Appeared: Virtual Reality Software and Technology (VRST) '94, pp. 287-298, August 23-26, 1994.

Virtual and Real Object Collisions in a Merged Environment

DANIEL G. ALIAGA[1]
Department of Computer Science
University of North Carolina at Chapel Hill
Chapel Hill, NC 27599, USA

ABSTRACT

See-through head-mounted display capability is becoming an important part of Virtual Environment applications. In such applications, it may be desirable to model the physical behavior of the virtual objects and their interaction with the real objects. This paper describes a software system which integrates interactive collision detection, collision response and see-through head-mounted displays. The system employs a static model of the real world environment and allows for arbitrary convex virtual objects to be placed in the environment. The user may control the positions and velocities of the virtual objects. An approximately constant time collision detection algorithm and a Newtonian Mechanics based single point contact collision response is used to model the apparent physical interaction of the virtual and real objects for moderately complex environments.

Keywords: Virtual Reality, See-through Head-Mounted Displays, Collision Detection, Dynamics, Parallel.

1. Introduction

1.1 Motivation

Among the original applications considered part of Virtual Environments are head-mounted display (HMD) applications. A typical HMD setup consists of a powerful graphics engine, one or more tracking devices and a head-mounted display. The user experiences the illusion of being in a (synthetic) world where the images seen are generated by a computer program. By using optical lenses or video camera technology it is also possible to present the user with images of the virtual environment and the real environment simultaneously. A helmet with such characteristics is called a see-through head-mounted display.

[1]Internet email: aliaga@cs.unc.edu

See-through capability opens up an even larger number of potential applications. A historically early example of the merging of the virtual environment with the real environment is a helicopter pilot's helmet where the pilot can see information about the helicopter's orientation, speed and location as well as a cross-bar that could be tracking an enemy; all superimposed on the pilot's view of the real environment. Soon we could hope to see applications such as an architectural design system. Such a system could allow for an architect to make actual-size modifications to an existing building or for a home-owner to decorate an empty house. A similar application, could allow children to design and build a virtual toy which could be used simultaneously with real toys.

The real world, obviously, correctly models our physical environment, such as the effects of gravity, friction and collisions. In future applications of merged virtual and real environments, it may become useful to model the laws of nature; otherwise, the interaction of the two worlds may not be convincing at all. A significant amount of work must be done for a virtual environment to convincingly simulate such properties. Consider an office environment where the user has a virtual notepad. It would not be convincing if when the notepad is placed on the table, it apparently falls through the table. Similarly, in the previous example of a home-owner decorating an empty house, the home-owner might desire the addition of a sliding door or venetian blinds. These virtual additions should properly interact with the surrounding (real) house.

1.2 Integrated System

In this paper, I demonstrate the design and implementation of an *integrated* system, which uses computational power readily available today, for modeling interactive collision detection and collision response for moderately complex environments containing both virtual (computer generated) objects and real objects.

I will be referring to this system as Virtual and Real Object Collisions (or VROC). It uses a see-through head-mounted display together with a powerful graphics engine (also developed here at UNC-Chapel Hill) to present to the user virtual objects superimposed on a real environment. The user provides a model of the static real environment which describes the positions and sizes of the real objects. The user then creates any number of virtual objects. A hand-held tracker is provided with which the user can grab and control the linear and angular velocities of the virtual objects. The system constantly performs collision detection and computes a Newtonian Mechanics based collision response to model the interactions (i.e. collisions) between virtual and real objects, as well as the interactions among the virtual objects themselves.

2. Previous Work

Over the last decade, multiple approaches have been developed for collision detection and collision response. No one collision detection or collision response algorithm can be said to be the optimal solution. One must take into consideration the application being designed and determine what information can be made available to the algorithms and what information will be needed from the algorithms.

Hahn [1988], Baraff [1989], Moore and Wilhelms [1988] assume (convex) polyhedral objects for their collision detection and collision response systems of rigid bodies. The general approach is to consider all possible intersections of vertices, edges and faces of polyhedral objects. Canny [1986] suggested another collision detection method for polyhedra moving among polyhedral obstacles. Cameron [1990] explored the possibilities of 4-space intersection testing. Hubbard [1993] introduced a fast and interruptible collision detection algorithm.

Many mathematical models exist that implement physical behaviors. But, since even a single aspect of physical behavior can be difficult to model, implementations usually only model a small number of physical behaviors and perhaps crudely approximate a few others. Wilhelms et al. [1988] present the general idea behind the use of physical simulations. Goldstein [1950] describes most of the issues of Classical Mechanics, while Baraff [1992] provides a good general overview of rigid body dynamics for 3D animations. Li and Canny [1990] describe a model for rigid bodies with rolling constraint. Baraff [1993] gives a formulation for contact forces between objects in resting contact. Other methods allow for objects to deform upon impact. For these methods, a quite complex analytical approach is taken. Pentland and Williams [1989] use vibration-mode ("modal") dynamics, a method of breaking down non-rigid dynamics into the sum of independent vibration modes.

