IPT-EGVE Symposium (2007)
B. Fröhlich, R. Blach, and R. van Liere (Editors)

High Ecological Validity and Accurate Stimulus Control in VR-based Psychological Experiments

M. Wolter[1], C. Armbrüster[2], J.T. Valvoda[1], and T. Kuhlen[1]

[1]Virtual Reality Group, RWTH Aachen University, Germany
[2]Institute for Psychology, RWTH Aachen University, Germany

Abstract

Virtual Reality (VR) has become a useful tool in psychological therapy, rehabilitation, and basic research. It provides the therapist or scientist with high ecological validity and complete control over multimodal stimuli and the virtual environment. To build lifelike virtual environments, commercial software or game engines are often used. On the other hand, many psychological experiments require an accurate timing of presented stimuli and user reactions, which is not easily obtained by modern hard- and software systems. In this work, we present a VR software for psychological experiments which combines modern computer graphics and Virtual Reality techniques with accurate stimulus and event control. We describe the system features and possibilities for VR-based experiment generation and propose approaches for precise stimulus and event timing. These approaches are evaluated for achievable accuracy in stimulus control and performance monitoring within typical virtual environments.

Categories and Subject Descriptors (according to ACM CCS): C.4 [Performance of Systems]: Measurement Techniques, I.3.7 [Computer Graphics]: Virtual Reality, J.4 [Social and Behavioral Science]: Psychology

1. Introduction

Virtual Reality (VR) has found its way into psychological research in the last decade. VR applications are used as research methods in general psychology, neuropsychology, clinical psychology, motor rehabilitation, and in various applied disciplines. Due to its advantages, which are high ecological validity (in short: the degree to which observed behavior in a study reflect the real-world behavior), high experimental control, high generalizability of experimental findings, high experimental realism, and the ease of implementation and conduction of experiments [LBB99], more and more areas of application were discovered especially in therapy, rehabilitation, and basic research. On the other hand Virtual Reality is examined as research topic. For example, basic research is concerned with the influences of 3D presentation, depth perception, visuomotor coordination, and the influence of visualization techniques on memorizing and encoding processes.

In [RK05], Rizzo et al. systematically assess strengths, weaknesses, opportunities, and threats to the field of VR rehabilitation and therapy. One weakness reported is the missing flexibility and compability of software tools. Platform compatibility concerns operating systems as well as applied hardware (trackers, input devices, etc.). Common users are only familiar with a specific operating system. The number of available VR hardware is restricted and existent systems should be useable for different tasks.

Furthermore, it has also been criticized that due to a lack of interoperable and flexible systems, the set-up costs for VR studies are too high [Riv05] [RK05]. Many applications are one-off creations and are restricted to a proprietary hardware and software. One way to shorten the set-up time is the usage of existent software which provides plenty of required functionality. In the field of psychotherapy, especially in exposure therapy, the usage of game engines has become common. Game engines provide a rich set of functionality to enable high ecological validity while maintaining a flexible interface to define the virtual world. This separation of content and functionality allows a use in different application areas [LJ02]. Functionally, game engines provide, among others, modern graphics techniques, multimodal output, physics, collision detection, particle systems, and hu-

manoids. The behavior of humanoids or objects as well as the layout of the environment can be defined on top of the game engine's core system. However, exact determination of the duration of a stimulus presentation, as well as measurement of the reaction times of users require changes in the engine's core, which is typically optimized for performance and not for accuracy.

Software toolkits that provide an accurate control of stimuli and measurement of reaction times are widely used for computer-based experiments in basic (neuro)psychological research. As accurate timing is a problem with modern multitasking operating systems, numerous software solutions are designed for the MS-DOS operating system or do not support Virtual Reality methods. Solutions for modern operating systems provide mostly primitive or two-dimensional stimuli only. In addition, to measure reaction times within millisecond accuracy, special hardware or additional software is required.

These two different types of software (i.e., game engines and psychological experiment generators) provide solutions specialized for the two major benefits of VR in neuroscientific experiments: high ecological validity on the one side, and high stimulus control and exact performance measurement on the other side. But, for accurate and valid experiments in virtual environments a combination of both types of software is required.

