Physically-Based Manipulation on the Responsive Workbench

Bernd Fröhlich
Henrik Tramberend
German National Research Center
for Information Technology

Andrew Beers
Maneesh Agrawala
Stanford University

David Baraff
Carnegie Mellon University

Abstract

This paper describes how a physical simulation can be integrated with our Responsive Workbench system to support complex assembly tasks involving multiple hands and users. Our system uses the CORIOLIS physical simulation package extended to meet the real time requirements for our highly interactive virtual environment. We develop a new set of interface tools that exploit the natural properties of physical simulation (i.e. the superposition of forces). Our tools are based on sets of springs connecting the user's hand to a virtual object. Visualizing these springs provides "visual force feedback" of the applied forces and facilitates the prediction of the objects' behavior. Our force-based interaction concept allows multiple hands and users to manipulate a single object without the need for locking the object.

1   Introduction

In real life, many tasks require the coordinated interaction between multiple hands and multiple people. For example, mounting a light on a ceiling requires at least two hands, and draping a table cloth over a table is best done with two people holding the table cloth at the four corners. Assembly tasks are prime examples of such multi-handed, multi-person interactions. In virtual environments however, assembly tasks have been difficult to perform. There are two main reasons for this: First, while we are accustomed to using multiple hands and multiple people for assembly tasks in real life, the access of multiple hands or multiple users to the same virtual object has been mostly forbidden. A virtual object is typically locked by the first user's access. The underlying reason is that users in these systems change the position and orientation of virtual objects directly, and it is not clear how to merge the input from multiple users' hands so that the object still moves sensibly. Second, the alignment of real objects is often based on contact, friction, as well as the forces applied to these objects. These physical influences have not yet been simulated for arbitrary three-

Figure 1. Two hands are used to fit the crankshaft into the bearing.

dimensional objects in real-time virtual environments. At most, VR-systems support fast collision detection, which gives users feedback when objects start to penetrate each other [8] [7]. The problem with only providing collision detection is that there is no further help from the system on how to change the position and orientation of the virtual objects to align them properly. In real life, we just apply forces to objects we are manipulating and the superposition of all these forces, in conjunction with the physical properties of the object and the environment, determine the object's behavior. In this paper we describe a system and interaction techniques that are based on this idea and we show how multiple hands and multiple users can access and work in coordination on the same virtual object (figure 1). In our system, we compute the virtual objects' behaviors using a real-time physical simulation, which performs collision

detection between objects, manages constraints, and takes physical properties of objects into consideration. The physical simulation in combination with our interaction techniques lets multiple users perform assembly tasks in coordination.

We use our system to drive the Responsive Workbench [10] [9], a table top stereo display based on a workbench metaphor. Workbench applications typically set virtual objects right on top of the horizontal display surface. This setup is ideal for assembly tasks since most objects are within an arm's length reach to the user. Recently, the two-user Responsive Workbench [1], the PIT [3], and Studierstube [13] have been presented, which support more than one active tracked user. In such environments we have possibly four or more hands acting in coordination. In this paper we show how this multi-handed input can be handled elegantly.

The main contributions of our work are the extension and tight integration of a real-time physical simulation with our virtual environment and the development of a new user interface paradigm exploiting the inherent properties of physical simulation. Our interface tools are based on sets of springs connecting the user's hand to a virtual object. We visualize these springs to provide "visual force feedback" of applied forces, which facilitates the prediction of the objects' behavior. We demonstrate our system using an assembly task and discuss our experiences.

2   Related Work

In the past few years, several VR-systems have been built that support two-handed input, but only a few systems use two-handed manipulations on a single object. Polyshop [11] uses symmetric two-handed tools for rotating, scaling, or stretching objects. Users can also align objects via anchors and constraints. The CHIMP system [12] uses two hands for object scaling. Cutler et. al. [6] developed a variety of two-handed tools for object positioning tasks. They also presented two versions of a two-handed grab tool, which allowed them to orient objects easily using two hands, and to pass objects from one hand to the other.

