Studierstube: An Application Environment for Multi-User Games in Virtual Reality

Anton Fuhrmann[1)], Werner Purgathofer[2)]

[1)] VRVis center for virtual reality and visualization, fuhrmann@vrvis.at
[2)] Vienna University of Technology, purgathofer@cg.tuwien.ac.at

Abstract: *Studierstube is an operating system for applications in virtual reality. It supports multiple users collaborating in a distributed system. While initially developed for scientific visualization, the generic approach supports almost any kind of VR application on a wide range of VR-hardware. Inherent concepts of Studierstube – like real-time graphics, high-level interaction methods, multi-user integration, and distributed execution – make it an ideal environment for multi-user games in virtual reality. In this paper we show how Studierstube implements these concepts and how game programmers can utilize them.*

1. Introduction.

Virtual Reality has become something of a buzzword. While the advent of head-mounted displays and cyber gloves about 20 years ago created the most impressive images for hyping this new technology in magazines and on TV, today – in computer terms almost an eternity later – almost the same images keep popping up when VR is discussed in the media. Additional boost for this popular conception of VR has come from an unexpected direction: the world-wide web and its commercial companion e-business have adopted the adjective "virtual" for almost all of their features. However, the term "virtual reality" is more justified when applied to 3D computer games. Many of these games aim to immerse the player in a simulation of reality, even if this simulation is only limited to the visual output of a computer screen. Which brings us to our definition of virtual reality:

"Virtual Reality (VR) is an immersive simulation of some properties of reality. In most cases these properties include the simulation of visual aspects and interactive responses."

While this definition does not apply to "virtual shopping on the internet" – which is about as realistic a simulation as catalog shopping – it certainly applies to most modern 3D games. These games deliver real-time graphics at about 50 frames per second, implement interaction with the virtual environment, and simulate physical properties like gravity, inertia and kinematics.

Game engines, which drive those 3D games, are commercially available. Some of these are even affordable for educational purposes [1]. While these games represent the low end of the spectrum of VR applications, they are by no means trivially implemented: the popularity of 3D games has probably driven more research in computer graphics hardware during the last years than any other application, and has allowed prices of 3D hardware accelerators to drop from workstation levels (>50k€) to mere commodity prices (approximately 200€).

Nevertheless the images of persons wearing bulky goggles and manipulating imaginary objects with wired gloves are deeply ingrained in the popular conception of VR. Some companies have already tried to systematically cash in on these expectations: One – now defunct - company tried to enter the arcade game market producing VR equipment for location-based entertainments (LBEs) and supplying their own games (most famous was the "Dactyl Nightmare"). Disney Corporation tested the potential of VR in the entertainment industry in their own amusement parks. Their installation took the players on a virtual ride on a flying carpet [5]. Both companies used purpose-built equipment and specifically designed software.

These two scenarios – commercially available game engines executing on commodity PCs and application-specific software executing on custom built hardware – define a range of sophistication available to VR game designers. On both ends of this range multi-user games

are available. Local and global interaction of multiple players via LAN and internet are supported for many games and seem to be a central feature of interest for many players. Competitions, teamwork and interaction with "real" opponents add spice to the experience of immersing oneself in a fictive world.

