The Responsive Workbench Simulator: a Tool for Application Development and Analysis

Michal Koutek and Frits H. Post

Faculty of Information Technology and Systems
Delft University of Technology
Mekelweg 4, 26 28 CD Delft, The Netherlands
{M.Koutek,F.H.Post}@cs.tudelft.nl

ABSTRACT

In this paper we present a software environment for visualization and interaction on the Responsive Workbench (RWB). Our main focus will be on the RWB Simulator. We will also briefly describe the architecture and the usage of the RWB Library.
The RWB Simulator is an excellent tool for development, evaluation and analysis of the RWB applications. It is a powerful introduction, learning and presentation tool for the RWB. We will also present a novel method for development of immersive VR applications. Finally we will demonstrate the RWB Simulator on some visualization applications.

Keywords: Virtual Reality, Responsive Workbench (RWB) Library, RWB Simulator

1 Introduction

The Virtual Reality Responsive Workbench (RWB) is a powerful system for 3D visualization and interaction [1]. It intensifies perception of 3D models and data. The RWB is a semi-immersive virtual environment: the user stands in the real world and is looking into a virtual world which is projected on the screen of the Workbench. One of the advantages of the RWB is its tabletop metaphor. It creates for the user an illusion of a laboratory table or a design studio, while the only real element is the wooden construction of the workbench, everything else is purely virtual. The RWB also offers a large screen to visualize the 3D models. Combined 2D and 3D interfaces can be used for the user interaction.

The RWB is complementary to the CAVE [3], where the user is almost fully immersed into the projected virtual environment. The usage of the RWB is a bit different than of the CAVE. The RWB benefits from the table metaphor although its field of view is rather limited. In the CAVE, all objects are usually virtual. In automotive industry they put mock-ups, car-seats inside the CAVE to have at least something real with a substance to be able naturally interact with the virtual environment, but they have to face the problems of interference with electro-magnetic tracking and thus wooden or plastic materials have to be used and sometimes even different tracking system must be applied. In the CAVE usually larger objects are visualized, and the user interacts with them often at a larger distance than on the RWB. For our application the RWB is more suitable than the CAVE. We mainly make use of the laboratory table metaphor.

1.1 Related Work

For controlling these types of immersive systems VR software and libraries are needed. Three years ago, when the RWB facility was installed at the HPaC Centre at TU Delft, there were not many software options. On one hand there were a few experimental libraries (like Avocado/Avango [2], VR-Lib [18], MR Toolkit [17], VrTool [15],SVE-lib [16], Studierstube [19]) used by VR researchers and on the other hand there was a commercial software, like the CAVE Library [13] or the WorldToolKit (WTK). Today we can see variety of VR systems and libraries more or less specialized on specific types of VR applications, like data visualization, industrial or medical applications, etc. VR systems, like VR-Juggler [9], MAVERICK [11] and others, follow the global trend and are available as open-source software. This gives us an opportunity to get better understanding about the concepts of VR systems and sometimes also to improve their implementation.

1.2 Our Perspective

We have found that the process of developing VR application needs a special attention.

With some systems the development has to be done directly on the VR facility. Some systems have today an option of making the VR applications off-line in a sort of "VR on desktop - simulator", like the CAVE Simulator. Systems like Avango, Lightning [10] or VR-Juggler and others, where the application developer can define the virtual application platform to be, for example, a desktop system driven by keyboard and mouse which emulate the 3D input from the immersive VR systems.

However, trying to simulate 6 DOF (degrees of freedom) input with 2 DOF mouse is not a very practical solution. It is impossible to simulate in this way a complex 3D interaction that the user does (would do) in the virtual environment (VE) with the implemented application. We think that the user interaction is one of the most important aspects of VR applications. Therefore we should devote enough attention to it during the development of an application. The best way of testing the real 3D interaction is in the immersive VE itself. However, almost everybody has found that it is much more effective to develop the VR application off-line on a desktop-system. The developer then pays the price of missing 3D interaction with the application, which can be compensated by using (semi-) 3D input devices like the Spacemouse, etc., but still that is not the same level of interaction as on the Responsive Workbench or in other immersive VEs.

