PORTAL; MERGING THE REAL AND VIRTUAL GAMING WORLDS

by

Ashley Gavin

A senior thesis submitted to the faculty of

Bryn Mawr College

in partial fulfillment of the requirements for the degree of

Bachelor of Arts

Department of Computer Science

Bryn Mawr College

May 2010

Copyright © 2010 Ashley Gavin

All Rights Reserved

BRYN MAWR COLLEGE

DEPARTMENT APPROVAL

of a senior thesis submitted by

Ashley Gavin

This thesis has been reviewed by the research advisor, research coordinator, and department chair and has been found to be satisfactory.

Date	Dr. Dianna Xu, Advisor
Date	Dr. Deepak Kumar, Research Coordinator
Date	Dr. Douglas Blank, Chair

ABSTRACT

PORTAL; MERGING THE REAL AND VIRTUAL GAMING WORLDS

Video games first became popular on Personal Computers and in giant Arcade boxes; however, much has changed since the video game industry's inception. Games like Guitar Hero and Rock Band, and systems like the Nintendo Wii and future systems like the Microsoft Natal, are further proving that the ultimate gaming platform, one that makes games accessible to much wider audiences, is one that allows users to feel fully immersed in game environments.

Merging the real and virtual gaming worlds is not an easy task, and is an area that has until recently been largely unexplored. However, as evidenced by recent trends and new research, immersive gaming can be characterized and achieved by the inclusion of three essential interface components. Firstly, games should allow the user's gestures and movements to manipulate their gaming environment. Secondly, the game's environment should extend beyond the TV screen and into the user's physical world. Lastly, the game must allow for objects to exist in both the real and digital worlds seamlessly. The Portal game system attempts to create such a gaming environment through the implementation of head-tracking, object-detecting, and gestural interfaces, harnessed through the power of the Nintendo Wiimote.

ACKNOWLEDGMENTS

First I must acknowledge those who have also conducted research in this area. Thank you to Brian Peek for the creation of the Wiimote library and for answering my questions. Thank you to Johnny Lee for his work on Wiimote based interfaces and his freely available code. Without their research and assistance this thesis would not have been possible.

I must, of course, thank my fellow Computer Science majors for their friendship, constant support, and shared love of learning. You have seen me at my best, my worst, and in my pajamas (often). Thank you.

Thank you to Jennifer Garrett. Your love and support throughout my college career has been instrumental in both my educational and personal development. I cannot thank you enough for helping me with my work, including this thesis, and through my personal struggles throughout my time here. Bryn Mawr would not have been Bryn Mawr without you.

Thank you to my mother and my sister. At the very least, you have made this education possible for me. You have been proud of everything I have done, and always believed in me. Thank you for giving me the confidence to believe in myself.

Most of all I must thank my professors, Dianna, Doug, and Deepak. If you havent noticed, I love Computer Science, and your presence in my life as educators and mentors has fueled that passion. The Computer Science department has felt like a family to me, and I am so grateful for my experience here and what you have allowed me to achieve. Bryn Mawr students are so lucky to have you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

The video game industry has changed drastically since its inception, and in recent years, the industry has completely reinvented the video game experience. Although many visualize keyboards or game pads as being integral to gameplay, trends in recent games and consoles are moving away from relying upon these apparatuses alone. Recently there has been an explosion in the popularity of music-based games, with non-traditional interfaces, that allow the user to feel as if they are playing real instruments[1]. These games include the highly popular games Rock Band, Guitar Hero, and Dance Dance Revolution. The Nintendo Wii represents another major milestone in video game interface development. Unlike previous consoles, the Wii capitlized on an interface system that allowed complete gestural and movement-based on input from the user. Though the Wii lacked the computing power of its competitors, the Sony Playstation 3 and the Microsoft Xbox 360, it still was able to dominate the market place, becoming the best selling of the three. The Wii's success largely relies upon its gestural interface, which has appealed to a far wider audience (e.g. young children, women in general, and the elderly) than had previously taken interest in video games [2] Upon noticing the Wii's success, Sony and Microsoft have begun developing their own gesture-based gaming consoles [15][16][6].