We introduce an in-between solution that has already been applied to a large number of (neuro)psychological experiments from different areas. The software system applied is called ReactorMan. It was designed to enable incomplex and flexible implementation of psychological experiments in virtual environments. In contrast to other software solutions, accuracy together with reusability were the main design goals right from the start. ReactorMan combines functions from modern computer graphics (scripting, avatars, shadows, large scene rendering), Virtual Reality (stereo rendering, VR hardware, acoustics), and software methods used in neuropsychological tools for precision and timing. It addresses commonly criticized weak points of today's VR software systems applied in psychological research. ReactorMan allows for the design and execution of accurate psychological experiments inside virtual environments.

The paper is structured as follows: in section 2 we will give a short overview about existing software systems used in neuroscientific experiments and VR-based studies. Thereafter, we give an overview of the ReactorMan system and its features in section 3. Section 4 illustrates the design of experiments using ReactorMan. The following section 5 discusses issues for accurate timing within virtual environments, which are evaluated in section 6. In section 7, we discuss the presented approaches and achieved results from a psychological point of view, followed by a conclusion and an outlook for future work.

2. Related work

The measurement of accurate reaction times in computer-generated psychological experiments is vital for many research studies. Several solutions are published or available within commercial software. The TAP software package [ZF02], available for MS-DOS and recently Windows systems, provides a rich set of attention-related tests. Most of these tests are reaction-time tasks with low complexity allowing the evaluation of very specific deficiencies. The current version uses a special stripped down Windows platform in combination with parallel port attached response devices to achieve high measurement accuracy. Tscope [SLVV06] is a C library providing timing functionality as well as basic graphics and sound. Experimenters can use Tscope to program their own experiments in C code. While this allows for very flexible experiment implementation, programming and creating efficient software is not easy for novice programmers. The DMDX software [FF03] uses DirectX for accurate video and audio control. The system generates frames in advance to avoid displaying delays. DMDX is written for Windows systems and therefore uses specialized functions provided by the operating system to achieve accuracy. Several other commercial experiment generators like E-Prime, Presentation, etc. are available. Unfortunately, most commercial experiment generators do not provide source code for independent testing and evaluation of their accuracy.

Concerning the field of VR-based psychological research, a large variety of different projects exists. A good overview of several fields of social and behavioral psychology can be found in [Riv05]. The Virtual Classroom software targets at assessment and treatment of attention processes, especially ADHD (attention deficit hyperactivity disorders) in children. The user sits inside a virtual classroom using an HMD and has to fulfill attention tasks. The experimenter can control different types of distractions (cars outside, people entering the room, noise, etc.), and the user's performance values are recorded. The later version of the Virtual Classroom [RBB*06] is based on the Unreal game engine for more sophisticated graphics. The authors mention the rapid prototyping possibilities as well as the drawbacks, which include heavy modifications on the engine to add required functions. The system was later used for other tasks like the Stroop test, and enhanced to support room-mounted display systems. Hoffmann et al. [HPM*04] created SnowWorld, a virtual environment applied in clinical pain control. The system is combined with fMRI (functional magnetic resonance imaging) to measure pain-related brain activity. The virtual world is created with the VirTools software. Furthermore, the virtual city of Tübingen [MOB01] is used to assess human navigation and orientation abilities. Here, modern graphics cards techniques were applied to interactively display and navigate inside an environment consisting of high-quality textures.

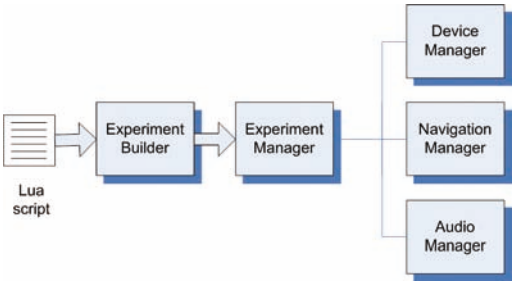

Figure 1: *Scheme of ReactorMan's software components. Main part is the Experiment Manager, which mediates information from all other managers.*

The NeuroMan software focuses on neuropsychological experiments in virtual environments [VAD*03]. The experiment is defined in a scripting language and presented in an interactive virtual environment. The system records user interactions for off-line analysis, and enables accurate reaction time monitoring using special hardware. NeuroMan was successfully applied in several projects to study depth perception [AWV*06], psychomotor behavior, brain activities while manipulating virtual objects, sleep deprivation [HVK*05], and several other (neuro)psychological problems.

The described systems provide solutions specialized either for accurate stimulus presentation and monitoring or immersive virtual environments. Some are created for specific psychological questions or therapies, while a few enable flexible definition of the analyzed scenario. Besides NeuroMan, none of these systems addresses timing accuracy combined with Virtual Reality techniques.