Our proposed solution for this problem is based on capturing the user's (developer's) 3D interaction with the prototype application on the Responsive Workbench. We measure the tracker data (user's head and 3D interaction devices) with a given frame rate and we also store timestamps of the RWB application running on the Responsive Workbench. This measured data are then used by the application developer within the RWB Simulator for the next stage of development and improvement of the RWB application. The RWB Simulator works on a desktop system and is controlled by mouse and keyboard commands. The tracker data steer the interactions of the virtual user inside the application. A powerful tool is available to the developer for tracing and simulating of application forward, in some cases also backward in time with captured 3D interaction.

It is not only a problem of simulating the real 3D interaction during VR application development that we solve with our concept of the RWB Simulator. The use of this tool has a strong impact on the processes of application development, analysis, evaluation and presentation. It reduces the time needed on the Responsive Workbench and increases the efficiency of the whole process. Detailed explanation on these concepts follows in this paper.

We give a necessary overview of the framework and the architecture of our RWB system. We want to put an emphasis on the process of application development using the RWB Simulator. We will also discuss the old problem of application developer and application user in a VR context. Animations and images of RWB Simulator will document the application spectrum.

2 RWB Basics

The Responsive Workbench is based on a stereo back-projection table system, which is combined with an electro-magnetic tracking system, see Figure 1. The stereo images are generated from the SGI ONYX 2 with 4 CPUs and the Infinite Reality 2 graphic card. We use the display resolution of 1120 $\times$ 840 pixels in 96 Hz in stereo mode.

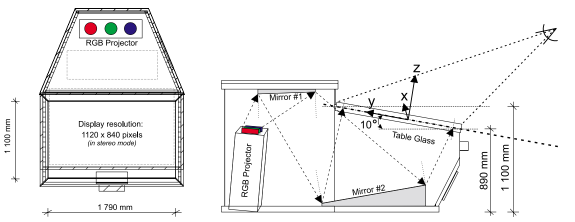

Figure 1: Top and side views on the Workbench

The image from the RGB projector is reflected though two mirrors and has to fit properly on the glass table which is tilted by 10°. To obtain a clear and a sharp image the RGB projector has to be well calibrated and precisely aligned. Crystal Eyes shutter glasses are used to create the stereo effect.

The virtual world projected on the screen of the RWB is represented in workbench table coordinates. The tracked positions and orientations of the user's head and hand must be converted to the same table coordinate system. We are using table-centered and table-aligned coordinate system, see Figure 2.

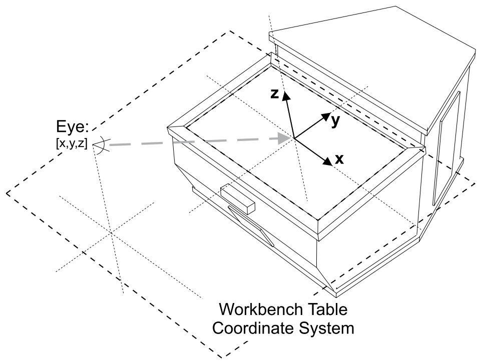

Figure 2: Workbench table coordinate system

In the RWB environment, the head-tracker updates user's viewpoint, and the tracking of the stylus pen forms the base for a 3D interaction.

3 The RWB Library

Considering the available hardware and our needs we have built our own RWB Library. The available hardware resources should be used effectively by the RWB application.

Besides the Onyx 2, we have also other SGI workstations in our laboratories: the Onyx, the Octane and several O2s. We want them to be used also in combination with our VR facility.

The typical RWB application needs real time 3D graphics and 3D interaction. Therefore we have chosen for Iris Performer which offers an optimized 3D graphics pipeline based on OpenGL, and which also provides a good support for multiprocessing and shared memory access in the IRIX 6.5 operating system.

