HaptiCast: A Physically-Based 3D Game with Haptic Feedback

Sheldon Andrews Javier Mora Jochen Lang Won Sook Lee

SITE, University of Ottawa
{sandr071 | jmora091 | jlang | wslee}@site.uottawa.ca

Abstract

Modern haptic technology allows users to receive realistic kinesthetic and tactile cues in a computer generated environment. When applied to video games, it gives players a higher sense of immersion as well as new and interesting ways to interact with the game environment. In this paper, we present a game which acts as an experimental framework for assessing haptic effects in 3D games. In *HaptiCast*, players assume the role of a wizard with an arsenal of haptically-enabled wands which they may use to interact with the game world. We discuss the integration of haptic feedback in this first-person shooter style video game that uses a "vanilla" 3D game engine. The haptic rendering algorithms and effects used in our approach are presented, along with some non-haptic features which enhance this modality.

Keywords: haptics, gaming, physics, force feedback

1 Introduction

Haptics refers to technology which stimulates the user's sense of touch. Users are able to literally touch and feel characteristics about computer generated objects such as texture, roughness, viscosity, elasticity, and many other characteristics. The human tactile and kinesthetic senses are stimulated through computer controlled forces which convey to the user a sense of natural feel about a virtual or remote environment.

The applications of haptic technology are widespread. For instance, in combination with a visual display, haptics technology may be used to train people for tasks requiring hand-eye coordination, such as surgery or ship docking maneuvers. Haptics may also be used for entertainment applications, such as video games. Players feel the physical properties of in-game objects, adding an extra level of interaction that traditional interface devices do not offer.

Currently, there is a variety of haptic interfaces available. Some devices, such as the Logitech Rumblepad or Microsoft Wingman, offer 2 degrees-of-freedom (DOF) interactivity and display simple haptic effects to the user— open-loop vibrotactile feedback, predefined force feedback signals. More sophisticated devices, such as the Phantom Omni and Novint Falcon, offer a higher level of interactivity by providing 6 DOF input and 3 DOF feedback to the user.

1.1 Haptics in Gaming

There are many games currently take advantage of the haptic effects offered by mainstream haptic device. For

example, in a car racing game, players may feel vibrations in their joysticks or steering wheels as they drive over a rough section of road. Or players of an action game may feel a rumble from their mouse as rockets shoot past their heads. While these devices can increase the level of immersion experienced by the user, we feel their use in games is often trivial or poorly planned. Granted, these devices cannot offer the level of interaction which is offered by modern haptic devices, but this is something which we believe will soon change.

Nintendo's president, Satoru Iwata, highlighted in his keynote speech at the GDC 2006 [15] that there is a cultural need for innovative entertainment; a need to revolutionize electronic gaming. The upcoming release of Nintendo's new game console, *Wii* [13], shows an attempt to satisfy the need for innovative gameplay and enhanced immersive gaming experiences. We believe that in the near future, haptic devices will become more accessible to the average computer and console user and will play an important role in providing innovative forms of entertainment. This trend is heralded by the announced release of devices such as Novint's Falcon, which has an affordable target, even for mainstream consumers. Opportunities are arising for developers to invest in haptic applications and gaming titles.

As suggested by Chang [1], haptic technology will become an integral part of the game design process and require creative planning in order to take full advantage of this bi-directional modality. Gamer habits may change too in order to incorporate their sense of touch, which gives them more complex interaction with the game environment.

In Section 2 of this paper we discuss projects similar to this one. In Section 3 we present a description of our game, as well as haptic effects and rendering techniques used by our game. We also present some other methods of interaction which may be combined with haptic interfaces. The results of user trials are presented in Section 4, and possible future improvements to the project are presented in Section 5. This paper concludes in Section 6.

2 Related Work

The amount of literature regarding haptic technology and rendering has increased substantially in recent years. A description of our rendering technique is given in Section 3. For a more complete background on haptic rendering and haptics in general can be found in other articles ([4], [5], [6], [14]).

Haptic Battle Pong [2], a pong clone with haptic support for SensAble Phantom devices, is one of the few attempts at introducing modern haptics to gaming. Force-feedback is used to haptically display contact between a ball and a paddle. However, interaction with the game environment is limited since players can feel only the transient forces generated as the paddle strikes the ball.