There is a variety of constraint-based systems. One of the most impressive systems is SKETCH [15], which allows the user to quickly sketch a three-dimensional scenario in a monitor-based environment. Constraints are either inferred, for example by placing new objects in contact with existing objects or explicitly sketched in addition to the geometry.

In distributed VR-systems, the issue of multiple users trying to grasp the same object is rarely addressed. As an example of a system that addresses this issue, DIVE [5] uses some form of distributed locking to lock an object by the first user's access. However, this prevents tasks like handing over an object between two users.

Today, sophisticated animation packages like Alias [2] use physical simulations to produce natural looking animations. In these systems , there is no strict real-time requirements for the physical simulation like there is in a VR system. Also, the user interface is designed for one user in front of a standard two-dimensional screen

Baraff[4] describes a 2D rigid-body simulator. CORIOLIS™ is the 3D extension of this system and handles arbitrarily shaped polyhedral rigid bodies, with an emphasis on persistent contact, collision and friction. We use CORIOLIS as the physical simulator in our system.

3   Interaction

Within physical simulations, objects are controlled by forces. The simulation uses these forces to update the current position and orientation of each object depending on environmental and user defined constraints as well as the current state and physical properties of the object and the whole environment. By using forces as the method of interaction in a physically-based simulation environment, we gain two benefits. First, having more than one hand or user acting on an object does not have to be special-cased – it is naturally handled by the superposition of forces acting on an object. Second, objects move naturally during interaction, since they can obey other forces or constraints imposed on them by the environment and other objects.

To interact with an object, we have developed spring-based virtual tools. One end of a virtual spring is attached to the user's hand and the other end is attached to the object. Clearly, one spring doesn't give the user much control over an object, since the three positional degrees of freedom are only "softly" constrained (assuming a spring with rest length zero). The attached object follows the user's hand, but it can still freely rotate around the attachment point. To get more control over the object we experimented with multiple springs attached in various ways to objects. Besides the number of springs, the position at which the springs are connected to the object play an important role. We tried different spring attachments relative to

the center of the object's bounding box: With this approach it appears as if the user is holding the object by its center. The main problem with this approach is that it can be difficult to control an object when its center of mass is not close to the bounding box's center.

the object's center of mass (figure 2 a,e,h,j): This configuration works very well for one-handed positioning tasks when four or more springs are used. The object follows very directly the user's hand movements. With only three springs the object tends to swing perpendicular to the plane defined by the three points, even though all six degrees of freedom are controlled.

We found, however, that this method does not work well for situations where two or more hands try to manipulate an object, since each hand controls the object very directly.

the user's hand (figure 2 b,c,d,f,g,i): This method lets the user choose where to hold on to an object. The version with two springs is similar to holding an object with two fingers and allows the object to freely rotate around the axis connecting the two points on the inner frame. This works well for two-handed situations where the second hand controls the unconstrained degree of freedom. With four or more springs, the user has good control over all six degrees of freedom of the object, but objects tend to swing when picked up with one hand far off the center of mass - as in real life. Here the user typically reaches in and uses the second hand (figure 1) or another user helps out. This method is our default, because it gives the user explicit control about the grip onto an object. It is also shown in the accompanying video.
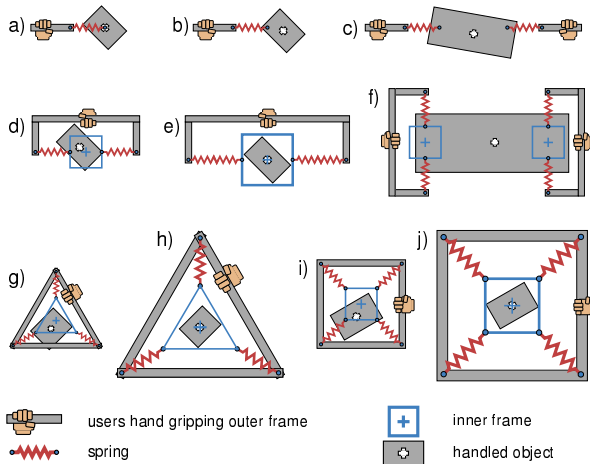