3. System overview

The ReactorMan system is an extension of the NeuroMan framework [VAD*03]. Based on this basic framework, the system has been enhanced to modern graphics features and a broad field of study and therapy functionality. ReactorMan bases on open software packages: OpenSG is used for scene graph management and rendering, OpenAL is applied for auditive stimuli. Compared to most commercial experiment toolkits using DirectX, the usage of OpenGL and OpenAL provides a very portable solution running on several operating systems. For Virtual Reality functionality, ReactorMan bases on the VR toolkit ViSTA and its multimodal interfaces [ALK05]. The animated virtual humanoids are based on the h-anim standard as provided by the VRZula module of ViSTA [VKB06]. Experiments are designed and controlled by the Lua scripting language.

The integration of open software modules allows for an easy deployment process, a good system portability, and it is essential for accurate system control as will be explained in section 5. As portability between display systems and operating systems is a basic principle of ViSTA, ReactorMan also runs on different platforms and was already applied with different VR display systems (e.g., HMDs, PowerWalls, a Workbench, a CAVE-like environment), tracking hardware (e.g., Flock of Birds, Polhemus, A.R.T. and Qualisys optical tracking) and special VR hardware (e.g., different datagloves, spacemice). The flexibility of the ReactorMan software counteracts the compability weakness of other software systems mentioned by [RK05].

The software scheme is depicted in figure 1. The decomposition into software modules allows for easy expansion and enhancement of functionality. Core module is the Experiment Manager (EM). It manages the execution and content of the current experiment. The experiment itself is created by the Experiment Builder, which parses the Lua file describing the experiment. The experiment design is explained in more detail in section 4. The functions of the three remaining managers are straightforward: special devices (i.e., devices not handled by the VR system), navigation metaphors, and auditive stimuli. All three managers communicate only with the EM, changing or reacting to the current experiment. Special devices include the hardware reaction time device mentioned in section 5.1, custom-built devices using the parallel port for input or output, and a device for synchronizing the experiment with an external fMRI trigger signal.

For stimulus presentation, besides basic primitives and geometry formats, ReactorMan provides several possibilities typically not found in experiment generator software. The OpenSG toolkit integrates several modern computer graphics techniques. For example, shadows in virtual scenes are an important depth cue, but are not supported by most experiment toolkits. In addition, OpenSG provides methods for efficient rendering of large scenes and complex models.

Virtual humanoids can be used either as animated agents that populate the virtual environment, or as avatars (i.e., virtual user representations). One main feature of avatars in ReactorMan is motion tracking. The user is configured with sensors, or special input hardware, like data gloves, whose motion is mapped on the kinematics of the virtual humanoid.

For accurate sound generation within virtual environments, the ViSTA binaural acoustics system [ALK05] is integrated. Arbitrary sound can be generated even at head-near positions for a moving listener. Only two loudspeakers and a tracking system are needed to apply this system.

4. Experiment design and execution

The separation of content and functionality is a concept found in all modern games. A game engine provides functionality, while the content is out-sourced in runtime interpretable scripts. ReactorMan uses the same concept by applying the Lua scripting language. Lua is a small, fast, and easy to learn language. For example, variables in Lua are