Various integrated dynamics systems have been developed in the past decade: Interactive Dynamics [Witkin90], ThingWorld [Pentland90], Bolio [Zeltzer89], Virtual Erector [Schröder90]. Each approaches computational dynamics in a different way, specializing in modeling certain behaviors. Few systems exist that have integrated virtual and real world imagery. ARGOS [Drascic93] provides a virtual pointer into the real world; Bajura et al. [1992] have experimented with superimposing ultrasound images of a fetus registered in place over a pregnant women's womb.

3. Implementation

3.1 Collision Detection

In order to maintain the interactive performance of the system, selection of a fast collision detection method is essential. Lin and Canny [1991,1992] describe a method which integrates well with a dynamics system. A typical dynamics simulation advances through time by taking small time steps. The algorithm by Lin and Canny provides approximately constant time collision detection between convex polyhedra from one frame to another by assuming that an object's position and orientation will not drastically change from one frame to another.

I implemented a version of this algorithm. The essentials behind the algorithm are rather simple. Given a pair of convex polyhedra, determine the closest features of the objects. For a 3D polygonal object, the features are vertices, edges and faces. Assuming a small enough time step, from one frame to another the objects will perhaps have rotated a few degrees and closed the distance between them. Figure 1 depicts what the configuration of the closest features might be during three contiguous frames.

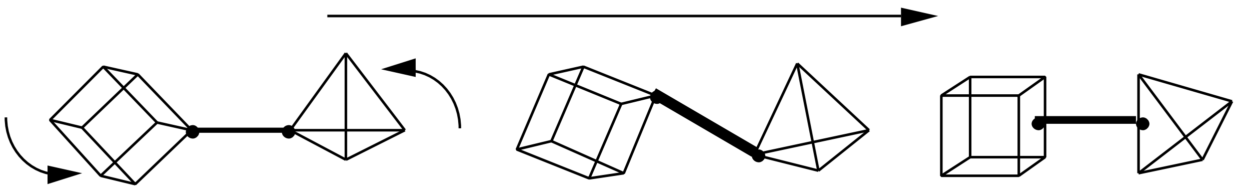
Figure 1: Closest features - three contiguous frames.

The angular velocity of the objects caused the closest features to change, but only to the next adjacent feature; namely from a vertex to the face (or edge) adjacent to it. Given larger angular velocities, it might take a few more checks of the adjacent features. In any case, the algorithm provides approximately constant time distance checking. When the distance between the features is less than a tolerance value, the objects are considered to have collided.

3.2 Collision Response

Once two objects have collided, a response must be computed. The VROC system assumes that all objects are rigid and have nearly inelastic properties. Furthermore, only single point contact between a pair of objects is modeled (since all objects are convex, this is generally the case).

I implemented a collision response based on Hahn [1988] and Moore & Wilhelms [1988] work. The collision response computation assumes that the point of contact, velocities at collision time and an orthogonal collision frame are provided. Based on the conservation of linear and angular momentum, the new velocities can be obtained as in Eq. 1 (once R is computed).

$$\begin{aligned} m_1\bar{v}_1 &= m_1 v_1 + R \qquad & I_1\bar{w}_1 &= I_1 w_1 + p_1 \times R \\ m_2\bar{v}_2 &= m_2 v_2 - R \qquad & I_2\bar{w}_2 &= I_2 w_2 - p_2 \times R \end{aligned} \qquad \text{Eq. 1}$$

The variables m, I, v, w describe each object's mass, inertia tensor matrix, linear velocity and angular velocity. The p vector is the relative vector from each object's center of mass to the point of contact. R is the impulse transfer vector (computed by inverting a 15x15 matrix). Each object has its own elasticity coefficient. In order to simulate (slightly) elastic collisions, the computed R vector is scaled by the minimum of the 2 elasticity coefficients.

3.3 Time

The collision detection and collision response algorithms are combined to form a dynamics simulation. This implies that all the computations must be parametrized by time. The user must specify the time step to use to go from one frame to the next frame. The main problem with key-frame collision detection is that objects with large velocities might penetrate or pass through each other in one frame transition. Given the maximum linear velocity and a collision distance (largest distance at which two objects are considering to have collided [Lin91]), it is possible to divide the frame time step into internal time steps. The internal time step is small enough to guarantee that no two objects will totally pass through each other. If at the end of an internal time step, the object pair is already in collision (penetration has occurred), a binary search method through time is used to find the time of collision to within a specified tolerance.