If there has been one significant trend in video game development over the past 10 years, it is the push to develop games that allow users to feel fully immersed in the worlds in which they play [3]. Players want to feel as if they are actually driving racecars, playing guitars, and swinging baseball bats. They do not simply want to observe the fictional worlds in which their games take place, they want to be a part of them; Portal, an immersive gaming platform, provides the opportunity for them to do so. This thesis will serve as an introduction to the Portal game environment, how it works, and the motivation behind its creation.

1.2 Immersive Gaming

Portal is an immersive, 3D, interactive gaming environment that intends to redefine the first-person gaming experience. Thus far we have identified that user immersion is key in developing entertaining and accessible games. However, we have not yet set specific criteria for what it means for a game to be immersive, how developers can go about creating such games, and how the Portal system meets these criteria. In order to fully understand what Portal is, and how it works, we must first understand techniques that immersive gaming systems employ.

Innovative trends in video game design have been abundant in recent years, and while they take shape in different forms (e.g. the guitar controller from Guitar Hero, or the gesture-based Wiimote,) they all have one essential theme in common: they are interface devices specifically created to immerse the user into a gaming environment. Games themselves have not changed too drastically over the years, tending to fall into one of several basic genres such as simulation or first-person shooter[4]. Rather, it is the way games receive input that has changed, and these changes have proven to be incredibly successful in targeting increasingly wider audiences [2]. This prompts an investigation of the different ways interface devices can seamlessly integrate the user into the gaming world, and how these tactics can be expanded upon. Generally, tactics in creating immersive gaming environments can be divided into three areas: gesture and movement-based input, the transference of digital objects into the real world, and the transference of real objects into the digital world.

1.2.1 Gesture and Movement

Generally speaking, gesture and movement-based interfaces allow the user to give input to the console or computer through the use of their body, typically their hands. The Nintendo Wiimote is perhaps the most famous example of a gestural and movement-based interface. The Wiimote, later explained in more detail, allows for users to interact with the gaming environment using gestures like shaking, waving, throwing, dropping, and drawing. Particularly important to the Wiimote's design is that it is a vague device, in that it can function as one of many props. For example, a Wiimote can function both as a golf club or a sword. Gestural and movement-based interfaces can also be implemented using a camera. The Sony EyeToy [5] uses computer vision to interpret player movements and gestures without using a physical device. The Microsoft Natal and Sony Move are rumored to use a camera for this purpose as well [9][15][16]. Lastly, head-tracking systems, though currently not a large area of research in video game development, could also be seen as a movement based interface, allowing the user's head movements to manipulate their gaming environment. For example, moving towards the screen would move the user's character through the game's world, turning one's head to the right would cause the perspective in the game to shift to the right. All of these techniques are considered immersive in that they are more natural to the way humans interact with objects in the real world.

Swinging a Wiimote in the fashion of a golf club feels more accurate to real life golf than using a typical gamepad. Turning one's head to view a distant object is more intuitive and natural than moving a joystick to achieve the same goal.

1.2.2 Digital Objects in the Real World

For a fully immersive experience, players must feel as if they exist inside the world in which their game takes place. This includes the process of taking objects that exist in the digital world (e.g. terrain, enemies, etc.) and attempting to make them feel" as if they are real world objects. In older games, and still common today, transferring digital objects into the world is often implemented as feedback to the user [4]. For example, slippery and sloped terrain that need be ascended may require more pressure on the joystick from the user than flat, non-slippery terrain. The Nintendo Rumble Pak, created in 1997, literally shook player's controllers when appropriate (e.g. falling from a great distance, being punched etc.) [18]. Today methods of incorporating digital objects into the real world are more nuanced. Wiimotes, for example, come equipped with a speaker that emits sound specific to an object or situation (e.g. the clang of a sword making contact). The Wiimote itself could be considered a manifestation of digital objects, serving as a real world representation of game objects. The Sony Move, which comes with a control system very similar to the Wii, recently released a demonstration that excels in the transference of digital objects into the real world. The demonstration uses two Sony Move Wands to simulate the act of shooting an arrow. As the user pulls the "string" further back, aiming becomes increasingly difficult, simulating the feeling of tension in a cross bow. While these methods are effective and innovative, there other means of transferring digital objects into the real world that Portal employs in addition to methods such as those listed above.