The tracker daemon is a separate application which reads the data from the tracking systems and converts them to the workbench coordinate system. On the side of the RWB Library there are functions to read the tracker data from the tracker daemon's shared memory. An advantage of such a solution is that multiple applications/processes can access the shared memory without using the HW ports of the tracking system. For compensation of static tracking errors we use the grid-calibration method [4].

We have set up the stereo-projection pipeline using the off-axis perspective projection. We have tried several stereo projection schemes including the on-axis perspective which we had proved not to be suitable for the RWB, especially because of the distorted perspective which was a serious problem for 3D interaction with the stylus pen. Head-tracking together with the stereo-projection creates for the user an illusion of 3D objects resting on the workbench tabletop.

Iris Performer together with our RWB Library offers many functions to access 3D geometry files or for creating a custom geometry and building the scene graph of the virtual world which is displayed on the RWB.

We have incorporated a generic class, the rwb-obj, which contains the geometrical information as well as interaction functions in the form of interaction, drawing and culling call-backs. This type of class also incorporates collision detection and intersection calculations.

3.1 The Structure of RWB Applications

We defined a multiprocessing scheme for a general RWB application, which consists of the main RWB process, the tracker daemon, the keyboard/mouse interaction process and the user application process. The main RWB process consists of application, culling and draw subprocesses as defined in Iris Performer. All processes can communicate through shared memory. For this purpose we have created unified shared memory object called "Shared". All the necessary parameters of an RWB application can be accessed from there.

We have designed the functions of the RWB Library to be clear, easy to use, and to minimize the programming for the developer of an RWB application. The user has to specify just the rwb-objects within the virtual world and to define the special functional/interaction call-backs of the application and the rwb-objects. The documentation of the RWB Library functions and example applications can be found at [4], [12].

3.2 3D Interaction and User Interface

The RWB Library offers the user interaction with devices like a keyboard, a mouse, a space-mouse (with 6 degrees of freedom), and a stylus pen (6 DOF). The space-mouse is used for navigation in large environments and to its 9 keys extra functionality can be assigned by the application. For a real 3D interaction the RWB applications use the stylus pen with a tracking sensor and one button. A widget set is available for building a 3D user interface with buttons, sliders, menus, or display, dial and type-in windows. At the position of the stylus pen the actual tool is displayed.

3.2.1 Interaction with virtual objects

When the stylus pen is located inside an rwb-object (colliding with it), the object is selected and changes its color to red.

The user can then invoke object's function by clicking on the button, or holding the button and simultaneously performing some motion. There are four basic *call-back functions* of a generic rwb-object: *touch/untouch, pick, manipulation, release.*

Each of these can be user specified. For example touch call-back can print object information, or for example if the user picks a door it starts an animation of opening the door.

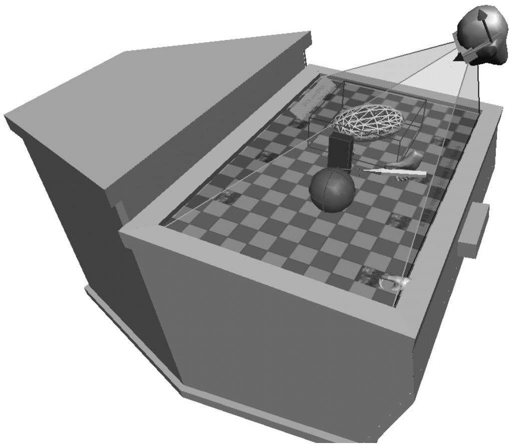

Figure 3: Object manipulation; FOV volume

Objects can be selected and manipulated directly with the stylus, or in the case of distant objects, using a ray-casting technique (see Figure 3).

Into the RWB Library we have also built a visual force-feedback method to provide a visual interface and to substitute a real force input. We use spring-based tools attached to objects assisting the manipulation. *The spring manipulation tools: spring, spring-fork, spring-probe* were introduced in [5]. During the design of these tools we have built a physics-based world and we were using the RWB simulator to simulate and trace manipulation tasks.