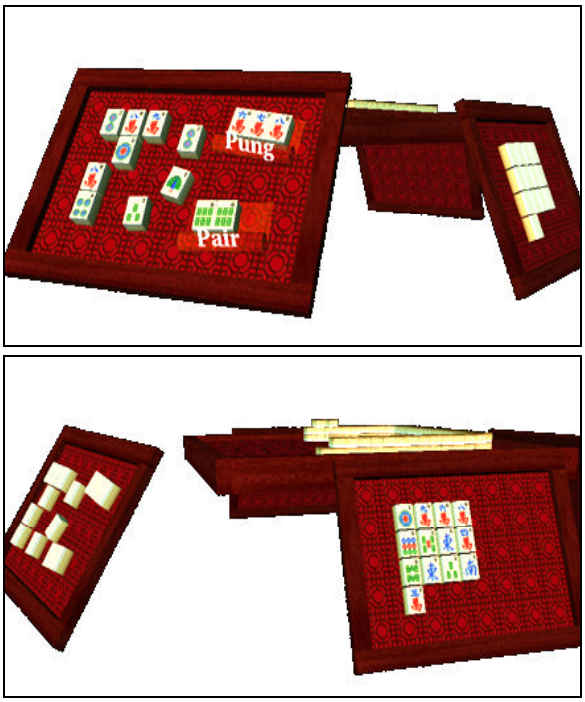

Figure 1: Personal displays secure privacy when playing Mahjongg – the left player (top view) cannot see his opponent's tile labels and vice versa (bottom view)

2. Studierstube – a collaborative virtual environment

Over the past 5 years, we have developed *Studierstube*, a virtual environment "operating system" providing a comprehensive application programmer interface (API) for developing virtual reality applications. The initial proposal [7] centered mainly on the multi-user or groupware aspect of the system. At first, Studierstube was mainly applied in scientific settings for visualization purposes [2]. Further development of the API allowed us to introduce students to VR application programming, which resulted in some small game-like applications [3] and even a complete implementation of the traditional board game "Mah-Jongg" (Figure 1) [9].

The initially proposed hardware set-up for the Studierstube environment [7] consists of multiple users wearing head-mounted displays (HMDs) and interacting with a pen-and-pad combination called the *Personal Interaction Panel* (PIP) (Figure 2). The PIP is a black plastic panel on which computer-generated images can be projected. Its position and orientation is tracked by the system and it can be used as output device or – in combination with the tracked stylus – as input device by displaying buttons and sliders.

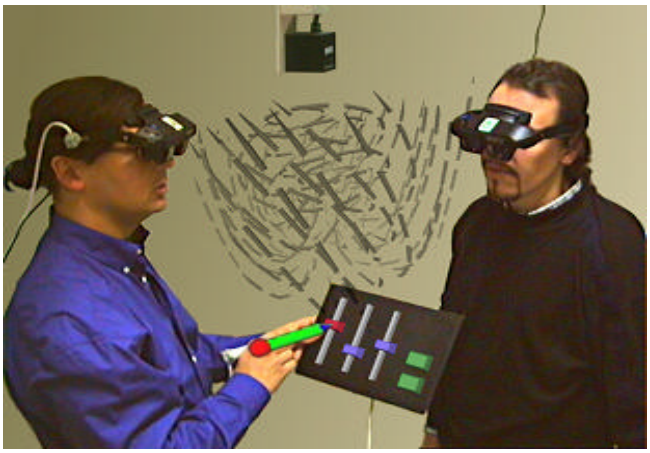

Figure 2: Two users wearing see-through displays and adjusting visualization parameters on the Personal Interaction Panel (PIP)

Studierstube possesses a lot of properties we see as essential for implementing a multi-user collaborative virtual environment. Most of these properties are prerequisites or at least of added value when implementing multi-user VR games:

real-time graphics Since a virtual environment has to update the graphics for each user continuously, real-time graphics are an essential feature of the whole system. As opposed to a normal graphical user interface, the ability to deliver graphics in real-time is one of the basic properties of Studierstube.

collaboration Multiple user can enter the same environment and interact with each other and the objects therein. In the context of a game, collaboration means of course also competition. Studierstube is mainly based on *local collaboration*, where all players are physically in the same location while using the virtual environment.

augmentation **T**he user sees not only the computer generated images, but also the real environment. This mixture of real and virtual reality – called *augmented reality* or *mixed reality* – not only allows us to integrate real objects in the virtual environment, but also reduces the users stress or claustrophobia. Furthermore augmentations allow direct social interaction between local players, reducing the necessity for additional communication features, e.g. communication channels between members of a team of players.

augmented props Computer generated images can be displayed over real objects. When the position and orientation of these objects is tracked by the system – e.g. using magnetic trackers – the object becomes a prop in the game. Such a prop may be integrated in the gameplay, as light saber, magic mirror or any other object the designer can think of. One of our primary interaction devices - the PIP - is such an augmented prop.