1.2.3 Real Objects in the Digital World

Most games and gaming systems currently do not attempt to merge real world objects into the digital worlds of video games. The closest example that currently-available games and systems have implemented takes shape in prop-based games such as Guitar Hero and Rock Band. Though these interfaces are restrictive in that they can only be used to play one particular kind of video game, they have proven to be extremely entertaining and appeal to a wide audience because of their immersive features[1]. In addition to using a psuedo-gestural interface, the Rock Band and Guitar Hero interfaces are popular because they utilize real props that exist in both the real and digital worlds. Though the props themselves exist in the physical world, the characters of the gaming world use these same devices (e.g. if one has chosen to play the drums, their game character will be a drummer). In these games' most recent versions, one can even see that the game's characters accurately represent the player's movements (e.g. which drum the player hit, whether the player missed a guitar strum). This kind of interaction, made possible by a real world prop, gives the user a sense that they exist within the digital world, even though they are using a real world prop.

The Microsoft Natal, through the use of an infrared projector and CMOS sensor [6], has the ability to sense real world objects, as does the Sony Eye. However, as based off recent demonstrations, it seems neither company makes an effort to include props as a fundamental point of gameplay. Rather, both systems seem to avoid these apparatuses as much as possible. We believe that physical device that accurately reflect objects in the real world serve as important sources of immersive gameplay, as well as highly important means of feedback[4], and have thus have made them easy to implement in the Portal gaming system.

1.3 What Is Portal

As stated above, Portal is meant to be a fully immersive gaming experience, and thus attempts to make as many of the above features as possible available to game developers. Some of the included features are head-tracking, object detection (using infrared tags), player-tracking (as based on the head-tracking system), and gestural interfaces. Portal is particularly novel in that it allows for all these features to be utilized at once, whereas most gaming systems only account for a few of these features. We believe that in order for a gaming system to be truly immersive, these features must be used in conjunction with one another, erasing the boundaries between the player's world and the world of the game. Though Portal will be developed for and run on a PC, it will rely heavily on the Nintendo Wiimote, the Wiimote libraries written by Brian Peek, (an independent developer and winner of the Microsoft C# MVP award) [14][11], and Wiimote interface research conducted by Johnny Lee, a CMU PhD graduate in the field of HCI and current researcher at Microsoft Research. The Wiimote's infrared camera, explained in detail later in this thesis, is almost entirely responsible for allowing the immersive features that Portal attempts to capitlize upon.

Portal's capabilities cannot be fully expressed without being demonstrated through the implementation of a game. For this reason, a straight-forward but entertaining three dimensional version of Space Invaders has been specifically designed and implemented for the Portal system. This version of Space Invaders has three points of input from the user that make it distinctively different from the original. Firstly, the Portal version of Space Invaders is entirely first-person, and uses head movements to control the character's locomotive movements. Secondly, a Nintendo Wiimote takes shooting input from the user. Lastly, a real world prop is implemented as a kind of shield for the user. More information on this demonstration of Portal is available in the chapter on Space Invaders.