Figure 2. To explain how hand movements control objects, the concept of two rigid frames connected by a set of springs is introduced. The outer frame is directly controlled by the users hand. The inner frame is rigidly attached to the object. Different spring configurations exhibit different control properties. The springs between these frames determine which forces are applied to an object based on Hooke's law and an additional damping term to avoid uncontrolled oscillations.

Other tools like sliders, buttons, switches, menus and such, which influence some non-geometric or abstract property in the environment are also modeled as three-dimensional objects and become part of the physical simulation. The position, orientation, velocity, and/or acceleration of the tool's geometric representation is then observed by the tool and used to control its internal state. A simple example is the slider in figure 3. This integration of the user interface elements into our simulation extends the multi-handed and multi-user capabilities seamlessly to the user interface.
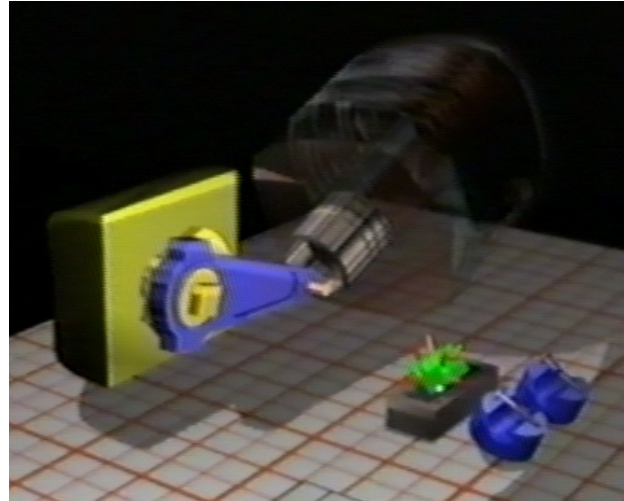

Figure 3. The slider is also part of the simulation and is operated by the 8-spring tool. It controls the transparency of the cylinder.

Constraints describe geometric relationships between virtual objects, which CORIOLIS is responsible for enforcing. Contact constraints, which keep different objects from interpenetrating, are enforced automatically by CORIOLIS . Typical linkage constraints such as keeping a point on a line and user-defined constraint types can be easily added. We make effective use of our computational resources by switching back and forth between contact constraints, which are typically computationally very expensive, and linkage and other user-defined constraints. Our current implementation allows us to predefine constraints at startup time, and also the conditions under which these constraints are added or removed from the system. A typical example used in our engine assembly scenario, shown in figure 5, is the insertion of the crankshaft into the bearing block. Once the crankshaft is approximately in place, we turn off collision detection between crankshaft and bearing block and replace these contact constraints by an axial constraint and an inequality constraint. The axial constraint aligns the crankshaft with the bearing's axis and the inequality constraint prevents the crankshaft from penetrating the bearing block. To prevent the crankshaft from slipping

out of the bearing we add a spring between the crankshaft and the bearing that keeps the crankshaft in place, but still provides some flexibility. Alternatively, we could use a point constraint, which would keep the crankshaft always in the ideal place, or two inequality constraints keeping the crankshaft within a certain tolerance around the ideal position. The solution using the spring has the advantage that removing the crankshaft from the bearing can be easily done by pulling hard enough on the crankshaft. At some point the spring virtually breaks and we remove the linkage constraints from the system and restore collision detection between crankshaft and bearing. This approach also shows that physical behavior can reduce the number of explicit state changes in an elegant manner.

4 Implementation

For the interaction with virtual objects on the Responsive Workbench we use two tracked pointing devices: a six degree of freedom Polhemus stylus with a button and a custom three-button pointer containing a Polhemus sensor. These input devices can be used by two different users or by the same user for two-handed manipulations.

