins. For example, data persistence relies on a plug-in encoding simulation states and events into a SQLite3 database. However, this type of data logging could be changed at any time by writing another plug-in (e.g., CSV files or MySQL). Plug-ins are represented by a directory containing a main Python script file exposing both the main plug-in class and an initialization function. Plug-ins are initiated at the start-up of the simulation platform and are launched and registered for the communication process (JSON packets). The fact that the main class should be written in Python does not prevent to use other languages: for example, our mobile application is written in HTML5/JavaScript (see Section 4.3). This system made particularly convenient the addition or substitution of heterogeneous external programs.

An additional important requirement for our platform exists in the abstraction of events and user-interaction in order to isolate them from the specific simulation environment in which they occur. Indeed, the game mechanics involved in our mobility application is a separated effort that needs to be portable across different testing environments. It is also implemented as a plug-in so as to make it independent from the 3D/physics engine for example or any other components. This game engine uses scripts written in plain Python and can provide access to simulation states and to user actions by mobile device so as to trigger e.g. specific events on the mobile application.

4.1 Flexibility: switching 3D/Physics engines

In a preceding version of our platform [2], we created a 3D map of Luxembourg city in order to rely on a realistic simulation environment. City data was drawn from OpenStreetMap while we used a 3D racing game (SpeedDreams 2) as 3D and Physics engine for the simulator platform. However, the preliminary usability studies of our mobile application do not require a comprehensive model of Luxembourg. Instead we thus use a car following task which is quite well specified [11] and often used in the context of in-car systems' usability evaluation [6, 1, 13, 12, 14]. For the same reason we integrated another 3D/Physics engine called OpenDS [8]. We chose this engine for two reasons: first, it is open-source and second it allows rapid scenario scripting through XML files. This made it a convenient choice for implementing a lead car speed changing scenario. However we plan to develop a complete API. It is worth to mention that using this engine required us to implement a few modifications only in order to send traffic data to our simulator platform. This allowed us to switch from SpeedDreams to OpenDS while keeping the same events logic in our mobile application.

4.2 Multi-Driver and Multiplayer

IVIS applications increasingly rely on data from more than one driver. DriveLab supports multiple cars/drivers (both real and simulator cockpits) as essentially each one is classed as a device. This approach allows for the exploration of social driving aspects and is of particular interest to our multiplayer "gamified" mobility experiences. As the data from each car and associated devices is logged, it gives the evaluators an easy way to compare what happens when a particular event is triggered that effects multiple drivers. For example if there is a road accident and information is provided by the IVIS it is possible to see how the drivers respond therefore allowing a direct comparison.

The server architecture also lets one trigger events from either the internal game logic engine (currently under development) or another external platform. For the latter one must add to the DriveLab a plugin which will transform messages received from the external platform to appropriate JSON packets processed by the server and the other way round. As with the rest of the platform such approach can be used to trigger events on the tablets or other IVIS devices. Furthermore, the DriveLab server can also send for example telemetry data to the game engine therefore allowing it to trigger events based on the up-to-date status of the various drivers. Our intention is to provide a skeleton game logic engine which supports human-factors analysis and gamification of driving which can be used to test basic concepts.

4.3 Rapid prototyping: employing mobile web technologies

One of our requirements for the platform is the ability to assess the usability of different kinds of user-interfaces in the car and to quantify the impact of these interfaces on the driving performance. In order to make the development process