3.2.2 Object Collisions

For a realistic object behavior in user interaction, collision detection is important to prevent objects from moving through each other. Detecting object collisions helps to create an illusion that virtual objects have a substance.

We have implemented the following object collision schemes: collision between stylus-object, ray-object and object-object. The stylus and ray intersection is supported by Iris Performer. Collisions between objects were implemented into our library [4].

3.3 Workbench In Workbench

Besides the functions and features of the RWB Library mentioned above, the collision and the intersection functions, the 3D user interface and the spring-based manipulation tools, there are some special functions and features of the RWB Library.

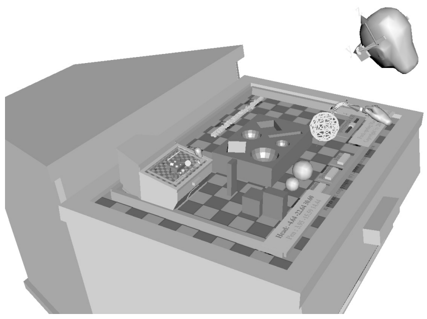

Figure 4: WIW: mini preview

For an improvement of the user's orientation in the virtual world projected on the RWB we have built in the library the Workbench In Workbench function, see Figures 4 and 5.

A small copy of the Workbench is projected onto the RWB table top. It contains the whole virtual world with all its objects as well as the user's head and the stylus. The user can navigate in a large world by looking onto the Workbench miniature and seeing which part of the world is displayed on the RWB. The WIW function helps to locate and manipulate objects which are not projected onto the RWB table top because they are outside the field of view (FOV), which is shown in Figures 3, 7. These objects are visible in the workbench miniature. In the RWB Simulator it can be seen which virtual objects collide with the FOV, see Figure 7. We should mention that when objects are cut-off by the FOV volume it destroys the 3D immersion. Thus, we have to try to avoid this by placing the virtual objects inside the FOV.

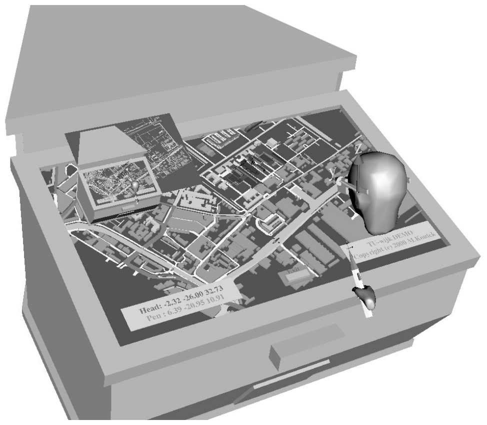

Figure 5: WIW: navigation assistance

The WIW metaphor is also used for collaboration with another user in distributed applications. On the miniature of the workbench the user can see what the other user does in the virtual environment.

The WIW function can also assist in learning how to use RWB applications, serving as a virtual application guide.

3.4 Monitoring of User Interaction

A very important aspect of the work with the RWB is to be able to monitor and debug an RWB application in a distributed VR environment. Therefore we have implemented a monitoring function of the user interaction. The principle is quite simple. During the runtime of the application the tracker data and application timestamps are written into a file (or sent through a network) for immediate or later use, such as animated replay of a workbench session in the RWB Simulator.

4 The RWB Simulator

We have already mentioned some interesting aspects of the RWB Simulator. Basically it is a desktop development environment for RWB Library applications. This tool provides application support in areas of: *development, evaluation, learning, presentation, navigation assistence and collaboration in distributed applications.*

4.1 The RWB Application Development

Development and implementation of VR application should be efficient. We try to minimize the time spent in immersive VE during development, because it is not always convenient and effective to debug or monitor an RWB application on the RWB itself. Sometimes the user performs application-specific tasks, and it is difficult to see if the task or the underlying algorithm works properly, when the user is just standing by the Workbench and wearing the shutter glasses. Usually, program variables will be written onto the screen and analyzed. On the real Workbench you cannot efficiently pause, slowdown and debug the application. This is even more complex if we consider the multiprocessing nature of the RWB application.