There has also been some other work concerning the integration of haptics into a 3D game engine. Nilsson and Aamisepp [3] explain the relevance of incorporating haptics in a 3D engine and a plug-in for *Crystal Space* [22] was developed to demonstrate this integration successfully. However, haptic interaction in the context of 3D gaming was not well explored by this project.

Other efforts [7] to combine haptic and graphical rendering are ambitious, but don't contain features which are desirable for 3D game development.

Our approach builds on the work discussed in this section. By using existing, well-developed game engine components—specifically, a scene graph library and physics engine – and augmenting them with haptic rendering, we create a highly useful haptic game development environment which we use to experiment with haptic interaction in 3D games—of which a by-product is an actual game. The algorithms used for haptic rendering are simple, fast, and use the capabilities of existing software components.

3 HaptiCast

Figure 1: A screenshot from HaptiCast

HaptiCast is a multi-player 3D game which places the players in a first-person shooter (FPS) death match setting. It is designed to provide fast-paced action and a high level of interactivity. A screenshot of the game is shown in Figure 1.

An important component of HaptiCast is the physics engine, which simulates Newtonian physics for all

objects in the virtual game environment. Variables such as mass, velocity, friction, and external forces all contribute to the realism of the game. This component provides collision detection and response amongst in-game objects, allowing character and object control using physical dynamics, as well as access to information used by haptic rendering algorithms. The physics engine we've chosen for the HaptiCast project is *Newton Game Dynamics* [9], which is a small, fast rigid body physics engine for development in C/C++. Every object in our game world has a physical representation, and therefore is capable of exhibiting realistic physical behaviour.

The scene graph library we've chosen for the HaptiCast project is *Irrlicht* [8], which is a fast, cross-platform 3D graphic engine that includes features such as Gourad shading, z-buffering, dynamic lighting, mesh loaders, particle systems, texturing, and many more. This component is responsible for displaying 3D objects on the screen, as well as a graphical user interface.

The supported platform for HaptiCast is Windows 2000/XP using a Phantom Omni or Desktop [21] device from SensAble Technologies. For good haptic and graphical rendering to occur, a Pentium 4 processor and hardware 3D graphics processor is recommended. Due to haptic rendering and synchronization constraints, players must currently use the same machine in order to compete against each other.

3.1 Haptic Wands

In our game, the player interacts with the game world using a series of wands. When the player uses a wand, a spell is cast which displays a haptic effect and offers a different way of interacting with the game environment. Figure 2 shows a selection of wands whose haptic effects are discussed in the next section.

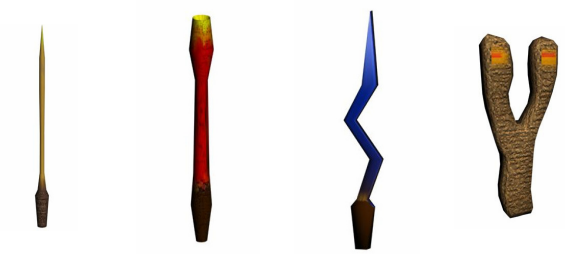

Figure 2: The haptic wands (from left to right): lift/swing, blast, bolt, lob

The haptic rendering of each wand is synchronized with the time-step updates of the physics engine. That is, force values at the haptic device are calculated and displayed each time the physics engine is updated. Since the physics engine simulation controls the position and orientation of all objects in the game, there should be no discontinuity between what the user sees and what the user feels while interacting with the game world.

3.1.1 Lift and Swing wand

The lift wand uses the most complex haptic rendering of all the wands. This wand, named because its intended function is to allow the player to pickup and lift objects in the environment, display the weight, momentum, and contact forces to the player. It allows any object in the environment to become a haptic probe, with smaller objects able to display finer details about the environment. Players may also manipulate objects so as to block oncoming projectiles or crush their opponents. The teaser for this paper shows the lift wand in action, as a player controls a haptic device in order to suspend a table over a gap in the floor.

A player selects an object by moving close to it, pointing at it with the wand, and holding the button on the haptic device. The player can immediately feel the haptic feedback of forces affecting the object.