For the physical simulation, we adapted the CORIOLIS™ rigid-body simulation system. CORIOLIS comes as a library of C++ classes. It does not provide any support for graphics output or file I/O. Classes representing objects, influences, and constraints can be instantiated and added to the simulation at runtime. CORIOLIS is able to simulate moderately complex environments at interactive rates.

We augmented the basic CORIOLIS kernel with additional infrastructure to make it usable in a head tracked highly interactive virtual environment like the Responsive Workbench. Head tracked systems are inherently vulnerable to low or varying rendering frame rates and latency. CORIOLIS uses an adaptive solver so simulation rates can vary by orders of magnitude depending on the actual object configuration. Therefore, interleaving the rendering and simulation tasks can lead to severe problems. A complex physical interaction can lead to long simulation cycles which would slow the rendering rate to unacceptable levels. Similarly, a visually complex scene could slow down the simulation. To overcome this problem we decoupled the simulation from the rendering task by executing it in a separate process.

For each object in the rendering scene graph, there is a *proxy* object that corresponds to an object in the CORIOLIS process. The communication between the CORIOLIS object and its proxy across the process barrier is handled through asynchronous message passing. A typical communication pattern for a CORIOLIS object and its proxy is the following: The user adds a new force (e.g. a spring) to the system and applies it to an object. The parameters of the
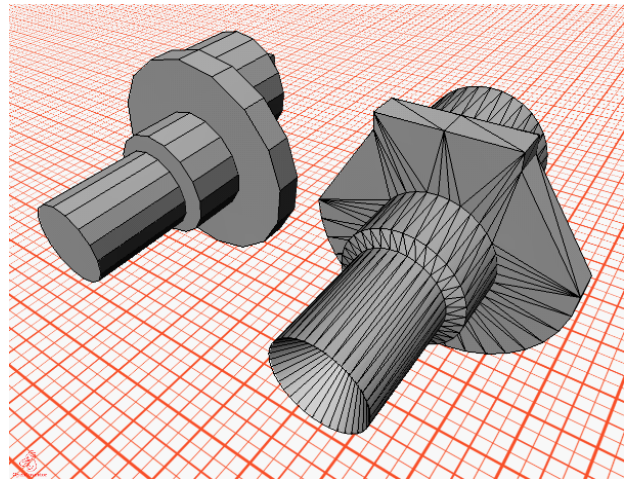

Figure 4. To save on computational resources in the simulation process while maintaining visual detail, different geometric representations are used for rendering and simulation. This figure shows the crankshaft. The reduced representation in the back contains about 15% of the polygons of the original version in the front.

force are encapsulated in a message and sent to the CORIOLIS process where the force is instantiated and added to the simulation. On the other side, each simulation-induced update to an object's position or orientation is sent back to the corresponding proxy object. The scene graph is updated so the change will be visible after the next rendering step. This implementation scheme nicely reflects the decoupling between user action and system response in contrast to standard interaction models like the one described in [14]. In CORIOLIS , a scene consists of a set of world space objects for which the physical simulation is performed. Each object can be described hierarchically, but there are no transformations contained in this object tree. Our proxy interface fully supports hierarchical scene graphs as they are common in standard graphics libraries like OpenInventor or Performer, and it takes care of their mapping into the simpler CORIOLIS object hierarchy.

When rendering complex scenes, objects are often displayed at different levels of detail to keep the frame rate constant. In our system, we use different levels of detail for the objects rendering and simulation geometries. Typically, the polygon count for our CORIOLIS objects is much lower than for the rendering objects since the simulation can only handle one to three orders of magnitude fewer polygons than high end graphics rendering hardware at interactive update rates. The simulation rate depends highly on the spatial arrangement of the objects. Depending on the

task we found that the CORIOLIS representation can often be extremely simplified (figure 4) without users noticing the difference while maintaining high visual fidelity for the rendered object.