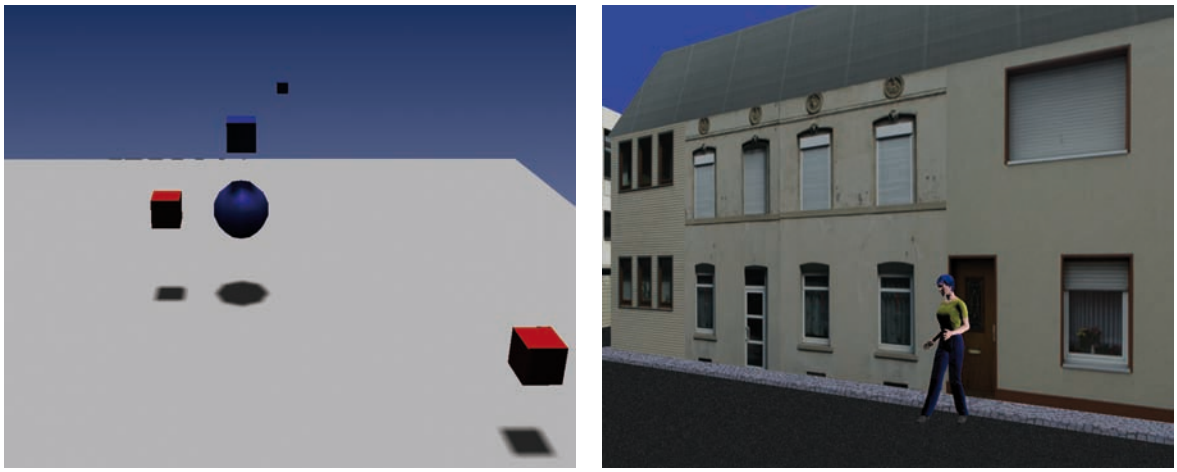

Figure 2: *Screenshots of typical ReactorMan scenes. Left: Basic objects with shadows. Right: Moving avatar inside a city.*

dynamically typed, which is an easy to understand concept for novice programmers. In addition, we do not stub the ReactorMan software core completely, but instead provide access to the ReactorMan core only at dedicated control points. While this prohibits the experiment designer to use every function at will, at the same time, comprehension for novice programmers is improved. In addition, the restricted control flow allows for higher performance, as all time-critical and complex tasks are done and measured in the core system.

To further increase the design convenience, the experiment designer can make use of a special ReactorMan language on top of Lua. This language consists of a restricted vocabulary commonly used by psychological experimenters (e.g., terms like sessions, blocks, trials, or stimuli). Most basic experiments can be built using this restricted language. At any time, the full Lua language can be applied to implement more sophisticated sequences.

Figure 3 shows a small example: The scene *stimulus* contains a geometry and a moving h-anim humanoid called *judy*. A new, unnamed trial is created which presents the *stimulus* scene for 500 ms at a predefined position *position_left*. Additionally, on each event of the device named *HEAD* (which is the head's tracker sensor), a function *validate_users_view_direction* is called.

The experiment script is split into two basic parts. First, the experiment setup describes the experiment sequence and the definition of the virtual world and its objects (in the example: *NewTrial*, *DoScene*, *AddGeometry*, *AddAvatar*). This includes the definition of a world, through which the user can navigate, a definition of visual and acoustic stimuli, including virtual humanoids, and the breakdown of the experiment into repeatable sessions, blocks, and trials. Second, the implementation of control points allows for dynamic behavior during execution, that is the reaction on user or system events (in the example: *RegisterEvent* and the Lua function *validate_users_view_direction*). Typical user events are movements or button presses, common system events include timers or collisions. Figure 2 shows screenshots of exemplary scenes built with the ReactorMan language.

```
-- ReactorMan example
stimulus = NewScene ()
    AddGeometry ("table.wrl")
    judy = AddAvatar ("judy.wrl",
           "judy_greeting.bvh")

NewTrial()
DoScene (stimulus, 500 ms, position_left)
RegisterEvent ("HEAD",
    "validate_users_view_direction")
```

Figure 3: *Example of the ReactorMan scripting language. A simple trial with a geometry and a moving virtual human is created. Events of the head sensor are processed by a Lua function named "validate_users_view_direction".*

5. Accuracy of measurements

The measurement of reaction times is crucial for certain psychological experiments. Reaction time is defined as the time span between the appearance of a multimodal stimulus and the reaction of the user. As the experimenter is interested in induced reaction time differences, the measurement should be as accurate as possible. The accuracy of a reaction time depends on the exact stimulus presentation time as well as the exact detection of the user's response. With the growing functionality and complexity of today's computer systems, the measurement of these points in time is not trivial for several reasons:

- Modern operating systems may interrupt the application or one thread of the application for the execution of other kernel or user programs. I/O operations like memory swapping or file access may block the application, too.
- Underlying software layers may introduce latencies. This includes used toolkits (e.g., scenegraphs, audio systems, haptics systems) or driver software. A detailed knowledge of the guaranteed functionality of applied software is therefore needed.
- The hardware components themselves may have a high or variable latency. For example, CRT monitors are more suitable for accurate presentation of visual stimuli than LCD displays, as some LCD displays need 10-20 ms to reach full brightness [PHT04].

To provide accurate timings, special care in the development of measurement software has to be taken. ReactorMan tries to avoid any file I/O during execution, that is loading stimuli as well as writing log files. High frequent or blocking functions are executed in own threads [VAKB04].