The RWB Simulator uses the captured tracker and application data and highly interactive applications benefit from it.

Time dependent simulations can make use of the unified RWB-time and the RWB Simulator can simulate their run any given speed, usually at lower speeds so that the developer is able to check the proper behavior of the simulated process. Some simulations can even be traced backwards in the RWB Simulator. Using our RWB Library and Simulator, the process of an application development runs as follows.

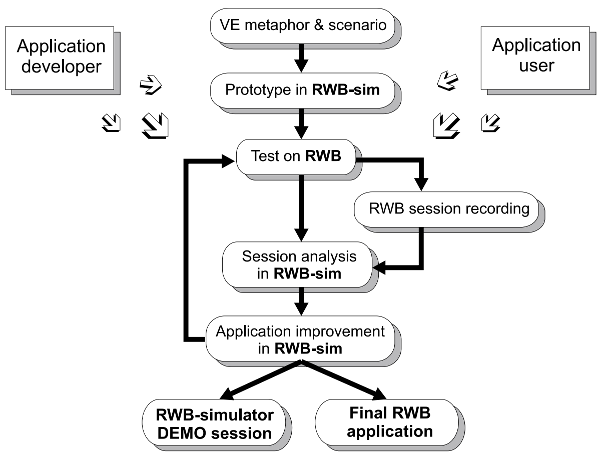

Figure 6: Application development scheme

First, there must be a clear idea about the RWB application - the application scenario. The proper VE metaphor also has to be developed.

The developer prepares a prototype of the application: the scene graph of the virtual world, basic functions and callbacks, the user interface, etc. During this preparation stage the user compiles the application in the simulator mode.

In the next step, the user/developer runs the application on the real RWB, and performs some tests and adjustments. The developer can record tracker data for having some user's interaction data, and switches back to the simulator mode. This process repeats until the implementation is finished. At later stages of this iterative process the real users of the RWB application can test the application and the developer can record their Workbench sessions. With this real user data the developer can adjust the implementation and check whether it will fulfill the real users needs or interaction abilities without them, because the developer already has their interaction data.

This way we can much better address the old problem that the best user of an application is its developer. With real user data the RWB application can be really "tuned" for the user and not the developer. We still have to learn a lot about this user-developer process, but we think that this might be the right direction.

After the final test of the application on the real

Workbench the RWB Simulator can produce images and animations for presentations. Also demo sessions can be produced for the application users to learn how to use it. These demo sessions can be previewed with the RWB Simulator version of the RWB application.

4.2 The RWB Simulator Implementation

The RWB Simulator is in fact a simulation mode of an application that uses the RWB Library. In practice it is a C++ compiler and linker option. There is also the RWB-SIM Library which has a lot of extra functions for tracing the RWB applications. The developer has thus two compilations of the application, one optimized for the workbench and the other for the simulator. For the application developer or the user it is very simple to use.

In the simulation mode, the tracker data are not read from the tracker daemon but from the tracker data file. The application then runs in the same way as on the real Workbench. On desktop systems we usually don't provide stereo.

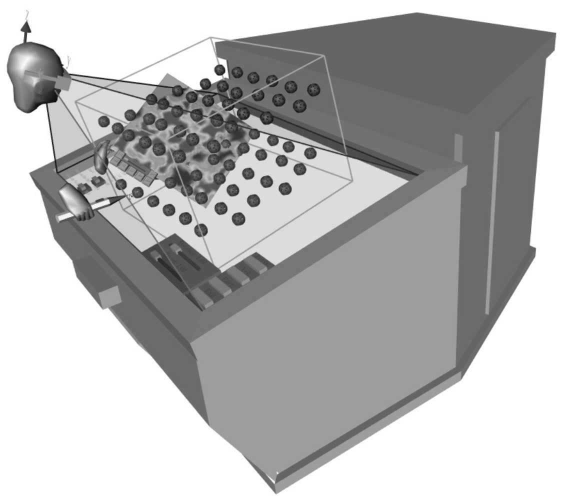

Figure 7: Molecular dynamics; FOV volume

On the real Workbench the user performs the interaction with the RWB and with the running application. The user resp. the developer can start to capture his interactions with the virtual environment into a file.