privacy The issue of defining what other may see or not of course only applies to multi-user environments. While used mostly as a structuring tool for reducing display clutter when using Studierstube as a groupware tool, in a gaming environment the ability to hide certain information from members of the opposing team can be essential (Figure 1). Imagine playing poker without privacy for your own hand!

high-level interaction Interaction methods are implemented using an event distribution mechanism. High-level interaction methods like widgets (buttons, sliders, trackball) and 3D windows (Figure 3) allow for easy design and implementation of structured input and output.

distributed execution Studierstube contains the *Distributed Open Inventor* extension [4], a library of functions developed to specifically allow for the distribution of our virtual environment over a network. Distribution takes place mainly transparent for the application programmer. This means a sufficiently clean coded application executes as well in a single or multi computer set-up.

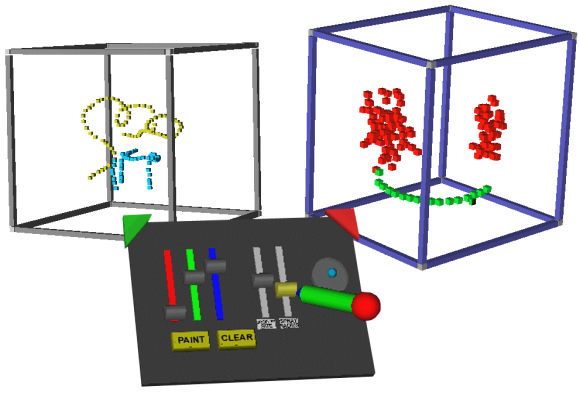

Figure 3: A simple 3D-painting application. Two 3D windows contain the paintable volumes and sliders on the PIP (foreground) can be used two select color and thickness of the lines.

All of the properties mentioned above resulted not specifically from game-oriented design of the API, but from necessities, which arose from elemental groupware applications. In combination they provide a rich and versatile base for game programming.

3. Application programmer's interface

Studierstube's software development environment is realized as a collection of C++ classes built on top of the Open Inventor (OIV) toolkit [8]. The rich graphical environment of OIV allows rapid prototyping of new interaction styles. The file format of OIV enables convenient scripting, overcoming many of the shortcomings of compiled languages without compromising performance. At the core of OIV is an object-oriented scene graph storing both geometric information and active interaction objects. Our implementation approach has been to extend OIV as needed, while staying within OIV's strong design philosophy.

The *Studierstube* API imposes a certain programming model on applications, which is embedded in a foundation class, from which all *Studierstube* applications are derived. By overloading certain polymorphic methods of the foundation class, a programmer can customize the behavior of the application. Studierstube can execute multiple such applications at a time. Applications can be dynamically loaded at runtime, or the system can load a single application at startup, preferably the method one would use for a multi-user game.

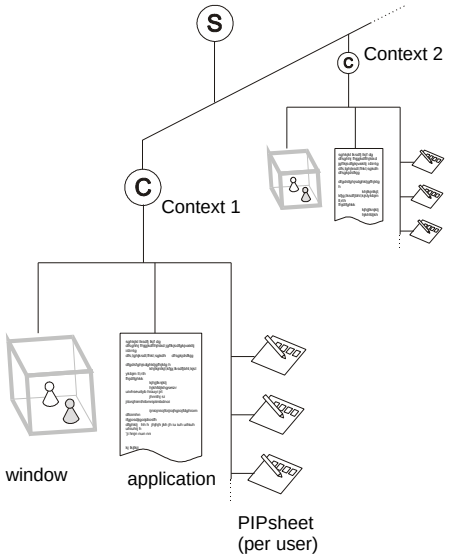

Figure 4: An application is implemented as a node in the scene graph, as are windows and PIP sheets. This allows for the organization of all relevant data in the system in a single hierarchical data structure.