5 Results and Discussion

Our Responsive Workbench system is hooked up to a Silicon Graphics Onyx system with four 196 MHz R10000 processors and InfiniteReality graphics. The most complex example we have used to test our system is an assembly sequence for a one-cylinder engine consisting of a crankshaft, crankshaft bearing, connecting rod, piston, and cylinder (figure 5). The rendering representation consists of nearly 6000 polygons whereas the simulation representation contains only around 400 polygons. The rendering update rate is about 24 frames per second with multi-pass shadows. The simulation update rate is between 5 and 50 updates per second depending on the spatial object configuration.

Overall the system feels very responsive as long as the number of simultaneously interacting polygons is in the range of a few hundred. If the simulation rate drops below approximately 5 Hz objects start to stick to each other. Tight peg and hole configurations cause oscillations and numerical instabilities, which sometimes break the system. We had to add at least 5 percent tolerance to avoid these problems. Since it is crucial to keep the simulation running above 5 Hz or even better 10 Hz, we have to carefully spend the available simulation time. The number of simultaneous contact constraints needs to be kept as small as possible. Techniques such as using simplified simulation geometries, switching back and forth between contact constraints and simpler linkage constraints, and not checking collisions between motionless objects are essential. Often it is unnecessary to include objects, for example the user interface elements, fully into the physical simulation, since they might not collide with other objects.

With our spring-based interaction concept, each hand tries to align the object with its orientation and position. When holding on to an object with two or more hands this can be a little confusing at first. For example when one hand is rotated by 90 degrees, the object turns only 45 degrees since the other hand tries to rotate the object back into the zero degree position. When one hand is released in this situation, the object snaps back to the position and orientation of the other hand. Sometimes this requires changing the grip on an object more often than in real life to get an object into the desired orientation. In other situations it would be desirable to be able to relax the springs during the interaction or before releasing the object.

In some ways the flexibility of our spring tools simulate the flexibility of our fingers when aligning an object with another object. Visualizing the springs gives the user an easy way to interpret the visual "force feedback" that helps show why an object behaves in a certain way. It also facilitates the prediction of the object's behavior in response to the forces applied. While these multi-spring tools could also be implemented using a single spring with torque, this approach would not be as intuitive and easy to understand.

6 Conclusions and Future Work

We have described the integration of a physical simulation with the Responsive Workbench system. Our new interaction concept is based on sets of springs connecting the users' hands to the virtual objects. This force-based approach made our system inherently multi-handed and multi-user capable enabling collaborative assembly tasks. Simple linkage constraints are used to replace computationally expensive contact constraints whenever possible.

Currently, we generate the different representations for the rendering and the simulation process manually, but we would clearly like to automate this process as much as possible. The questions are: which are the features of an object that need to be preserved for a physical simulation and how can we detect them? Is it possible to adapt mesh simplification algorithms developed for rendering to work for physical simulations?