While careful software design assists precise timing, no guarantees about timing precision or validations of reaction times can be made. Without statements about measurement accuracy, the system is not suitable for many psychological problems. The ReactorMan system provides two approaches to solve the timing accuracy problem: a hardware-based solution that uses external sensors and a software approach. Both solutions are applicable in virtual environments.

5.1. Hardware timing

The most accurate solution is to use special hardware that measures the physical appearance of a stimulus and the user's reaction directly. Without inaccurate or unknown computer hardware or software in-between, a high accuracy can be guaranteed. [VAKB04] presented such a solution for the ReactorMan software. This hardware system measures reaction times for visual stimuli using an optical sensor and button presses to a special device with an error of less than 0.01 milliseconds.

The system has been enhanced by additional stimuli sensors and reaction devices. First, the optical sensor can be replaced by an acoustical threshold hardware that reacts to a certain amount of noise produced by loudspeakers. Second, the reaction buttons may be exchanged by a proximity sensor. The user has to touch the sensor, which signals as soon as the touch ends. This device is used for experiments where users wait in a predefined position for stimuli and directly react to them (e.g., grasping or reaching for visual stimuli).

Drawbacks of the hardware solution are the efforts for building and maintaining the hardware, as well as their limited application to certain display and interaction devices. The reaction buttons are custom-built, which makes it difficult to use existent interaction devices like wands or spacemice.

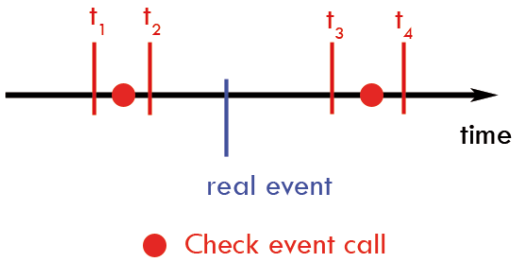

Figure 4: *Timeline for the occurence of a real event. Two times are measured, t_1 before a software event is checked and t_4 after the software event is detected. t_4 is the reported event time, the time span $t_4 - t_1$ is its uncertainty.*

5.2. Software timing

Measuring accurate software timings is nearly impossible without detailed knowledge about the used operating system, drivers, and hardware. As we like to provide basic timing functions on a portable layer, we abandon measuring exact times. Instead, we use a software scheme which tries to measure as good as possible and report an uncertainty or error value about all measurements made. The scheme used is based on the scheme described in [SLVV06]. Some commercial software systems like Presentation report in their documentation similar schemes. Therefore, monitoring measurement errors is an accredited method for many psychological studies.

Its idea is depicted in figure 4. As this is a software approach, one has to distinguish the time of the real occurance of an event and the time the software notices this event. We will refer to the latter as software event time. The software uses a function named *checkevent* to inspect if the software event has occured. A software event is generated if a real event from the hardware device has been propagated to the applications software layer (i.e., through all hardware components, the operating system, and drivers). Let t_1 be the time where the software event was reliably not noticed via *checkevent*. This is a timestamp measured just before a non-successful *checkevent* call. Further, t_4 is the time where the software event was reliably noticed. This is a timestamp just after a successfull *checkevent* call. Then, the software event has taken place between t_1 and t_4. t_2 is not reliable, as the software event could have occured between the non-successful *checkevent* and the measurement of t_2. t_3 corresponds to t_1 in the next loop iteration. It must be remarked that the real event may have occured before t_1, but the associated software event occured reliably after t_1. The system reports t_4 as event time and the time span $t_4 - t_1$ as uncertainty value. This corresponds to a worst case measurement

of the event, the user is free to use any point between t_1 and t_4 for the analysis.

While this measuring scheme is used for all events, for generating low uncertainties special care has to be taken. In the following, we restrict ourself to the appearance of visual stimuli and a button press reaction transmitted via the parallel port.

For measuring the stimulus appearance time, OpenGL functionality provided by OpenSG is used. Calls to *glFinish* and *glutSwapBuffers* are therefore removed from OpenSG. [Ste06a] evaluated rendering using only OpenGL on linux systems and showed its accuracy. As the appearance time on the screen is important, the vertical synchronization with the display device must be activated. While this option is typically turned off for higher performance, images are only displayed to the user with respect to the display device's update rate. In the rendering loop, after all data for a new image has been prepared and is ready, the system estimates the next time for the vertical synchronization. Using a double buffered approach, the frame buffer swapping is delayed just before the next resynchronization. If the rendering into the frame buffer is slower than the monitor's refresh rate, one or more synchronization points are missed. When the estimated time is reached, the buffer swap is called and ensured to be completed using *glFinish*. The more accurate the synchronization time estimation, the lower is the uncertainty.