To create a useful application, a programmer need only create a subclass of the foundation class and overload the 3D-window and PIP creation methods to return custom scene graphs (Figure 4). Typically, most of the remaining application code will consist of callback methods responding to certain 3D events such as a button press or a 3D direct manipulation event. This mirrors input concepts common to most graphical user-interfaces.

4. Hardware Set-Up

Studierstube has been initially developed on Silicon Graphics hardware and the IRIX operating system, but general availability of Open Inventor and the rapid development of inexpensive 3D graphics hardware mentioned above allowed us to port it to PC hardware and the Windows and linux operating systems. But the real challenge has been the integration of a wide range of the more or less exotic input and display devices necessary for virtual reality.

4.1 Tracking

Tracking is one of the main issues when implementing virtual reality. It means measuring the users position in space in *real-time*. In most cases the users head position and orientation in space is used to generate the correct view of the environment and the users hands are tracked to enable direct three-dimensional interaction.

Main issues for the application of tracking hardware in game applications are:

Delay and update rate of the tracking system. Systems having large delay lengthen response times and low update rates (<30Hz) make the system miss fast gestures.

Measurement range of the tracking system. Magnetical and most optical tracking systems only support operating areas of a few square meters, making the implementation of games involving larger playing areas (virtual soccer) impossible.

Wireless operation is desirable, otherwise the length of the wires limits the operating area and the wires can tangle.

Ruggedized hardware is a requirement for most games running in arcades or other locations open to the public.

For many gaming applications optical tracking like our own system [10] seems to be a viable solution: it provides fast, reliable tracking in a sufficient large operating area for many gaming applications. Its main disadvantage, the occlusion of tracked markers, is only of secondary significance when a single user set-up is desired. Further advantages are its ruggedized application - the user only carries lightweight retro-reflective markers, which are inexpensive and hard-wearing – and the lack of wires.

We support a range of magnetic tracking devices (ASCENSION, Polhemus), inertial tracking (InterSense) and our own optical tracking system as positional input devices for HMDs, stylus and PIP. The necessary hardware abstraction layer and configuration mechanism is implemented by our OpenTracker toolkit [6].

	Tracked display	**Tracked head**	**Desktop**
Field sequential	Sony Glasstron	Virtual Table	Fishtank VR with shutter glasses
Line interleaved	i-glasses	VREX VR2210 projector	i-glasses w/o head tracking
Dual screen	i-glasses Protec	Single user dual-projector passive stereo w/head track.	Multi-user dual-projector passive stereo
Mono	i-glasses (mono)	Virtual Table (mono)	Desktop viewer

Table 1: All combinations of camera control and display modes have distinct uses.

4.2 Displays

Studierstube is intended as an application framework that allows the use of a variety of displays, including projection based devices and HMDs. There are several ways of determining camera position, creating stereo images, setting a video mode etc. After some consideration, we implemented an OIV compatible viewer with a plug-in architecture for camera control and display mode.

This approach, together with a general off-axis camera implementation, allows runtime configuration of almost any available display hardware. Table 1 shows an overview of some devices that have evaluated so far.

Display mode in the leftmost column describes if and how stereoscopic information is transferred to the display, whereas camera control in the top row describes how the viewpoint can be changed by the user: “Tracked display” means tracking of a head-mounted display, “Tracked head” means a stationary display is used and users head position is tracked (e.g. a CAVE or back projection wall), and “Desktop” means that the user selects the viewpoint via a mouse and keyboard interface.

5. Game design in Studierstube

Studierstube supports game design on different levels of the design process.

Modeling can be performed in any 3D design tool exporting VRML files [11]. VRML (Virtual Reality Markup Language) is a standardized file format implementing polygon based modeling and simple interaction and animation.

Animation scripting – as mentioned above – can be implemented directly in the VRML file for simple animations (e.g. periodic motions, path or keyframe animation, controlled transformations).

Widgets can be used to implement simple, three-dimensional interactions like the opening of doors, placement of objects and so on.

Complex interactions can be implemented in C++, utilizing the full power and complexity of the OpenInventor framework together with the Studierstube 3D event distribution mechanism. Typically game-specific interaction is realized on this level.