1.4 Motivation

Although graphics have become increasingly realistic, interfaces have changed, and general computing power has increased over the past 30 years, the same basic premises of video game development still hold true. Players have a desire for entertaining, balanced games, with characters in whom they can become invested [4][5]. The goal of this research is to not only create an environment in which the user can truly immerse themselves, but one that proves to be fun. Almost no research has been conducted in the areas of combining different kinds of immersive gaming interfaces such as head-tracking and gestural interfaces, and whether these kinds of combinations create an exciting video game interface. The purpose of Portal is to follow and expand upon these trends, combine them, test them, and refine them to create a first-person gaming experience that is both to the user, prompts the user to become fully invested in their games, and entertains the user. We believe that the combination of head-tracking, player-tracking, real-world object detection, and gestural interfaces will not only create an effective and entertaining video game environment, but is the future of truly immersive video game experiences.

Chapter 2

Related Works

Very little research has been conducted in the area of immersive gaming that combines all the areas of research that this thesis explores. For example, much has been researched in the way of intuitive interfaces such as head-tracking, and gestural interfaces, however little research has been conducted on head-tracking based games, or games that utilize both head-tracking and gestural interfaces. This thesis has compiled information primarily from four areas of research; gestural interfaces, head-tracking interfaces, Wiimote technology, and general principles of gameplay. Each of these areas of research, and the ways these areas interact with one another, was given great consideration during Portal's implementation.

2.1 Gestural Interfaces

A gestural interface is one that uses gestures, hand movements, etc. in order to take input from the user. As exemplified by the success of the Nintendo Wii, and by music-based games such as Rock Band and Guitar Hero (though these interfaces are not purely gestural), there is a lot to be said in favor of gestural interfaces. Having seen the success of the Wii, both Microsoft and Sony are moving toward incorporating gestural interfaces into their next video game console [6][15][16]. Gestural based interfaces can manifest themselves in many ways: through motion, as the Wii does using an accelerometer, through computer vision, or through haptic technology. Microsoft's Natal serves as a particularly interesting comparison to the Portal system. Like Portal, Natal attempts to eliminate the boundaries separating the digital and real gaming worlds through the use of an infrared projector and CMOS sensor. The projector and CMOS sensor are able to sense the player's location and movements, and even sense objects within the player's room such as furniture. However, the most important component of Natal is the ability to interpret gestural input [13].

Regardless of how they are implemented, most find that users are not only satisfied with gestural interfaces, but prefer them to more traditional interfaces such as the mouse, keyboard, and game pad [7] [1][2]. The use of a vague physical apparatus, such as the Wiimote, or no apparatus at all, also allows for developers to use gestural technologies in theoretically infinite ways. For example, Rock Band restricts users to singing, drum playing, or guitar playing alone. Conversely, the use of a device-less gestural interface would allow the user to play whichever instrument they want, or perhaps even invent their own [8][1].

Gestural interfaces, however, have their limitations. There are many actions, particularly locomotive actions, that cannot be translated to simply one gesture. Often times it is best to choose another method of implementing these actions, rather than relying on gestures alone [8]. By using both a gestural interface as well as a head-tracking interface, problems with locomotion related to gesture are eliminated in Portal.

2.2 Head-tracking Interfaces

Head-tracking has been generally studied outside the video game arena for quite sometime as it has applications in many areas of interface design. Though it has many different kinds of applications, the underlying principle behind head-tracking is the same. Based on the heads location and angle relative to the screen, the camera will make appropriate translation, rotation, and scalar transformations in order to represent the users point of view [9], such as in Figure 2.1.