3.4 Optimizations

3.4.1 Scheduling scheme

Since objects have continuous motion, it is possible to construct a sorted list of possible collision times (a heap structure is used) [Lin92]. Given the distance between two objects, the bounds on the maximum linear velocity and linear acceleration, it is possible to predict the earliest time at which an object pair could collide. Since in addition objects have angular velocities, the difference between the radius of the inscribed sphere and the radius of the circumscribing sphere of each object must be subtracted from the inter-object distance for a safe prediction.

3.4.2 Static and Dynamic Objects

Since it is not known which objects will collide in an environment, it is necessary to perform collision checking on all possible pairs. This gives a maximum of $O(n^2)$ collision pairs, where n is the number of objects. Fortunately, in the environments that the VROC system tries to simulate, many of the objects (virtual or real) are not expected to move (i.e. tables, monitors, etc.). These objects are considered *static* and no collision checking is done between two static objects. For example, if an environment uses 100 static objects to construct a desktop and only one moving (*dynamic*) object, then only 100 object pairs are checked as opposed to the more than 10,000 pairs that would have to be checked otherwise.

3.4.3. Contact

Real world objects are never perfectly inelastic. In fact, most real objects will come to rest on a surface relatively quickly. The larger number of collisions that this will cause requires more computation and will reduce performance. Thus, it becomes necessary to model contact between objects in a more efficient manner.

The VROC system implements a simple contact scheme. When an object's linear and angular displacement fall below a threshold, the object is put into a contact state. The object does not get affected by gravity and simply rests on top of the surface it came into contact with. This model works well for many convex objects; but, other types of objects, which come to rest with multiple contact points, may require more sophisticated contact models. In many cases though, it might suffice to use the convex hull of the objects to determine the contact.

3.5 Parallelization

3.5.1 MIMD Requirements

The VROC system can be parallelized using a MIMD architecture. Each processor of the MIMD machine performs a portion of the collision detection and collision response computations. Furthermore, the object database is distributed across the parallel machine, thus increasing the potential number of objects. A message passing scheme is required to send object update messages between the multiple processors.

The VROC system is implemented on Pixel-Planes 5 [Fuchs89], though the original version was prototyped on a HP-750 TVRX-T4 workstation. Pixel-Planes 5 is a high-performance, scalable multicomputer for 3D graphics. Pixel-Planes 5 has a front-end of typically 10 to 40 processors (Intel's i860s). These general-purpose processors can be programmed by the user to perform application computations and interact with the fast rendering hardware.

3.5.2 Distribution

A portion of the VROC system runs on each processor. Recall that the collision detection scheme potentially requires a check to be performed between all possible object pairs. These checks can easily be performed in parallel. Furthermore, each processor will compute the collision response for the object pairs it stores.

The set of object pairs that have to be checked for collisions is constructed based on the static model of the real world and on the set of virtual objects that "co-exist" with the real objects. Recall from Section 3.4.2, that the number of object pairs is typically significantly less than n^2, where n is the number of objects in the merged environment. The object pairs are distributed in a round-robin fashion among the multiple processors. Each processor will construct a collision heap for the object pairs it owns. Consequently, each processor will only have to instantiate a subset of the total number of objects. An object may reside on multiple processors, but few objects will exist on all processors.

If an object pair is determined to be in collision, the associated processor will compute the collision response. Typically, each processor will only need to compute a single collision response. If other processors encounter a collision, they will compute their own collision response. Afterwards, processors that computed a collision response must broadcast the new velocities to all processors that have a copy of the objects involved in the collision. Hence, on average, the collision response computations for multiple simultaneous collisions across the system take the same amount of time as one collision response (though some additional time is needed for the update messages sent between processors).

3.6 See-through Head-Mounted Display

The see-through head-mounted display used by the VROC system was developed by the head-mounted display research group at UNC-Chapel Hill Computer Science, in the spring of 1992. It is a prototype built (with off-the-shelf components) to gain experience for a wide field of view model with custom optics and CRT. It is an optical see-through HMD which uses a pair of 2-inch LCD displays that project the computer generated image onto a pair of half-silvered mirrors (Figure 2, see Color Plates Section). The user's eyes then perceive a combined image of the real world and the computer generated world. In the background of Figure 3 (see Color Plates Section) is a monitor displaying the computer generated image which the user sees superimposed on the real staircase.