Then within the RWB Simulator, the developer sitting at a common workstation can watch what was happening on the real RWB. The application world is displayed on a top of the model of the Workbench. The developer can observe the run of the RWB application, how the user performed with it and the simulator user can navigate around the RWB model with the mouse via a trackball metaphor. The keyboard can be used to steer the simulator (e.g. pause, trace back/forward or reset simulation, reposition the user's head or the stylus). In the case when there are no interaction data for the simulator, the 3D interaction can be emulated with keyboard and mouse controls, similar to the CAVE Simulator.

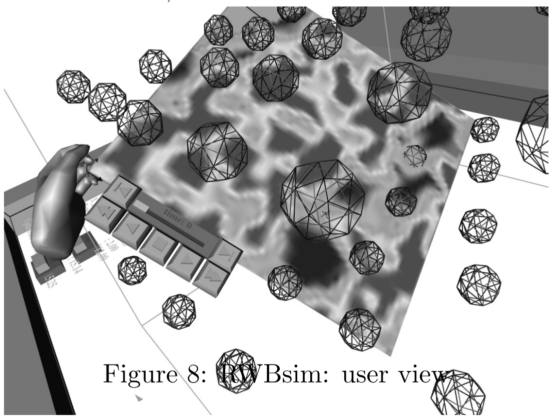

Figure 8: RWBsim: user view

In the RWB Simulator the default viewpoint is above the Workbench table overseeing the whole virtual setup with the user and the Workbench, see Figure 7. There is also the possibility to setup the view from the user's eye position, see Figure 8. Using the track-ball metaphor we can navigate with mouse around the Workbench model in the RWB Simulator

A big advantage of the RWB Simulator lies in its portability. The RWB applications can be implemented and developed on common graphic workstations with Iris Performer and the RWB Library. Currently the library works exclusively on SGI workstations. Since the Performer is now available for Linux we work on porting the RWB Library to a PC environment.

4.3 Presentation of the RWB Application

Another aspect of VR research is demonstration and presentation of results. It is not possible to take stereo/immersive pictures of a user working with an RWB application. Usually, we switch the projection to a monoscopic mode and then we adjust the perspective to align the user with the virtual world, we take a camera and make the picture. The RWB Simulator is very convenient for making pictures/animations of the RWB application, see Figures 3-12.

4.4 Collaboration in Distributed Applications

Currently, we experiment with simple collaborations within the SGI workstations in our lab. We can collaborate on the level of RWB Simulators with the Responsive Workbench.

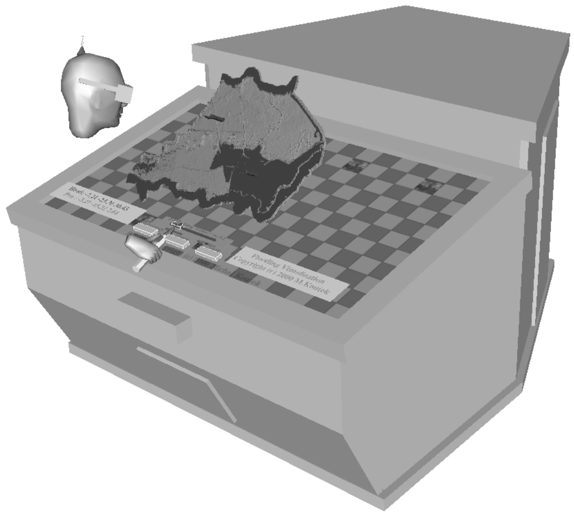

Figure 9: Delft WL|Hydraulics: visualization of the flooding simulation

We can already mention that the preliminary results show a potential of the RWB Simulator also for simulating distributed applications.