As for the reaction device, we recommend using parallel port devices. While most modern PCs are not equipped with parallel ports any more, the parallel port is reported to have least hardware latency [Ste06b]. Checking input devices inside the rendering loop, i.e., frame-synchronous, is not accurate enough, especially if the rendering loop is synchronized with the display device as above. Observed input devices are therefore checked using a separate thread which polls the appropriate *checkevent* function. Polling is done here to ensure that the system does not suspend the thread without the application noticing it. Thread suspension of this polling thread is detected with the uncertainty measure, as any delay in the interesting loop iteration will affect the time span $t_4 - t_1$.

It should be mentioned that when using any kind of software measurement, the user must be informed about latency or jitter of applied hardware and software components. Even a single delay-inducing component may result in inaccurate measurements. While the software approach is less accurate and must be used with care depending on the required accuracy, it does not depend on external hardware and is therefore useable in a more flexible way.

6. Results

We evaluated the software measuring approach using different PC hardware, display systems and scenes. The hardware systems used are specified in table 1. System A was equipped with a standard CRT monitor. Two monitors were attached to system B's two graphics outputs, rendering left and right eye view like on HMD's or passive stereo systems. System C is used to drive an active-stereo Workbench.

Table 1: *Applied PC systems with different display devices.*

System	Description
A	NVIDIA GeForce 6600 GT CRT monitor, 1280 × 1024 at 75 Hz Intel Pentium 4 3.0 GHz 1 GB RAM
B	NVIDIA GeForce 6800 Ultra Two CRT monitors, 2560 × 1024 at 60 Hz Intel Xeon 3.2 GHz 2 GB RAM
C	NVIDIA Quadro FX 4400 Workbench active stereo, 1280 × 2048 at 75 Hz Dual Intel Xeon 3.2 GHz 2 GB RAM

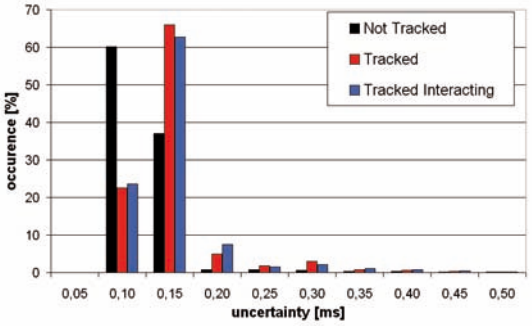

Figure 5: *Influence of tracking and interaction on the uncertainty values of the parallel port input device (measured on system C).*

For testing the parallel port latency, we applied a signal forwarding hardware. Using an ECP mode parallel port, we measured the round-trip time of a signal written and read from parallel port. The signal was written to a data port, forwarded by hardware to a control port, where it was read again. The time needed to write and read again was measured. All measured round-trip times were below 10^{-5} seconds, which corresponds to the expected results [Ste06b].

The next measurement analyzes the magnitude of occuring uncertainties. Therefore, we recorded stimulus presentation and button response uncertainties for different display modes and experiment scenarios. Figure 5 shows a histogram for the parallel port device's uncertainties. A number of 60,000 *checkevent* loops was measured, the figure depicts the number of uncertainty values (as percentage) falling into a certain range. Three different execution modes were measured on system C: disabled tracking, enabled head tracking with an idle user, and enabled head tracking with an interacting user. Uncertainties are clearly below 0.5 ms in all modes, with a majority less than 0.2 ms. Enabled tracking, regard-

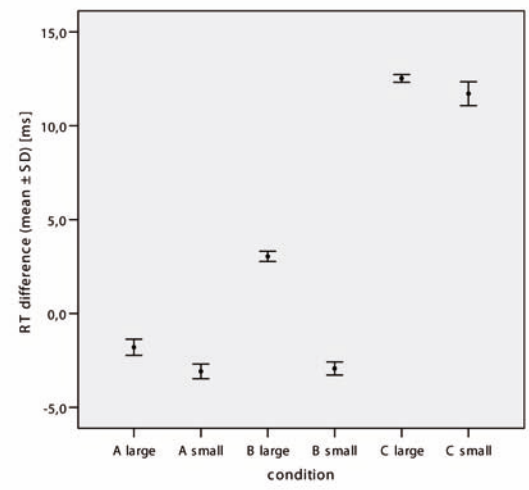

Figure 6: *Mean values and standard deviations (SD) for the hardware and software comparison. The reaction time (RT) difference is measured as software reaction time minus the measured hardware reaction time. While the absolute differences amounts to several milliseconds, the standard deviations stays below 1 ms.*

less of the user's movement, produces slightly higher uncertainty values.