4. Examples

4.1 Virtual Environment Example

Arbitrary environments can be designed in the VROC system. The following sections present various environments I have modelled. None of the animations shown in this section required any special coding. They are simply different input files given to the VROC system. Figures 4 demonstrates an office environment consisting of a workstation, desk and bookshelf. Initially a virtual cuboid object is resting on top of the bookcase. A small heavy (yellow) ball pushes a cuboid object so that it falls off the bookcase onto the tabletop knocking the tabletop objects in multiple directions (frame rate: 15-18 stereo frames per second).

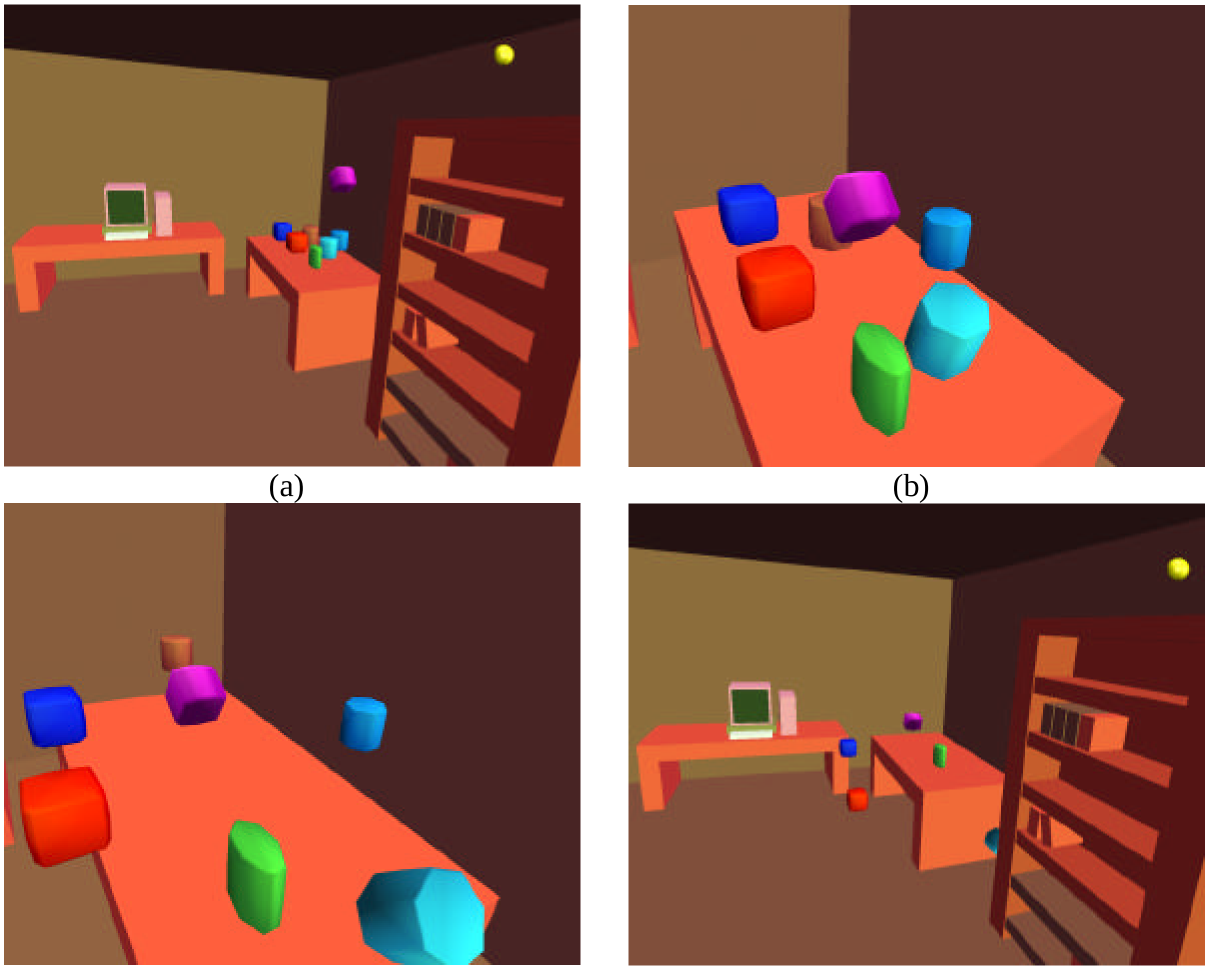

Figure 4(a-d): Virtual Environment Example, (a) box is about to hit the tabletop, (b-d), objects disperse**.**

4.2 Merged Environment Examples

First, the user must first provide a model of the static real world. In the optical see-through HMD, the real objects are drawn in the background color (i.e. not drawn); for demonstration purposes, the following figures have the real objects drawn in wireframe. The virtual objects are then superimposed over the real objects. VROC will model the proper interaction between the virtual and real objects. The images in Figure 5 and 6 (see Color Plates Section) were taken by placing a small camera behind the left half-silvered mirror of the see-through HMD (frame rate: 18-21 stereo frames/second). Note the disparity between the real and virtual hand; an example of the calibration problem (Section 5.1).