The most important and challenging task is the improvement of the physical simulation itself. The numerical stability needs to be significantly improved to deal with real world examples. Another promising improvement is the parallelization of the collision detection method and the differential equation solution method employed by the physical simulation.
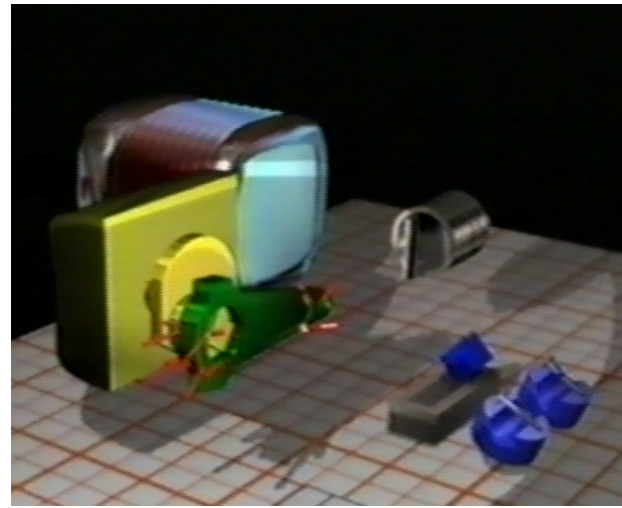
We have opened the door into a new world of interaction paradigms and user interface techniques, which we have only begun to explore. As long as unobtrusive force-feedback and haptic feedback are unavailable, a real-time physical simulation delivers the most convincing interactive feedback we have experienced to date.
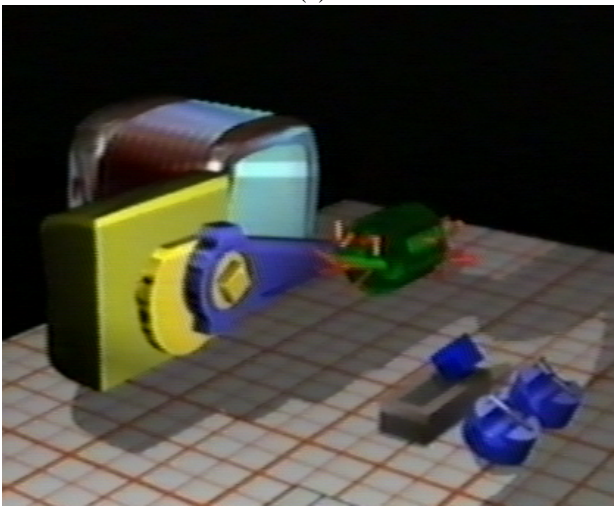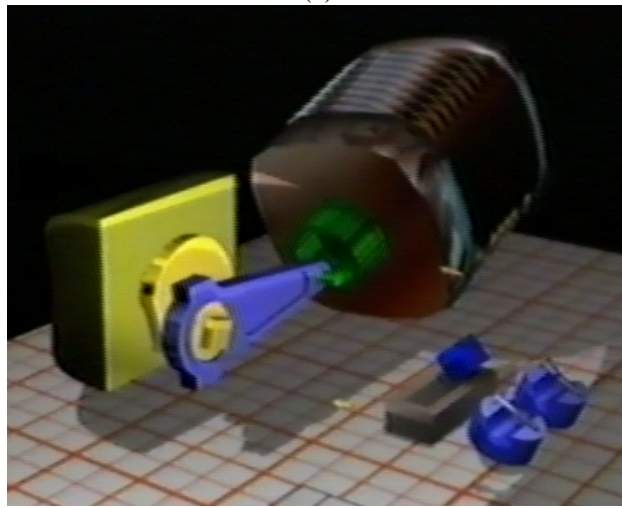
7 Acknowledgements
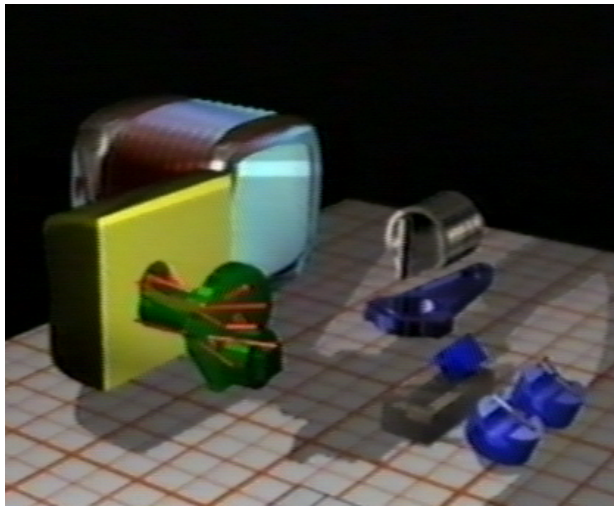
Figure 5. Four snapshots of an assembly sequence for a one-cylinder engine. (a) The user picks up the crankshaft with two hands and tries to slide it into the crankshaft bearing guided by the bearing block's surface. After the crankshaft is in place, we turn off collision detection between crankshaft and bearing and replace these geometrical constraints by an axial constraint. This greatly reduces the number of constraints in our system and results in a perfect axle/bearing attachment. (b) The connecting rod is mounted on the crankshaft. (c) For the attachment of the piston, the connecting rod is frozen to simplify the insertion of the piston's bolt into the rod. (d) After the user manages to slide the cylinder onto the piston, the engine is complete. The images were taken from sequences of the video. We recorded these sequences in monoscopic view from one display while showing a stereoscopic view to the user at the workbench on a second display.