Finally, we compared the software timing with the hardware timing (for visual stimuli) described in section 5.1. The software stimulus presentation timer and the optical sensor worked independently. The optical sensor was attached to the used display device, the software only observed the frame buffer swapping. The button press signal was split and then sent to the reaction time hardware as well as to the PC's parallel port. Using this setup, a comparison in accuracy for both systems is possible. Two different scenes were displayed: a small scene with a single geometry and a large city scene. The small scene was displayed at all systems with full frame rate (i.e., the monitor's refresh rate), while the large scene was displayed with half frame rate. The measured differences (software measured reaction time minus hardware measured reaction time) are depicted in figure 6, the difference's minimum and maximum values can be found in table 2. While there are measurement differences between hardware and software, these differences stay relatively constant during all measurements. The large absolute difference for system C (cf. table 1) is explained by a different position of the optical sensor for technical reasons. The standard deviation of all comparisons is below 1 ms, the maximal jitter of the software measured values is below 4 ms.

Table 2: *Comparison of hardware and software measurement. Depicted are minimum and maximum differences as well as the differences' standard deviation (SD).*

System	Scene	Minimum [ms]	Maximum [ms]	SD [ms]
A	small	-3.74	-2.19	0.39174
A	large	-2.61	-0.86	0.43192
B	small	-3.88	-2.31	0.34578
B	large	2.0	3.6	0.27247
C	small	9.11	12.43	0.63775
C	large	11.69	12.77	0.20680

7. Discussion

In simple reaction time tasks average reaction times for acoustic tasks lie between 120 and 130 ms and in visual tasks between 160 and 170 ms. It is common practice to classify reactions below 100 ms as anticipations.

The absolute value of reaction times is rarely the object of observation. Instead, differences between trials induced by different stimuli or virtual environments are analyzed. Therefore, the differences between hardware and software measurement are irrelevant, as long as these differences are of constant value. When absolute reaction time values are of interest, for example for a comparison with reaction times in the literature, the constant value must be measured as done in section 6. More important is the standard deviation, as well as the maximum deviation of the measuring differences. Assuming the hardware measurement represents the valid reaction time, deviations are influences on the software measurement by some unknown factor other than the factors induced by the experimenter (cp. [FF03]). Deviations above 10 ms would be critical for simple attention paradigms because differences between conditions could not be identified.

With growing scene and display complexity, the accuracy requirement for reaction times decreases, as typically more complex tasks have to be solved by the user. Therefore, it may be possible to permit small inaccuracies in the form of more flexible input devices into the system. Nonetheless, the experimenter must decide which accuracy is required for the study and therefore must be informed about possible inaccuracies.

Stable software reaction times with a standard deviation below 1 ms independent of the complexity of the scene and the applied display system qualify the system for a large class of psychological experiments in virtual environments.

8. Conclusions

We presented the ReactorMan software as a toolkit for VR-based psychological experiments. It aims at combining high ecological validity with precise stimulus control. Experiments can be designed by non-computer scientists and ex-

ecuted on a variety of Virtual Reality platforms. The software was successfully applied in several studies using Virtual Reality as a research tool as well as a research topic. The main part of this paper dealt with timing accuracy in virtual enviroments. Two solutions were presented and compared, a more accurate hardware solution, and a more flexible software solution for measuring reaction times. The results achieved with the software measurement and evaluated by the hardware approach revealed, that accurate stimulus presentation can be combined with a high ecological validity.

As future research, commonly used VR devices will be examined for reaction time experiments. The focus of future work will concern improved usability and easier experiment design. While scripting is a flexible solution, graphical tools may be more intuitive for the psychological researchers.

Acknowledgments

Parts of this work are supported by a grant from the Interdisciplinary Center for Clinical Research "BIOMAT." within the faculty of Medicine at the RWTH Aachen University. The authors are grateful to all researchers from the University Hospital and the Institute for Psychology who contributed to ReactorMan with their ideas.

References

[ALK05] ASSENMACHER I., LENTZ T., KUHLEN T.: Binaural Acoustics For CAVE-like Environments Without Headphones. In *IPT & EGVE Workshop* (2005), pp. 31–40.