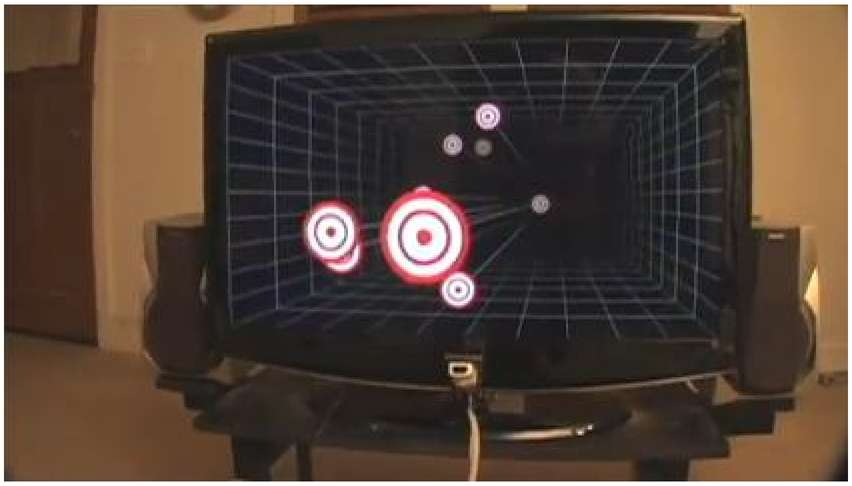

Figure 2.1 Johnny Lee's demo of head-tracking using the Wiimote

Though almost all head-tracking systems have this in common, there are several different ways of incorporating head-tracking into a video game environment. For example, head-tracking can be used to enhance presence in a game by literally superimposing the players face into a game[5]. Because gestural interfaces cannot fully capture the essence of locomotive movements [7], and because actually performing these locomotive movements can become tiring [2], head-tracking can be used as a substitute to actually performing the acts of running, jumping, ducking, etc. [10]. Head-tracking can also most obviously be used to create a virtual reality environment, allowing the user full control over their characters perspective [2]. Portal will implement both perspective control and locomotive control.

2.3 Wiimote Technology

Though there is no published information offered by Nintendo on the Wiimote, many developers and engineers have taken an interest in reverse engineering the Wiimote to learn how it works. Each Wiimote comes equipped with an infrared camera with a resolution of 1024 x 768 pixels. This camera's field of view is 45 degrees, which is important when considering making design choices as it physically limits areas of play. It has a high refresh rate of 100 Hz. This camera, in conjunction with infrared LEDs, makes the Wiimote a perfect apparatus for experimentation in head and player tracking. The Wiimotes most famous device is perhaps its accelerometer, which makes gestural interfaces possible. A vibration motor is also included in the Wiimote for tactile feedback, as well as a speaker for audio feedback [2].

Figure 2.2 Nintendo Wiimote

Because the Wiimote has Bluetooth capabilities, many developers have begun building libraries to work with the Wiimote. The most comprehensive, and the one we have chosen to use in the creation of Portal, is the Managed Library for Nintendos Wiimote, written in C# by Brian Peek. Tools necessary for supporting head-tracking, object tracking, gestural input, are already supported by the library [11].

2.4 General Gameplay Research

Though Portal is meant primarily to serve as a research project in the field of video game interface design, we believe that the most fundamental element of game design, and crucial to the success of any game or gaming platform, is whether or not the user has fun playing [4][3]. Because entertainment, immersive and balanced gameplay, and fun are the most important criteria in a gamers experience, we have considered these to be necessary elements of the Portal experience. These factors have been considered through Portals design and implementation process. At each step of Portals development, our Space Invaders implementation was tested for balance, ensuring that the game itself is both fair and challenging. It was critical that the game itself be entertaining, so that flaws of the game and flaws of the Portal system were not confused with one another, or misinterpreted.

Chapter 3

The Portal Demonstration

To exhibit the power and capabilities of the Portal gaming interface, we have chosen to implement a game using the Portal system. The game we have chosen is based on the classic game Space Invaders (also commonly known as Galaga). This demonstration is meant to utilize all the most fundamental parts of the Portal system (e.g. head-tracking, prop detection, and gestural control).