The rendering algorithm used by this wand is similar to existing techniques [5] which use a penalty-based rendering technique. A distance vector from the center of the selected object to the virtual position of the haptic interface point (HIP) is used to calculate a spring force which is then displayed at the haptic interface. The equation used to generate the force displayed at the haptic device is calculated as:

$$\mathbf{F}_{\mathbf{haptic}} = \mathbf{Q}_{\mathbf{haptic}} \times \mathbf{Q}_{\mathbf{player}} \times (\mathbf{P}_{\mathbf{HIP}} - \mathbf{P}_{\mathbf{obj}}) \cdot \mathbf{k}$$

where F_{haptic} is the force to display at the haptic device, Q_{player} is the transformation of a point in the global frame to player frame, Q_{haptic} is the transformation of a point in the player frame to haptic device frame, P_{obj} is the current position of the selected object, and k is a scalar which controls stiffness. P_{HIP}, which is the haptic interface's position in the virtual game world, is calculated as:

$$\mathbf{P}_{\mathbf{HIP}} = \mathbf{P}_{\mathbf{player}} + \mathbf{Q}_{\mathbf{player}}^{-1}[\mathbf{P}^{\mathbf{0}}_{\mathbf{obj}} + \mathbf{Q}_{\mathbf{haptic}}^{-1}(\mathbf{P}^{\mathbf{0}}_{\mathbf{haptic}} - \mathbf{P}_{\mathbf{haptic}})]$$

where P_{player} is the current position of the player, P^0_{obj} is the starting position of the object (when the object is picked up), P_{haptic} is the current position of the haptic device, and P^0_{haptic} is the starting position of the haptic device. Using this rendering equation, haptic forces are always rendered from the player's viewpoint, which we find is more intuitive for the player.

As indicated by the rendering equation, the first step of the algorithm is to calculate the position of the HIP relative to the position of the selected object. Initially, these two positions are the same, but as the player manipulates the haptic device the object tries to "keep up". A force is rendered at the haptic device which is directly proportional to the vector (times the stiffness scalar) between the selected object and the position of the HIP. A scaled version of this force is also applied to the object so that it moves towards the HIP's position in the game world. Imagine that the player is dragging the object through space using a virtual spring, with one end attached to the HIP and the other to the selected object. A free-body diagram explaining this interaction is shown in Figure 3.

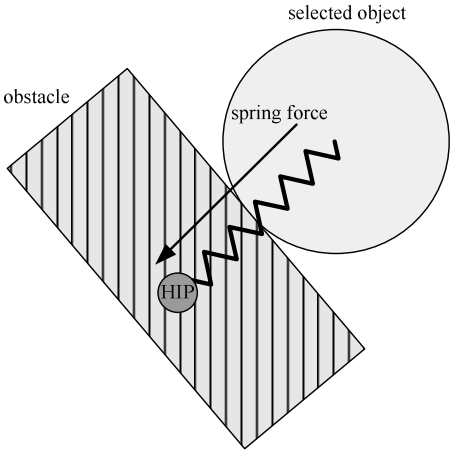

Figure 3: An illustration of the lift wand rendering algorithm.

In the figure, the HIP's position has penetrated a static obstacle. Since the selected object cannot move to the HIP's position, a spring force is displayed at haptic device and the user can feel a collision response.

A benefit of using a spring force (based on Hooke's spring law) to move and direct the selected object is that the physics engine should keep objects from penetrating by performing collision detection and response. Instability only becomes an issue if the spring force becomes unusually large, or if the simulation time step is too large. Special precautions need to be taken so that such cases never arise. The algorithm doesn't directly alter the position of the object since this would override any collision detection and response performed by the physics engine.

Another benefit of using forces to move and direct the selected object is that the user feels "drag" when they move heavier objects, which is a direct result of Newton's second law. While this feature is not wholly physically accurate, it does provide interesting feedback to the user.

If the physics engine is also capable of simulating frictional forces between objects in contact, these forces are indirectly displayed to the user. "Sticking" occurs when the selected object is in contact with another object and the HIP is moved so as to slide the selected object along its surface. Very high frictional coefficient values result in the object rolling across the surface, whereas low values result in no observable sticking or frictional force.