5. Observations

5.1 Technology Problems

5.1.1 Overview

The VROC system integrates collision detection, collision response and head-mounted display technology to present to the user a merged world of virtual and real objects interacting. Having implementing VROC, it is apparent that the following technology problems remain to be solved in order to bring such Virtual Environment applications into the practical world.

5.1.2 Environment Complexity

In order to create a merged virtual and real world, the computer system must know where the static real world objects lay. As the environment complexity increases, the creation of the real world model becomes an even more time-consuming task. Creating a model of the real world does not only include creating a correct polygonal representation, but also creating the appropriate textures and coloring for the virtual objects. The virtual objects that the user can manipulate should also seem as real as their real world equivalent.

The Lin and Canny [1991,1992] collision detection algorithm can be expanded for concave objects. Namely, decompose each concave object into a hierarchy of convex components (storing at each level the convex hull of the current subtree). Given the trees of two objects, start at the root and test for intersection using the convex hulls. Thus the same basic algorithm can be used for general (concave or convex) objects allowing for more complex scenery.

5.1.3 Feedback

How real the interaction of the virtual and real worlds appears to the user is not totally dependent on the visual cues from the head-mounted display. Other sensory input is also needed.

Sound feedback is important [Astheimer93]. When two real world objects collide, a specific sound is produced dependent on the objects involved. Similarly, when a virtual and a real object collide, a sound should be emitted. Stereo or 3D sound would improve the realism of the merged worlds. Current audio technology is advanced enough to produce such effects reasonably well.

A more important (and difficult to implement) feedback is force feedback. The user may have a virtual object in his hand. The object might be considerably larger than his hand. It would be helpful if the user could feel when the virtual object's surface has collided with a real object. For example, an application which allows the user to place virtual furniture in an empty real room, could give the user force feedback when a virtual sofa being placed hits a wall.

Force feedback is not only useful for virtual and real object interaction but also for user and virtual object interaction. The user may wish to touch and feel the contours of a virtual object. The illusion of realism would certainly be improved if the user could run his hand along the stairs of the environment of Section 4.2 and feel the presence of the virtual balls. Another example is a sculpting environment. Tactile feedback is essential in order to provide an effective sculpting tool.

5.1.4 Static Calibration

Proper calibration of the static real objects and their computer generated counterparts depends not only on a properly measured model but also on the compensation for the optical distortion generated by the see-through head-mounted display and the approximate perspective computations used for the virtual objects and computer models of real objects.

Solving the calibration problem would help to improve the apparent location of real objects from within the head-mounted display thus enabling precise collision responses and other feedbacks (sound, force, tactile, etc.). Furthermore, virtual objects could be accurately obscured (partially obscured or totally obscured) by the real objects in front of them.

5.1.5 Object Tracking

All of the above environments have assumed a static model of the real world. This limits the range of possible environments. It is not clear how to go about removing that assumption. Information about the movement of real objects could be obtained from a tracker placed in each of the moving objects. Although this is implementable in the very near future, it unfortunately still does not allow total freedom of movement. Another approach could be to use imaging technology and reconstruct from a 2D camera view of the world, the 3D object's contained in it. Each 3D object would be tracked by comparing the current frame to the previous frame [Tomasi92]. In any case, this is certainly still a difficult problem.

5.1.6 Head-Mounted Display Technology

The head-mounted displays used with the VROC system have a Polhemus 3Space Tracker (30Hz update rate) and a Polhemus 3Space Fastrak (up to 60Hz update rate). Studies done here at UNC Computer Science by Mine [1993], have measured the lag induced by these trackers to be from 10 to 30 milliseconds. If you add to this the refresh delay required for drawing a frame for each eye and the lag induced by the graphics system pipeline, you can get delays of 60 milliseconds (Pixel-Planes 5) or more (dependent on the environment complexity). This lag, though it sounds like a small amount of time, is very noticeable especially with a see-through head-mounted display. It makes the illusion of virtual objects lying on real objects much less convincing.

For example, consider a user turning his head at a rate of 200 degrees/second [Bishop84] (which is a perfectly comfortable speed; studies have shown that people regularly turn their head at speeds above 300 degrees/second; in fact, fighter pilots turn their heads at speeds in excess of 2,000 degrees/second). If the user turns his head for one second and the combined tracker and graphics system introduces a lag of only 50 milliseconds, the generated image will be off by approximately 10 degrees. A typical head-mounted display has a field-of-view of 60 degrees. Thus, the image will be shifted to one side by one sixth the display resolution!