Below we describe three simple game implementations in Studierstube.

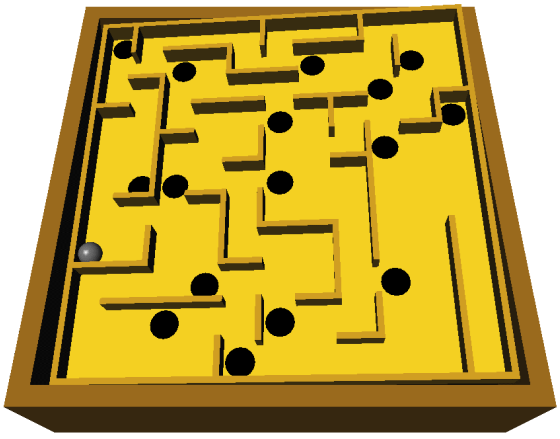

Figure 5: Maze - a simple game of skill and coordination.

5.1 "Maze"

OpenInventor comes with a simple demo application implementing a classical game of skill. It simulates a wooden tablet with a simple maze containing holes on strategic positions (Figure 5). By tilting the panel, one is able to navigate the ball from a given start point to the goal.

One of our first game implementations in Studierstube was an adaptation of this game. The user takes the PIP, which is virtually overlaid with the maze, and tilts it accordingly.

The only implementation work was to remove the 2D mouse event to 3D orientation mapping of the demo and replace it with the orientation of the real PIP, supplied by the attached tracker. Some house-keeping functions for scoring and ending the game finished the conversion from window-based 2D-application to virtual game executing in Studierstube.

5.2 "Blockout"

The goal of the old arcade game "blockout" was the complete removal of all bricks of a wall by bouncing a ball of them. Originally played in 2D, the player had to manipulate a paddle in one dimension to direct the ball to the remaining stones.

We adapted this game for VR by allowing the player to move his paddle (Figure 6, right) directly via a handle tracked with sic degrees of freedom. Thereby not only position, but also orientation of the paddle could be used to aim the ball.

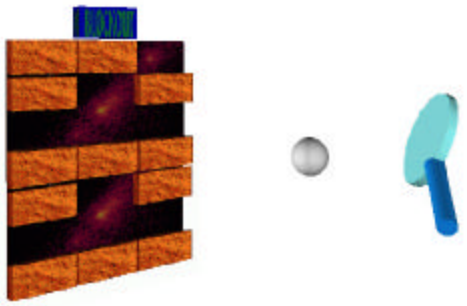

Figure 6: The game "Blockout" simulates a popular arcade game testing reflexes. The user tries to remove all the blocks on the wall by bouncing the ball of them with the paddle.

Implementation was quite straightforward. Wall and bricks (Figure 6, left) as well as ball (Figure 6, middle) were modeled in VRML and copied into the Studierstube scenegraph, where their position and – in case of the bricks – visibility could be manipulated by the application via simple OpenInventor fields. The geometry of the paddle was bound to the transformation supplied by the tracker on the real rod. For every time-step of the applications main loop, collisions between ball, paddle, and bricks were evaluated, and the state of the game – ball position and direction, number of bricks, score – manipulated accordingly. Since OpenInventor supplies all essential geometric calculations like

intersections between lines, ray object intersections and so on, the coding was kept at a minimum.

5.3 "Virtual Casino"

Our most sophisticated game environment so far has been the "Virtual Casino", a multi-user environment for virtual gambling. In this game two or more users are able to play against the bank in a fully functional casino simulation (Figure 7).

The players use their PIP as purse, distributing their jetons on the table at will. Bets are commented and displayed by flyover texts (Figure 7, inset). When all players have placed their bet, the wheel is started and the bets are collected. We used a combination of widgets, procedural animation and specially coded interaction to implement this game.

Studierstube supplies via its API a general interface to multiple users, allowing the application programmer to response to input depending on which user is interacting.