The player also has the option of enabling display of other forces at the haptic device. Scaled versions of gravity and impulse forces due to collision that affect the selected object may also be included in the rendering equation. This allows the user to feel the weight of the object as well as the impact of the object hitting an obstacle. These forces are calculated for us by the physics engine.

The swing wand is similar to the lift wand, but some of the forces generated during haptic rendering also affect the player's in-game character. The intended function of this wand is to allow the player to latch onto static objects in the environment and swing, or be suspended in midair, by using the spring forces to direct their movement.

3.1.2 Bolt and Blast wand

These wands do not make use of sophisticated haptic rendering, but display novel haptic effects to the user. A recoil effect is felt by the user as projectiles are fired out the end of the wand. With the blast wand, the user can feel the haptic force ramp up over time. The magnitude of the force is proportional to the energy and speed with which a fiery projectile is fired from the wand. The bolt wand displays small, transient forces to the user as energy bolts are rapidly fired from the wand. Users may take advantage of the 6 DOF input of the haptic device in aiming their wand.

3.1.3 Lob wand

This wand resembles a slingshot, which gives the user an indication as to its functionality. The lob wand allows the user to throw a grenade-like projectile at an enemy. The force displayed at the haptic device is directly proportional to the force with which the projectile will be flung from the wand. The user estimates the distance to their target and pulls the slingshot to the appropriate tightness.

3.2 Gesture Recognition

Gesture recognition allows a computer to recognize human gestures using a mathematical algorithm. This is a non-haptic feature which we are currently implementing for HaptiCast. We feel gesture recognition is a necessary progression which will allow game developers to take full advantage of the high degree-of-freedom input capabilities of modern haptic devices.

The orientation and workspace of the Phantom series of haptic devices allow the user to make natural, human gestures using a stylus. The idea of waving a haptic stylus through the air in order to cast spells is appealing in that it makes the player feel as if they really are a wizard. This is a feature-in-progress, but current results look promising. Simple 2D shape recognition is done using a neural network-based recognition engine, using an approach similar to others ([10], [11]), whom also provide good introductions to neural networks.

The workspace of the haptic device is separated into discrete regions which represent states. As players manipulate their wands, the neural recognition engine is fed a sliding window of the most current state transitions. The engine is trained to recognize which sequences of state transitions are meaningful within the context of the game, and which are not. Figure 4 shows the high-level design of the recognition engine, which uses a time-delay neural network (TDNN) to perform shape recognition. Some of the gestures we have trained the network to recognize using a haptic device as input include clockwise, counter-clockwise circles and diagonal strokes.

For test trials, the TDNN uses 16 input nodes, a hidden layer with 24 nodes, and 4 output nodes. A total of 25 states were used for the haptic workspace—a 5 by 5 grid in the x-y plane. The z-axis position value of the device was disregarded. A sliding window of the 8 most recent state transitions was also used. During experiments, the average calculation time for a single recognition was less than 10 ms. A large part of completing this feature will be to generate a training set which will enable the TDNN to accurately identify a pre-defined set of gestures.

3.3 Speech Recognition

An early prototype of the game included a speech recognition feature, and by using a microphone players were able to trigger wand functionality by saying a spell command phrase. For example, the player could say the phrase “fire blast” and a fire ball would shoot from the end of his wand. The speech recognition feature increased the level of interactivity with the game, certainly made the player feel more like a wizard casting spells, but was prone to several problems. Because of these problems, this feature was removed.