Multiple methods to compensate for this lag are being developed. A tracker with a high update rate is beneficial. Reducing the rendering pipeline latency would also reduce the lag. Unfortunately, there will always be some delay that cannot be easily overcome. Thus, prediction methods are also being considered as viable solutions and in fact used in some systems, such as flight simulators.

The limited field-of-view (FOV) of most head-mounted displays (the optical see-through HMD used has a FOV of only 30 degrees) makes it easy for the user to get lost in the environment. I observed that rapidly moving objects and complex scenery often confused the user when only a small FOV was available; thus the complexity of the environments used was not limited by VROC's performance, but by the FOV of the see-through HMD used.

5.2 Parallelism Problems

5.2.1 Object Pair Distribution

VROC distributes the object pairs of the simulated environment across the multiple processors. Each processor only needs to instantiate a subset of the total number of objects. This reduces the number of update messages sent between objects when a collision occurs as well as the memory requirement per processor.

Rather than using a round-robin distribution method, a more complex method could be used. This method would distribute the object pairs among the processors in such a fashion as to minimize the number of processors where an object is instantiated.

5.2.2 Simultaneous Collisions vs. Non-simultaneous collisions

VROC distributes the collision detection and collision response computations among the processors. Recall from Section 3.5.2 that, on average, a different processor is used to compute each collision response for a set of simultaneous collisions.

If a large number of *non*-simultaneous collision responses need to be computed over a short period of time, performance is degraded since the potential parallelism offered by the multiple processors is not fully exploited. For example, assume that collision detection is trivial and during the next 100 subframes, 100 *non*-simultaneous collision responses need to be computed. It will take the same amount of time to compute the collision responses no matter how many processors are present (in fact, when more processors are present, more update messages need to be sent); a single Pixel-Planes 5 front-end processor can compute approximately 98 collision responses per second (while still performing collision detection and message passing). In order to efficiently handle environments where a large number of *non*-simultaneous collision responses are needed, it might be appropriate to distribute a single collision response computation among the multiple processors rather than employing only one processor to compute a single collision response.

6. Conclusions

To summarize, the intent of VROC was to create an integrated system to model the interaction of virtual and real objects in a see-through head-mounted display system. With this system, the problems that still need to be solved for such systems have been exposed. Hopefully the observations made and problems presented (more complex real world models, calibration, larger field-of-view see-through head-mounted displays, reduced lag, efficient modeling of additional physical behaviors) will benefit further research of such systems.

7. Acknowledgments

I would like to thank my professors, Henry Fuchs, Gary Bishop and Dinesh Manocha for their patience and wisdom. I would like to thank the additional people who have helped me throughout this project; Jack Goldfeather and John Halton for answering my multiple questions, Anselmo Lastra for answering my questions regarding Pixel-Planes 5, Rich Holloway for assisting with the see-through head-mounted display hardware, Mark Mine, Marc Olano and especially William Mark and Amitabh Varshney for comments on this paper.

Grants from the National Science Foundation, Advanced Research Projects Agency and the Supercomputing Center for Science and Technology made this work possible.

References

[Astheimer93]	Astheimer P., "What you See is What you Hear - Acoustics Applied in Virtual Worlds", *IEEE Symposium on Research Frontiers in Virtual Reality*, pp. 100-107, Oct. 25-26, San Jose, CA, 1993.
[Bajura92]	Bajura M., Fuchs H., Ohbuchi R., "Merging Virtual Objects with the Real World: Seeing Ultrasound Imagery within the Patient", *Computer Graphics*. vol. 26, no. 2, July 1992.
[Baraff89]	Baraff D., "Analytical Methods for Dynamic Simulation of Non-penetrating Rigid Bodies", *Computer Graphics (Proc. SIGGRAPH)*, vol. 23, no. 3, pp. 223-232, July 1990.
[Baraff92]	Baraff D., "Rigid Body Dynamics", *Computer Graphics Course Notes: An Introduction to Physically Based Modeling (Proc. SIGGRAPH)*, pp. H3-H29, 1992.