3.1 Space Invaders

The game we have chosen to implement is Space Invaders. It has been chosen for both its simplicity in rules and its timeless appeal. Developed in Japan in 1978 as an arcade game, Space Invaders still exists today in console form, PC form, and in flash games by numerous duplicators [12]. The rules are simple as the game takes place in a 2-Dimensional world. Aliens exist atop the screen, in many rows. Throughout the game, they continually move both across the screen and downward. The protagonist, whom the user controls, is only able to move on the X axis at the very bottom of the screen. There are several small areas of protection throughout the game's world that the protagonist may hide behind; however, they can be slowly destroyed by the aliens. The user attempts to shoot all the aliens before they either destroy the user's ship, or reach the bottom of the screen. See figure 3.1.

3.1.1 The Portal Implementation

Though the most basic rules of the original implementation of Space Invaders will be perserved, the Portal implementation will have some stark differences, mostly pertaining to interface differences and three dimensional versus two dimensional gaming environments. Additionally, the Portal implementation of Space Invaders takes on the first-person perspective neccessary for implementing head-tracking.

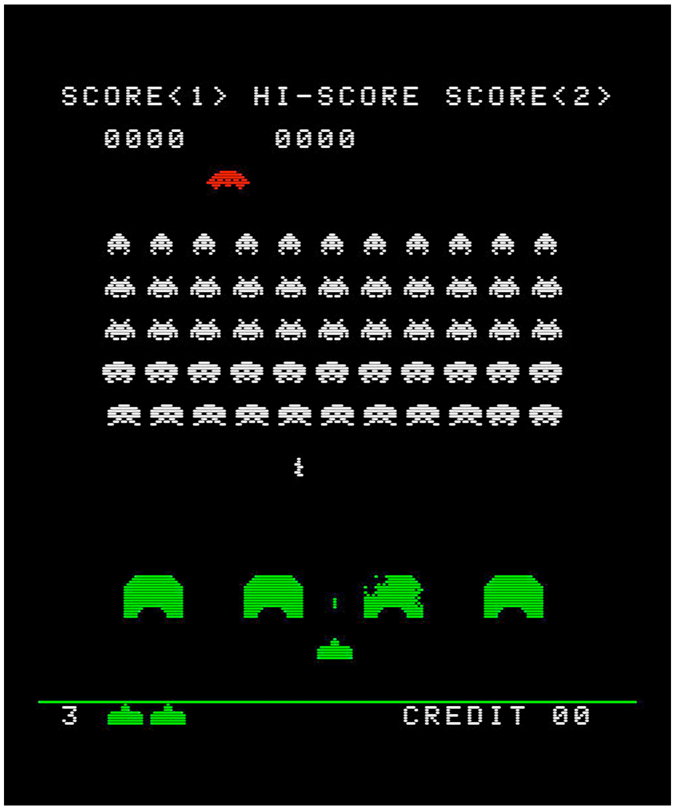

Figure 3.1 Space Invaders

Space Invaders in Three Dimensions

Because we are working in three dimensions, the aliens will move toward the user on the Z axis. However, they can also move, somewhat sporadically, on the X and Y axes.

The user has some freedom on the Z axis, restricting them from moving too far forward or backward, and nearly full freedom on the X and Y axes. Restriction on the X and Y axes exists only to ensure that the user and aliens remain within reasonable distance from one another. The user will control their movement on the Z axis by moving either toward or away from the screen. The user will control X axis movement by moving side to side, and control Y axis movement by moving up and down (e.g. ducking, jumping). Additionally, these movements will alter the perspective on the screen. For example, ducking down and moving left will cause the aliens to appear overhead and on the right side of the screen. Even if the user does not physically move their body, moving their head will be enough to affect their perspective. This implementation of head-tracking will create a simulated three dimensional environment, and give the user the feeling that they have penetrated through their television and now exist in the games world. However, this does not