[AWV*06] ARMBRÜSTER C., WOLTER M., VALVODA J., KUHLEN T., FIMM B., SPIJKERS W.: Virtual Reality in Experimental Ergonomic Research. In *Proceedings of 16th World Congress on Ergonomics (IEA)* (2006).

[FF03] FORSTER K., FORSTER J.: DMDX: A Windows Display Program with Millisecond Accuracy. *Behavior Research Methods, Instruments, & Computers 35*, 1 (2003), 116–124.

[HPM*04] HOFFMAN H., PATTERSON D., MAGULA J., CARROUGHER G., ZELTZER K., DAGADAKIS S., SHARAR S.: Water-friendly Virtual Reality Pain Control during Wound Care. *Journal of Clinical Psychology 60*, 2 (2004), 189–195.

[HVK*05] HEBER I., VALVODA J., KUHLEN T., STURM W., FIMM B.: Asymmetries of Visuo-Spatial Attention in Virtual Space after Sleep Deprivation. *Journal of Cognitive Neuroscience, Supplement: 51* (2005).

[LBB99] LOOMIS J., BLASKOVICH J., BEALL A.: Immersive Virtual Environment Technology as a Basic Research Tool in Psychology. *Behavior Research Methods, Instruments, & Computers 31*, 4 (1999), 557–564.

[LJ02] LEWIS M., JACOBSON J.: Game Engines in Scientific Research - Introduction. *Communications of the ACM 45*, 1 (2002), 27–31.

[MOB01] MEISSNER M., ORMAN J., BRAUN S. J.: Case Study on Real-Time Visualization of Virtual Tuebingen on Commodity PC Hardware. In *VIS '01: Proceedings of the Conference on Visualization '01* (2001), pp. 433–436.

[PHT04] PLANT R., HAMMOND N., TURNER G.: Self-validating Presentation and Response Timing in Cognitive Paradigms: How and Why? *Behavior Research Methods 36* (2004), 291–303.

[RBB*06] RIZZO A., BOWERLY T., BUCKWALTER J., KLIMCHUK D., MITURA R., PARSONS T.: A Virtual Reality Scenario for All Seasons: The Virtual Classroom. *CNS Spectrums 11*, 1 (2006), 35–44.

[Riv05] RIVA G.: Virtual Reality in Psychotherapy: Review. *Cyberpsychology & Behavior 8*, 3 (2005), 220–230.

[RK05] RIZZO A., KIM G. J.: A SWOT Analysis of the Field of Virtual Reality Rehabilitation and Therapy. *Presence - Teleoperators and Virtual Environment 14*, 2 (2005), 119–146.

[SLVV06] STEVENS M., LAMMERTYN J., VERBRUGGEN F., VANDIERENDONCK A.: Tscope: A C Library for Programming Cognitive Experiments on the MS Windows Platform. *Behavior Research Methods 38*, 2 (2006), 280–286.

[Ste06a] STEWART N.: Millisecond Accuracy Video Display using OpenGL under Linux. *Behavior Research Methods 38*, 1 (2006), 142–145.

[Ste06b] STEWART N.: A PC Parallel Port Button Box provides Millisecond Response Time Accuracy under Linux. *Behavior Research Methods 38*, 1 (2006), 170–173.

[VAD*03] VALVODA J. T., ASSENMACHER I., DOHLE C., KUHLEN T., BISCHOF C.: NeuroVRAC - A Comprehensive Approach to Virtual Reality-based Neurological Assessment and Treatment Systems. In *Proceedings of Medicine Meets Virtual Reality 11* (2003), pp. 370–372.

[VAKB04] VALVODA J. T., ASSENMACHER I., KUHLEN T., BISCHOF C.: Reaction-Time Measurement and Real-Time Data Acquisition for Neuroscientific Experiments in Virtual Environments. In *Proceedings of Medicine Meets Virtual Reality 12* (2004), pp. 391–393.

[VKB06] VALVODA J. T., KUHLEN T., BISCHOF C.: Interactive Virtual Humanoids for Virtual Environments. In *Short Paper Proceedings of the Eurographics Symposium on Virtual Environments* (2006), pp. 9–12.

[ZF02] ZIMMERMANN P., FIMM B.: A Test Battery for Attentional Performance. *Applied Neuropsychology of Attention. Theory, Diagnosis and Rehabilitation* (2002), 110–151.