5 Examples of RWB Applications

There is a wide range of applications running on the Responsive Workbench. GIS, architectural, landscape planning/observation applications profit from the large overview, the high level of immersion and the 3D interaction. In Figure 5, the Workbench In Workbench function assists the navigation in a GIS application. The user is observing a model of the TU-Delft campus. In Figure 9, the user performs an interactive visualization of a flooding simulation.

The RWB, can also be used for various experimental simulations and applications such as shown in Figures 4 and 10, where the dynamic object manipulation with the spring-based tools is shown.

In Figures 7 and 8 an interactive molecular dynamics simulation and visualization is shown. The user is exploring the kinetic energy data of the particle system using a data color-slicer. The variety of visualization techniques can be combined to produce the best visualization for a given application. The user gets better understanding of the data thanks to the immersive visualization and the ability to interact in 3D with the VE.

6 Conclusions and Future Work

In this paper we have presented the RWB Library and the RWB Simulator, our basic environment for the development of RWB applications.

With our system, workbench applications can be developed using desktop systems as well as the RWB itself. The use of the RWB Simulator strongly increases the efficiency of application development. Most of the development time of the RWB application can be now spent on a desktop system and not on a walking between the Onyx 2 console and the Responsive Workbench.

The system provides us with new simulation and steering capabilities. We can simulate (replay) RWB applications including the user's 3D interaction at a given speed because of an adjustable virtual clock mechanism inside the RWB Library. Thanks to this we can time-correctly debug complex and complicated applications that run fast on the SGI Onyx 2, but are much slower on SGI O2s. Even with the simulator running on the Onyx 2 we could not guarantee the same timing of the RWB application on the Workbench and later in the simulator without an adjustable timer.

We use this system for several research case studies. Some of them were mentioned in this paper. The interactively most complex RWB application that we have tested with the RWB Simulator was a physics-based world for an assembly task on which we were designing the spring-based manipulation tools.

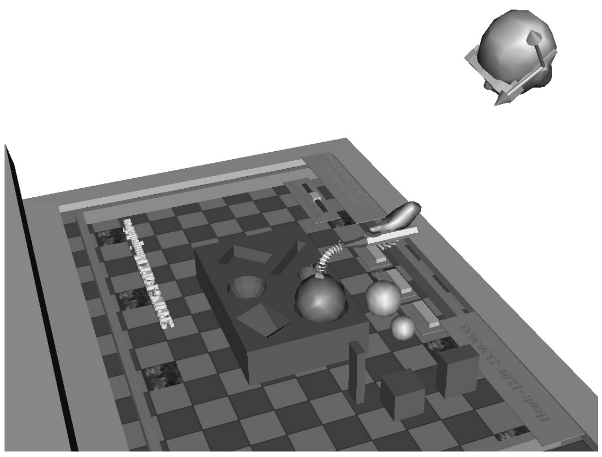

Figure 10: RWB-Simulator: the spring-fork used for manipulation of objects in the mini-world

Object collisions and the dynamic tools were simulated in a real time. We have used the RWB Simulator not only in the process of development and design of the tools and their application, but we have also produced a lot of pictures and animations with it. The reader can compare the images of the same application in the RWB Simulator (Figure 10) and the real RWB application image taken by a camera (Figure 11).

Our system is still under development and we are adding more functionality, such as an interface for *vtk* (Visualization Toolkit) [14]. Currently, we implement on the RWB a system for real-time concurrent visualization and simulation of particle systems from molecular dynamics.

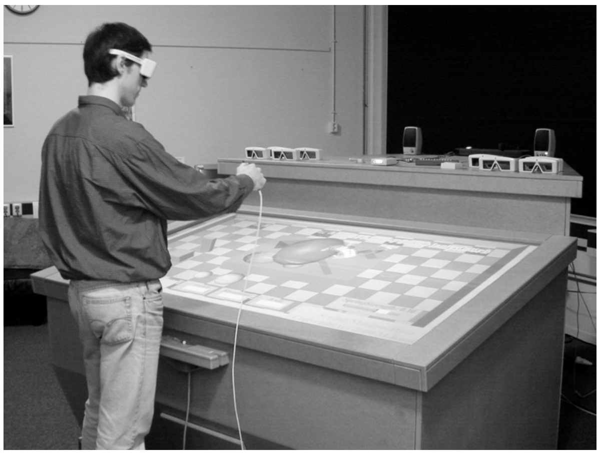
Figure 11: RWB-overview: the spring-fork used for manipulation of objects in the mini-world