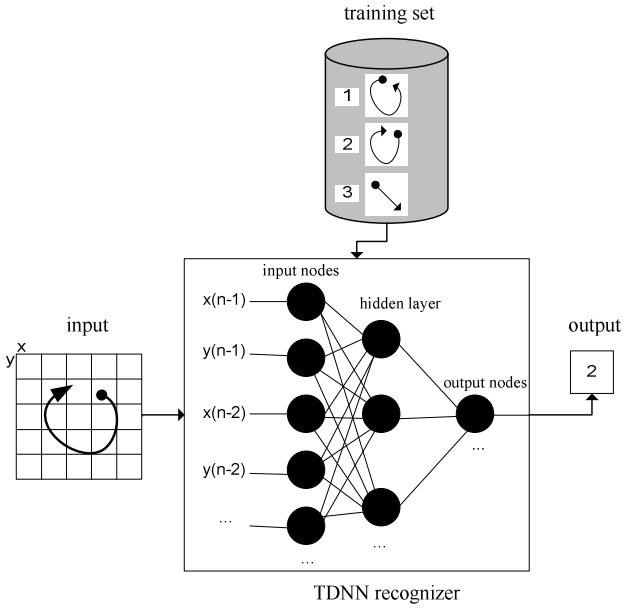

Figure 4: The gesture recognition engine

False positive recognition was one problem which contributed to the eventual removal of the speech recognition feature. This resulted in the incorrect spell being cast by the wand (initially command words were used to select the spell for a single wand), or a spell being cast when the user had spoken no command.

Delays caused during the speech recognition process resulted in awkward user interaction with the game environment. Speech recognition times would vary, but usually resulted in 1 or 2 second delay before a wand’s functionality was activated. This was deemed unacceptable. Some speech recognition engines do allow a trade-off of speed vs. accuracy. However this created more false positive/false negative recognitions.

Perhaps the biggest reason for removal of the speech recognition feature is that during test trials (see Section 4), many users felt uncomfortable speaking aloud the command words which cast spells.

4 Game Description and Results

After using HaptiCast, there is no doubt that the integration of haptics into the game increases its realism, the level of immersion felt by the user, as well as the entertainment value of the game.

Each player is given a set of wands, as described in Section 3, which they may use to interact with the game environment and to battle their opponent. Players start with a certain amount of health which may decrease if the player is hit by a wand projectile (e.g., from the bolt and blast wand), is hit by an object, or falls from a high enough height. A player may increase their health by running into power-ups scattered throughout the level. When a player's health reaches zero, they die and are resurrected at a starting point.

Each wand, too, is given a certain amount of energy which is used to supply power to the wand. When a wand's energy reaches zero, the player may no longer use that wand until its energy is replenished. A wand's energy may be replenished by running into the appropriate power-up.

An early version of the game was showcased during a course project presentation and at the University of Ottawa Engineering open house. At each event students, alumni, faculty and their families provided us with feedback about the game. Some of them were eager to play our game and we got very useful feedback from their experiences.

Adult players showed a tendency to be reluctant to say out loud the words to cast a spell, which lowered the player's motivation to fully immerse themselves into the role of a wizard, as well as defeating the purpose of integrating speech recognition into the game. The quality of speech recognition also varied, depending on the user's pitch, speed, and accent while speaking. This influenced us to change our game design and re-evaluate the speech recognition feature (also discussed in Section 3).

Children seemed to be more haphazard in exploring the game and made creative use of the magic spells. However, they had a larger learning curve and needed instruction while handling the haptic device. Adults supervised the session and children were instructed to firmly grasp the wand while lifting heavy objects. Both children and adults expressed awkwardness when initially using the haptic device, but most became more agile in their use of the Phantom stylus once they had some practice.

The recommended haptic update rate of 1000 Hz [5] was not achieved in trials of HaptiCast. However, frame rates of more than 60 fps (frames per second) were common. Though high update rates were not possible for the haptic rendering, there were no observable discontinuities or instability.

5 Future Work

A next step for the HaptiCast playground is to integrate haptic texture rendering into the game world. Bump maps have been shown as a suitable method to render tactile surface features of objects in a haptic virtual environment world [20]. The benefit of using a bump map or height field is that many 3D graphic drivers also support rendering of these textures.

Other approaches for haptic texturing use stochastic statistically-based models [17], spectral analysis [18], and virtual springs to render the roughness of a surface [19]. The suitability of each method for use in 3D video game needs investigation.

Another intended feature is the integration of network play into the game. This would allow multiple players to play HaptiCast using a LAN or Internet configuration. However, due to the highly interactive nature of the game, special consideration needs to be taken in order for stable, high-quality haptic rendering to occur. As previous work indicates [16], network latency is a source of instability and discontinuity in haptic rendering across a network. The physical simulation of the game world, too, must be synchronized across all network nodes.

6 Conclusion

From the test trials and user feedback we've received, HaptiCast promises to be a suitable test bed for experimenting with haptic interaction and effects in 3D games. The wands described in Section 3 contribute significantly to the immersion and entertainment value of the game, yet other methods of interaction using haptic interfaces are to be explored. Our contribution will be to develop these interaction techniques and to report on them to the game development and haptic communities. For this purpose, an open source project has been created for HaptiCast and may be found online [23].