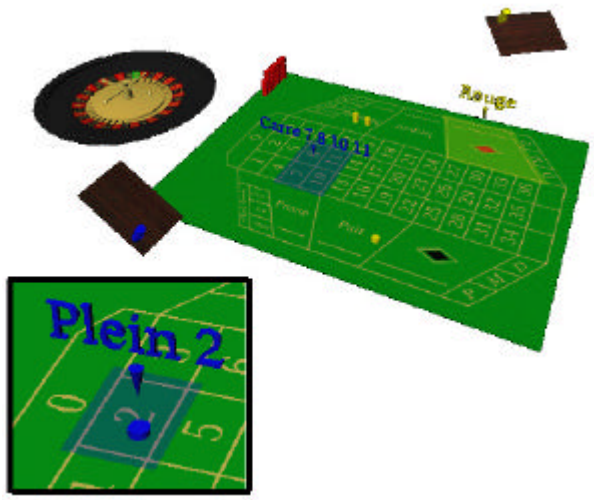

Figure 7: A "Virtual Casino", utilizing widgets with flyover explanations (inset).

6. Conclusion

Studierstube provides a powerful application programmer interface, network distribution mechanism and the necessary device interfaces for the implementation of virtual reality games. Its main strengths are high-level interaction methods, an integrated multi-user concept and the adaptability to many different hardware set-ups.

Most of the games implemented hitherto are relatively simple, but we expect this to change with the impending open source distribution of Studierstube.

Future work will include the integration of a scripting language (JAVA or Python) to allow for faster prototyping of game behavior.

7. References

[1] Conitec Datensysteme GmbH, web page, http://www.conitec.net/a4info.htm

[2] A. Fuhrmann, H. Löffelmann, D. Schmalstieg, M. Gervautz. Collaborative Visualization in Augmented Reality. IEEE Computer Graphics & Applications, Vol. 18, No. 4, pp. 54-59, IEEE Computer Society, 1998.

[3] "Collaborative Games in Augmented Reality", TU-Wien, http://www.cg.tuwien.ac.at/research/vr/gaming/

[4] Hesina G., D. Schmalstieg, A. Fuhrmann, W. Purgathofer. Distributed Open Inventor: A Practical Approach to Distributed 3D Graphics, Proc. VRST '99, London, pp. 74-81, Dec. 1999.

[5] R. Pausch, J. Snoddy, E. Hazeltine, Robert Taylor and Scott Watson: Disney's Aladdin: First Steps Toward Storytelling in Virtual Reality. SIGGRAPH 96 Conference Proceedings, pp.193-204, Addison Wesley.

[6] G. Reitmayr, D. Schmalstieg. OpenTracker - An Open Software Architecture for Reconfigurable Tracking based on XML. Appeared as a poster in: IEEE Virtual Reality 2001, Yokohama, Japan, March 13-17, 2001.

[7] D. Schmalstieg, A. Fuhrmann, Z. Szalavari, M. Gervautz: "Studierstube" - An Environment for Collaboration in Augmented Reality. Extended abstract in proceedings of Collaborative Virtual Environments '96, Nottingham, UK, Sep. 19-20, 1996. Full paper in: Virtual Reality - Systems, Development and Applications, Vol. 3, No. 1, pp. 37-49, 1998.

[8] P. Strauss, R. Carey. An object oriented 3D graphics toolkit, Proc. SIGGRAPH '92, pp. 341-347, 1992.

[9] Z. Szalavári, E. Eckstein, and M. Gervautz: "Collaborative Gaming in Augmented Reality" Proceedings of VRST'98, pp.195-204, Taipei, Taiwan, November 2-5, 1998,

[10] Miguel Ribo, Axel Pinz, Anton L. Fuhrmann: "A new Optical Tracking System for Virtual and Augmented Reality Applications", technical report: http://www.vrvis.at/TR/2001/TR_VRVis_2001_004_Full.pdf

[11] Rikk Carey, Gavin Bell: The Annotated VRML 2.0 Reference Manual. Addison-Wesley, 1997.