REFERENCES

[1] W. Krüger, B. Fröhlich, C.A. Bohn, H. Schüth, W. Strauss, G. Wesche, *The Responsive Workbench: A Virtual Work Environment*, IEEE Computer, July 1995, pp. 42-48.

[2] P. Dai, G. Eckel, M. Göbel, G. Wesche, *Virtual Space: VR Projection System Technologies and Applications*, Internal report on AVOCADO framework, GMD, 1997.

[3] C. Cruz-Neira, T.A. Sandin, R.V. de Fanti, *Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE*, Proc. of SIGGRAPH, 1993, pp. 135-142.

[4] M. Koutek, F. H. Post, *A Software Environment for the Responsive Workbench*, Proc. of annual conference of the Advanced School for Computing and Imaging (ASCI 2001), Netherlands, 2001.

[5] M. Koutek, F. H. Post, *Spring-Based Manipualtion Tools for Virtual Environments*, Proc. of Immersive Projection Technology and Virtual Environments 2001, Springer, Stuttgart, Germany, pp. 61-70

[6] R. van de Pol, W. Ribarsky, L. Hodges, F. Post, *Interaction Techniques on the Virtual Workbench*, Proc. of Eurographics Virtual Environments '99 workshop, Springer, Vienna , Austria, 1999.

[7] D. Bowman, L. Hodges, *User Interface Constrains for Immersive Virtual Environment Applications*, Proc. of IEEE VRAIS, 1997, pp. 35-38.

[8] S. Bryson, *Approaches to the Successful Design and Implementation of VR Applications*, ACM SIGGRAPH'94, Course Notes, 1994.

[9] A. Bierbaum, C. Just, P. Hartling, K. Meinert, A. Baker, C. Cruz-Neira, *VR Juggler: A Virtual Platform for Virtual Reality Application Development*, Proc. of Virtual Reality Conference 2001, Yokohama, Japan, pp. 89-96

[10] M. Bues, R. Blach, S. Stegmaier, U. Häfner, H. Hoffman, F. Haselberger, *Towards a Scalable High Performance Application Platform for Immersive Virtual Environments*, Proc. of Immersive Projection Technology and Virtual Environments 2001, Springer, Stuttgart, Germany, pp. 165-174

[11] R.Hubbold, J.Cook, M.Keates, S.Gibson, T.Howard, A.Murta, A.West and S.Pettifer; *GNU/MAVERIK: A micro-kernel for large-scale virtual environments*, Proc. of VRST'99, ACM Symposium on Virtual Reality Software and Technology, 1999.

[12] *The RWB Library and the RWB Simulator*, http://www.cg.its.tudelft.nl/~michal/RWBlib

[13] *The CAVE Library and the CAVE Simulator*, http://www.ncsa.uiuc.edu/VR/

[14] *The Visualization Toolkit*, http://www.kitware.com/vtk.html

[15] *VR Developers Toolkit*, http://www.lincom−asg.com/VrTool/

[16] *The Simple Virtual Env.(SVE) Library*, http://www.cc.gatech.edu/gvu/virtual/SVE/

[17] *MR (Minimal Reality) Toolkit*, http://web.cs.ualberta.ca/ //graphics/MRToolkit.html

[18] *VRlib*, http://www.ait.nrl.navy.mil/ people/ekuo/vrlib-doc

[19] D. Schmalstieg, A. Fuhrmann, Z. Szalavari, M. Gervautz, *"Studierstube" - An Environment for Collaboration in Augmented Reality*, Virtual Reality - Systems, Development and Applications, Vol. 3, No. 1, pp. 37-49, 1998.