[Baraff93]	Baraff D., "Issues in Computing Contact Forces for Non-Penetrating Rigid Bodies", *Algorithmica*, vol. 10, no. 2/3/4, pp. 292-352, August-October 1993.
[Bishop84]	Bishop G., "Self-Tracker: A smart Optical Sensor on Silicon", Ph.D. Dissertation, University of North Carolina at Chapel Hill, TR. 84-002, 1984.
[Cameron90]	Cameron S., "Collision Detection by Four-Dimensional Intersection Testing", *IEEE Transactions on Robotics and Automation*, vol. 6, no. 3, pp. 291-302, June 1990.
[Canny86]	Canny J., "Collision Detection for Moving Polyhedra", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 2, pp. 200-209, March 1986.
[Drascic93]	Drascic D., Grodski J., Milgram P., Ruffo K., Wong P., Zhai S., "ARGOS: A Display System for Augmenting Reality", *INTERCHI* (technical video), Amsterdam, April 1993.
[Fuchs89]	Fuchs H., Poulton J., Eyles J., Greer T., Goldfeather J., Ellsworth D., Molnar S., Turk G., Tebbs B., Israel L., "Pixel-Planes 5: A Heterogeneous Multiprocessor Graphics System Using Processor-Enhanced Memories", *Computer Graphics (Proc. SIGGRAPH)*, vol 23, no. 3, pp. 79-88, July 1989.
[Goldstein50]	Goldstein H., *Classical Mechanics*, Addison Wesley, Reading, MA, 1950.
[Hahn88]	Hahn J., "Realistic Animation of Rigid Bodies", *Computer Graphics (Proc. SIGGRAPH)*, vol. 22. no. 4, pp. 299-308, August 1988.
[Hubbard93]	Hubbard P., "Interactive Collision Detection", *IEEE Symposium on Research Frontiers in Virtual Reality*, pp. 24-31, Oct. 25-26, San Jose, CA, 1993.
[Li90]	Li Z., Canny J., "Motion of Two Rigid Bodies with Rolling Constraint", *IEEE Transactions on Robotics and Automation*, vol. 6, no. 1, pp. 62-72, February 1990.
[Lin91]	Lin M., Canny J., "A fast algorithm for incremental distance calculations", *IEEE International Conference Robotics and Automation*, pp. 1008-1014, 1991.
[Lin92]	Lin M., Canny J., "Efficient Collision Detection for Animation", *Third Eurographics Workshop*, Cambridge, England, September 1992.
[Mine93]	Mine M., "Characterization of End-to-End Delays in Head-Mounted Displays", University of North Carolina at Chapel Hill, TR 93-001, 1993.
[Moore88]	Moore M., Wilhelms J., "Collision Detection and Response for Computer Animation", *Computer Graphics (Proc. SIGGRAPH)*, vol. 22, no. 4, pp. 289-297, August 1988.
[Pentland89]	Pentland A., Williams J., "Good Vibrations: Modal Dynamics for Graphics and Animations", *Computer Graphics (Proc. SIGGRAPH)*, vol. 23, no. 3., pp. 215-222, July 1991.
[Pentland90]	Pentland A., Essa I., Friedmann M., Horowitz B., Sclaroff S., "The ThingWorld Modeling System: Virtual Sculpting By Modal Forces", *Proceedings of 1992 Symposium of Interactive 3D Graphics*, vol. 24, no. 2, pp. 143-144, Snowbird, Utah, March 25-28, 1990.
[Schröder90]	Schröder P., Zeltzer D., "The Virtual Erector Set", *Proc. 1990 Symposium on Interactive 3D Graphics*, 1990.

[Tomasi92] Tomasi C., Kanade T., "Shape and Motion from Image Streams under Orthography: a Factorization Method", Intl. Journal of Computer Graphics, Vol 9, no. 2, pp. 137-154, 1992.

[Wilhelms88] Wilhelms J., Moore M., Skinner R., "Computer Animation Based on Dynamic Simulation", *Proc. International Conference on Computer Graphics*, pp. 85-95, 1988.

[Witkin90] Witkin A., Gleicher M., Welch W., "Interactive Dynamics", *Proc. 1990 Symposium on Interactive 3D Graphics*, 1990.

[Zeltzer89] Zeltzer D., Pieper S., Sturman D., "An Integrated Graphical Simulation Platform", *Proc. Graphics Interface*, London Ontario, pp. 266-274, 1989.