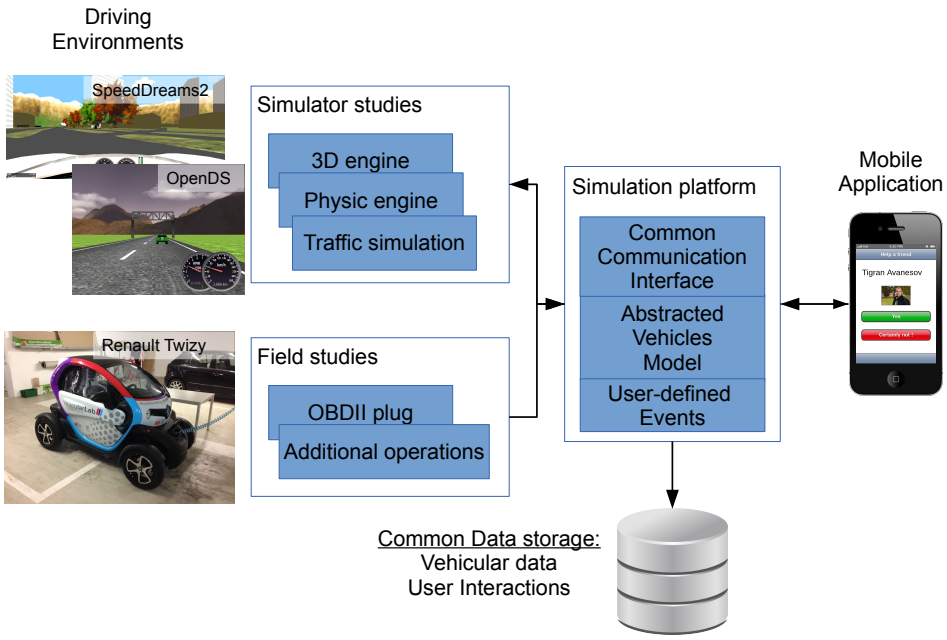

Figure 1: Plug-in architecture allowing usability testing of a mobile application in different driving environments.

faster and the resulting application more portable we use standard web and mobile technologies such as HTML5 and JavaScript. The connection with the simulation platform relies on the use of WebSockets. The game engine triggers events as a function of user actions and simulation states in order to display specific screens and tasks on the mobile phone. The ability to use interactive prototypes along side to independant modules shows that designers can take advantage of the modularity of DriveLab in their workflow. Once the experimental script developped, designers can then focus on user-interfaces and iterate quickly on their prototype.

5. STUDIES SET-UP AND MONITORING

As we want to use a car following task scenario as a standard task we developed tools around the simulator platform in order to generate this kind of driving scenario and easily run them. The experimenter can generate car following scenarios with a couple of options such as length of the trial, number of speed changes, randomization of lead vehicle behaviour and decoration along the road. The trials are managed by another application allowing the experimenter to set-up participant's profile and choose among available driving scenarios. We are in the process of building tools for automatic data analysis and reporting based on common driving performance and usability measurements (see an example in Fig. 2).

As known from PC user testing evaluation platforms we also offer the ability to remotely monitor the current car status (either a real car or the 3D model) e.g. location, speed, steering wheel position, gear, use of the pedals and the cockpit view (at present provided as a direct camera feed outside of the evaluator console environment). This is done via the evaluator console which runs on a remote PC. As the platform evolves, the evaluator console will also support the triggering of remote events, for example triggering a change in the user interface on one of the in-car devices. In addition, it will also support real-time data information from devices such as eye trackers and provide statistics such as lane deviation etc.

The modularity of our approach allows us to go further and to replace the 3D simulation engine by a real driving context. Indeed, although we replace the simulated environment with a real car, we can still run our simulation platform with the same data logging, mobile application and game mechanics. The only change required is that the source of the realtime data for the platform has to be set to receive from the car bus. We plan to implement a new component of our platform using associated work of our research group which allows capturing car data from the Renault Twizy (see Fig. 1, bottom-left side of the diagram) and forward them to mobile applications through a Bluetooth connection. The Twizy itself uses a proprietary data format and we are also in the process of extending our platform to be able to make use of data from more traditional cars as well. Therefore, we will be able to run the same in-car application, the same interaction logic and the same data logging structure in real-world driving situations. It enables us to compare usability data coming from our simulated environment to data coming from real-world driving. Such a comparison will allow a progressive and consistent usability testing approach along the project lifetime. This connection to real cars is currently