References

[1] M. Agrawala, A. C. Beers, B. Frohlich, P. Hanrahan, I. Mc-Dowall, and M. Bolas. The two-user responsive workbench: Support for collaboration through independent views of a shared space. *Computer Graphics*, 31(3A):327–332, Aug. 1997.

[2] Alias/Wavefront. *Learning Alias V8*. Alias/Wavefront, 1996.

[3] K. Arthur, T. Preston, R. M. T. II, J. Frederick P. Brooks, M. C. Whitton, and W. V. Wright. Designing and building the pit: a head-tracked stereo workspace for two users. In *2nd International Immersive Projection Technology Workshop*, Ames, Iowa, May 1998.

[4] D. Baraff. Interactive simulation of solid rigid bodies. *IEEE Computer Graphics and Applications*, 15:63–75, 1995.

[5] C. Carlsson and O. Hagsand. DIVE — A platform for multi-user virtual environments. *Computers and Graphics*, 17(6):663–669, Nov.–Dec. 1993.

[6] L. D. Cutler, B. Fröhlich, and P. Hanrahan. Two-handed direct manipulation on the responsive workbench. *1997 Symposium on Interactive 3D Graphics*, 1997.

[7] S. Gottschalk, M. Lin, and D. Manocha. OBB-Tree: A hierarchical structure for rapid interference detection. In H. Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 171–180. ACM SIGGRAPH, Addison Wesley, Aug. 1996. held in New Orleans, Louisiana, 04-09 August 1996.

[8] T. C. Hudson, M. C. Lin, J. Cohen, S. Gottschalk, and D. Manocha. V-COLLIDE: Accelerated collision detection for VRML. In R. Carey and P. Strauss, editors, *VRML 97: Second Symposium on the Virtual Reality Modeling Language*, New York City, NY, Feb. 1997. ACM SIGGRAPH / ACM SIGCOMM, ACM Press. ISBN 0-89791-886-x.

[9] W. Krüger, C.-A. Bohn, B. Fröhlich, H. Schüth, W. Strauss, and G. Wesche. The responsive workbench: A virtual work environment. *IEEE Computer*, pages 42–48, July 1995.

[10] W. Krüger and B. Fröhlich. The responsive workbench. *IEEE Computer Graphics and Applications*, pages 12–15, May 1994.

[11] D. P. Mapes and J. M. Moshell. A two-handed interface for object manipulation in virtual environments. *Presence*, 4(4):403–416, 1995.

[12] M. R. Mine. Working in a virtual world: Interaction techniques used in the chapel hill immersive modeling program. *Technical Report 1996-029*, 1996.

[13] D. Schmalstieg, M. Gervautz, Z. Szalavari, K. Karner, F. Madritsch, and A. Pinz. Studierstube - a multi-user augmented reality environment for visualization and education. *Technical report TR-186-2-96-10, Vienna University of Technology, Austria*, 1996.

[14] P. S. Strauss and R. Carey. An object-oriented 3D graphics toolkit. In E. E. Catmull, editor, *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 341–349, July 1992.

[15] R. C. Zeleznik, K. P. Herndon, and J. F. Hughes. Sketch: An interface for sketching 3d scenes. *Proceedings of SIGGRAPH 96*, pages 163–170, August 1996. ISBN 0-201-94800-1. Held in New Orleans, Louisiana.