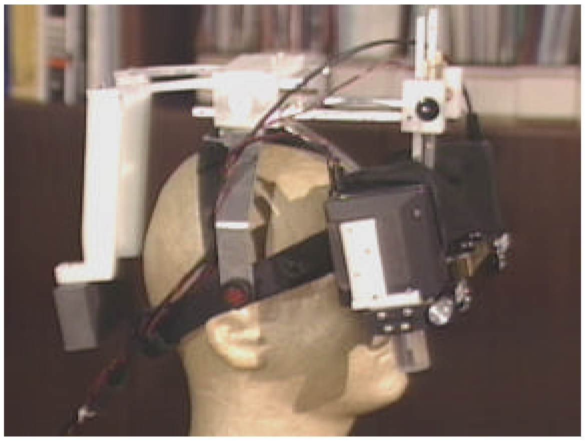
Figure 2: Optical See-through HMD

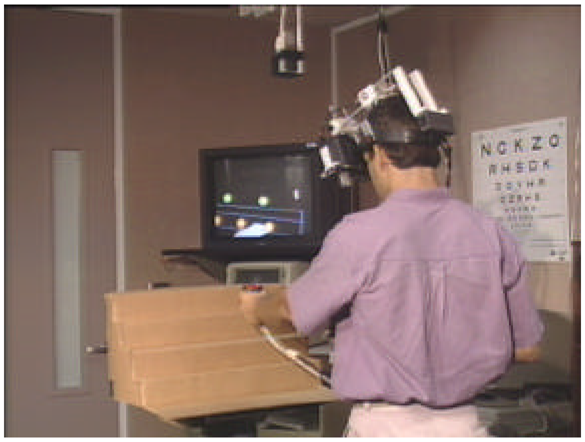

Figure 3: Optical See-through HMD in use

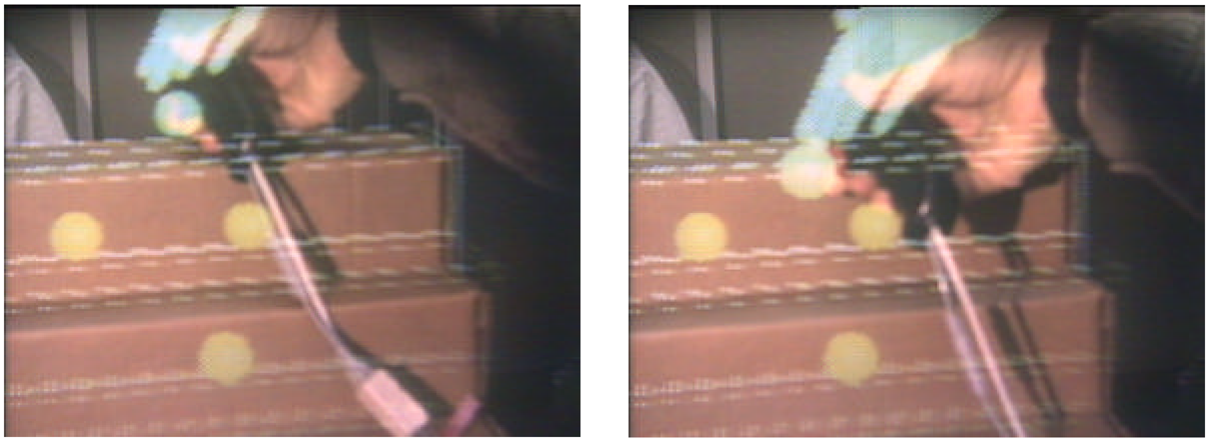
(a) (b)

Figure 5(a-b): (a) the user grabs a virtual object, (b) and throws it against the staircase.

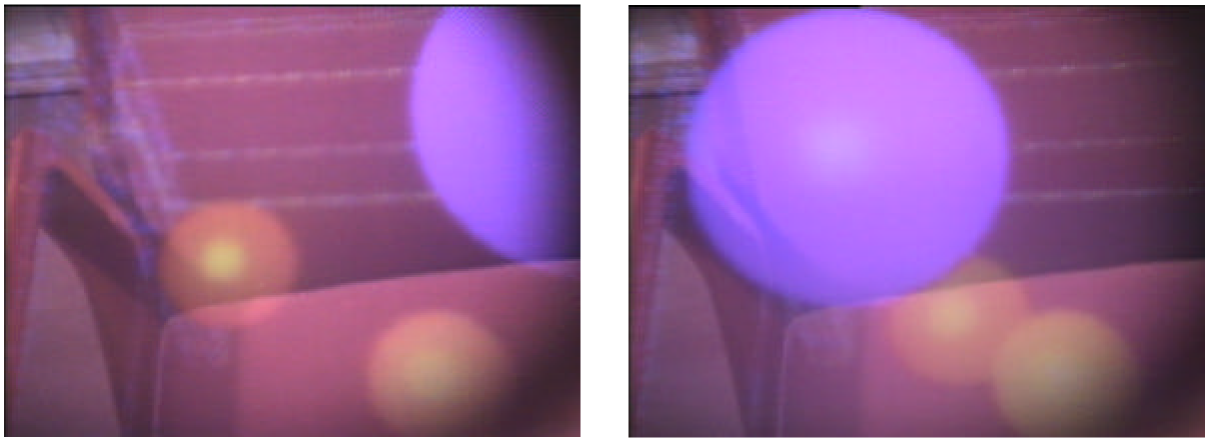
(a) (b)

Figure 6(a-b): (a) user places larger ball over a chair, (b) drops ball towards chair.