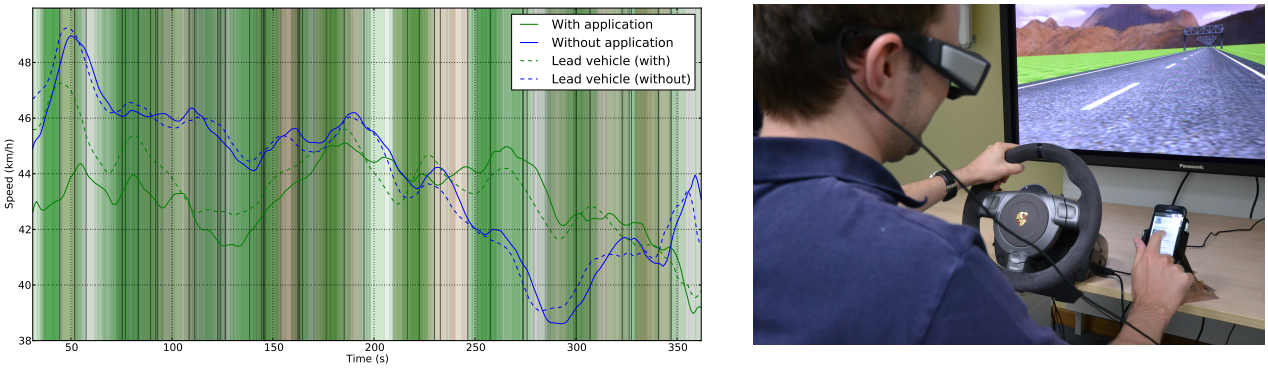

Figure 2: Pilot data (left panel) from our platform representing participant's speed (continuous lines) during trials with (green lines) and without (blue lines) the mobile application. Dashed lines represent the lead vehicle speed in the two conditions. The vertical color strips show when the participant actually interacted with the application and the correctness of the answer (green for correct answers and red for wrong ones. The simulator setup is presented on the right panel.

being implemented and will give rise to numerous challenges in terms of accuracy of sensors and real-time processing.

6. CONCLUSIONS

Our platform is an integrated way to quickly prototype and test user interfaces. It provides a convenient way to consistently test interaction mechanics between the user and the application in different contexts. Indeed, the 3D simulation engine can easily be switched without impacting the interaction mechanics with the application or the user study design. Finally, the 3D simulation engine can be replaced by a real car in order to compare data at different levels of fidelity with regard to the target situation.

7. ACKNOWLEDGMENTS

The I-GEAR project was funded by the National Research Fund, Luxembourg (Project code: 11/IS/1204159). We also thank other colleagues and students from the I-GEAR team (`www.igear.lu`).

8. REFERENCES

[1] H. Alm and L. Nilsson. The effects of a mobile telephone task on driver behaviour in a car following situation. *Accident Analysis & Prevention*, 27(5):707–715, 1995.

[2] T. Avanesov, N. Louveton, R. McCall, V. Koenig, and M. Kracheel. Towards a Simple City Driving Simulator Based on Speed Dreams and OSM. *Automotive UI in Adjunct Proceedings – Work in Progress Paper*, 2012.

[3] P. Burns and T. Lansdown. E-distraction: the challenges for safe and usable internet services in vehicles. In *Internet Forum on the Safety Impact of Driver Distraction When Using In-Vehicle Technologies*, 2000.

[4] T. Fishman. Digital-age transportation: The future of urban mobility. Technical report, Deloitte University Press, 2012.

[5] P. Green. *Driver distraction, telematics design, and workload managers: Safety issues and solutions.* Society of Automotive Engineers, 2004.

[6] S. L. Jamson and A. H. Jamson. The validity of a low-cost simulator for the assessment of the effects of in-vehicle information systems. *Safety Science*, 48(10):1477–1483, 2010.

[7] A. Kemeny and F. Panerai. Evaluating perception in driving simulation experiments. *Trends Cogn Sci*, 7(1):31–37, Jan. 2003.

[8] R. Math, A. Mahr, M. M. Moniri, and C. Müller. Opends: A new open-source driving simulator for research. *GMM-Fachbericht-AmE 2013*, 2013.

[9] R. McCall and V. Koenig. Gaming concepts and incentives to change driver behaviour. In *Ad Hoc Networking Workshop (Med-Hoc-Net), 2012 The 11th Annual Mediterranean*, pages 146–151. IEEE, 2012.

[10] R. McCall, V. Koenig, and M. Kracheel. Using gamification and metaphor to design a mobility platform for commuters. *International Journal of Mobile Human Computer Interaction (IJMHCI)*, 5(1):1–15, 2013.

[11] National Highway Traffic Safety Administration. Visual-Manual NHTSA Driver Distraction Guidelines for In-Vehicle Electronic Devices. Technical Report NHTSA-2010-0053, National Highway Traffic Safety Administration, Washington, DC: U.S., 2012.

[12] D. D. Salvucci. Distraction beyond the driver: predicting the effects of in-vehicle interaction on surrounding traffic. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 3131–3134, New York, NY, USA, 2013. ACM.

[13] D. D. Salvucci and K. L. Macuga. Predicting the effects of cellular-phone dialing on driver performance. *Cognitive Systems Research*, 3(1):95–102, 2002.

[14] D. D. Salvucci, D. Markley, M. Zuber, and D. P. Brumby. ipod distraction: Effects of portable music-player use on driver performance. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 243–250. ACM, 2007.