Acknowledgements

We wish to give special thanks to the University of Ottawa and NSERC for providing equipment and funds to pursue our research. We also thank Dr. A. El Saddik for the opportunity to begin the HaptiCast project as part of his course curriculum. Finally, we thank SourceForge.net for hosting our project online.

References

[1] Chang, D., *Haptics: Gaming's New Sensation*, Computer, Volume 35, Issue 8, pp.84 – 86, 2002.

[2] Morris, D., Neel, J., and Salisbury, K., *Haptic Battle Pong: High-Degree-of-Freedom Haptics in a Multiplayer Gaming Environment*. Experimental Gameplay Workshop, GDC 2004.

[3] Nilsson, D., and Aamisepp, H., *Haptic hardware support in a 3D game engine*. Master thesis, Department of Computer Science, Lund University, May 2003.

[4] Srinivasan, M., and Basdogan, C., *Haptics in Virtual Environments: Taxonomy, Research Status, and Challenges*. Computer and Graphics. 21(4), pp. 393-404, 1997.

[5] Basdogan, C., *Haptic rendering tutorial*. Available at: http://network.ku.edu.tr/~cbasdogan/Tutorials/haptic_tutorial.html

[6] Ho, C.H., Basdogan, C., and Srinivasan, M. A. *Efficient point-based rendering techniques for haptic display of virtual objects*. Presence 8, 5, pp.477-491, 1999.

[7] Conti F., Barbagli F., Morris D., Sewell C., *CHAI: An Open-Source Library for the Rapid Development of Haptic Scenes.* Demo paper presented at IEEE World Haptics, Pisa, Italy. March 2005.

[8] Irrlicht Engine. http://irrlicht.sourceforge.net/

[9] Newton Dynamics game engine. http://www.newtondynamics.com/

[10] Boukreev, K., *Mouse gestures recognition*. Code Project article. Available at: http://www.codeproject.com/cpp/gestureapp.asp

[11] Murakami, K., and Taguchi, H. *Gesture Recognition using Recurrent Neural Networks.* Proceedings: Conference on Human Factors in Computing Systems, pp. 237 – 242, 1991.

[12] Novint Technologies. http://www.novint.com/

[13] Nintendo Wii. http://wii.nintendo.com/

[14] W. Mark, S. Randolph, M. Finch, J. V. Verth, and R. M. Taylor II. *Adding Force Feedback to Graphics Systems: Issues and Solutions*. In SIGGRAPH'96, pp. 447-452, August 1996.

[15] Carless, S., *Detailed Nintendo Keynote Coverage*. Gamasutra article. Available at: http://www.gamasutra.com/php-bin/news_index.php?story=8656

[16] Fukuda, I., Soju, M., Iijima, M., Hikichi, K., Morino, H., Sezaki, K., and Yasuda, Y., *A Robust System for Haptic Collaboration over the Network*, from "Touch In Virtual Environments", Prentice Hall, 2002.

[17] Siira, J., and Pai, D., *Haptic Texturing – A Stochastic Approach*, Proc. International Conference on Robotics and Automation, pp. 557-562, 1996.

[18] Wall, S.A., and Harwin, W.S., *Modelling Of Surface Identifying Characteristics Using Fourier Series*, DSC - Vol.67, Proc. ASME Dynamic Systems and Control Division (Symposium on Haptic Interfaces for Virtual Environments and Teleoperators), pp. 65-71, 1999.

[19] Minsky, M., Ming, O., Steele, O., Brooks, F.P., and Behensky, M., *Feeling and seeing: issues in force display*, Proc. of the 1990 Symposium on Interactive 3D Graphics, pp. 235-241, 1990.

[20] Theoktisto, V., Fairen, M., Navazo, I., Monclus, E., *Rendering detailed haptic textures*, Workshop on Virtual Reality Interaction and Physical Simulation, 2005.

[21] SensAble Technologies. http://www.sensable.com/

[22] Crystal Space 3D. http://www.crystalspace3d.org/

[23] HaptiCast. http://sourceforge.net/projects/hapticast