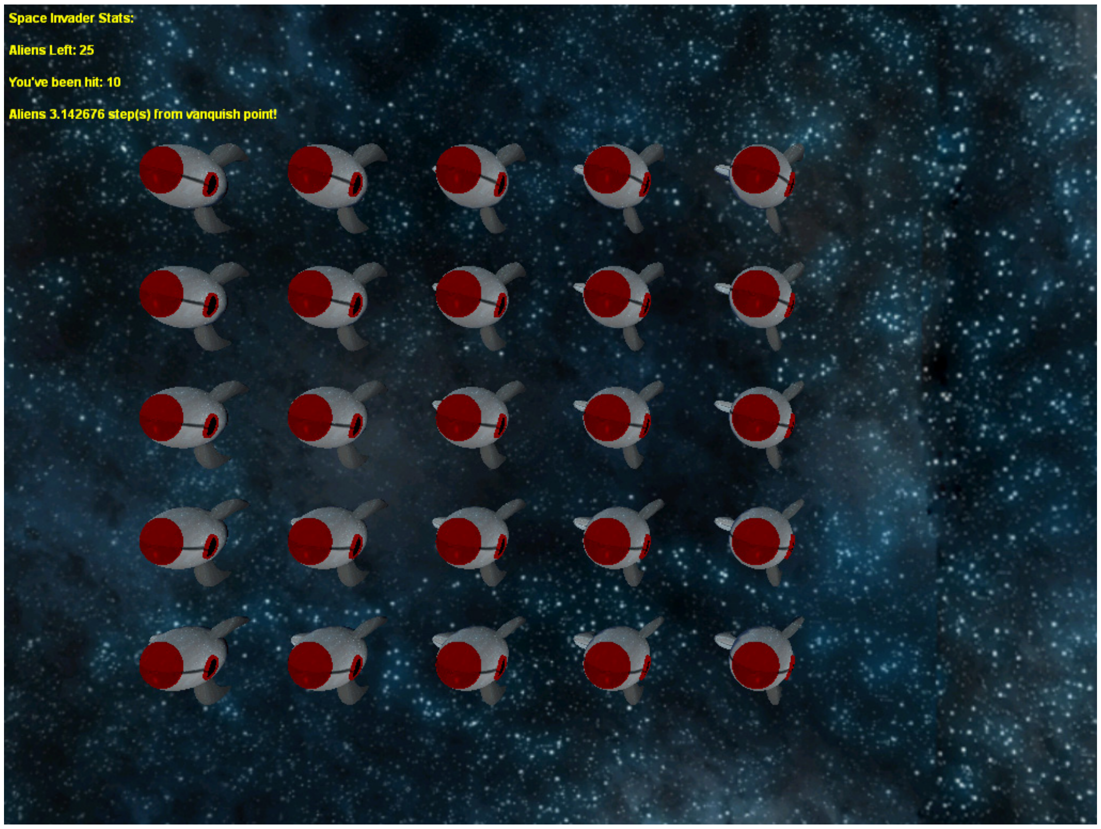

Figure 3.2 Space Invaders in 3D

help the user feel as if the game characters and environment have also become a part of their world, nor does it allow input for shooting the aliens. For this, props and a pseudo-gestural interface will be used.

Wiimote Input

Integral to Space Invaders is the ability to shoot aliens. The user will be given a Nintendo Wiimote, housed in a laser gun like apparatus as seen in figure 3.3, in order to shoot the aliens (the user will push a button to indicate that a shot has been fired). This device will be referred to as a laser gun for the remainder of this document. Though this device does not implement true gesture-based input, as it does not utilize the Wiimote's accelleromoter, we believe that it comes close enough to doing so, particularly when it is housed in its laser gun case. However, one could easily alter this demonstration to include truly gesture-based input through utilizing the Wiimote's accellerometer. In addition to simply shooting, the Wiimote will also give the Portal system data from which the game can display an on-screen cross-hair, in order to give the user feedback about where they are aiming. Without such an apparatus, an implementation that the Microsoft Natal might endorse, feedback is extremely limited. Not only does the cross-hair provide the user with the security of knowing their laser gun is working, but the sensation of pressing a button, combined with seeing a result on-screen, provides the kind immersive feedback that is common in real world interfaces [4].

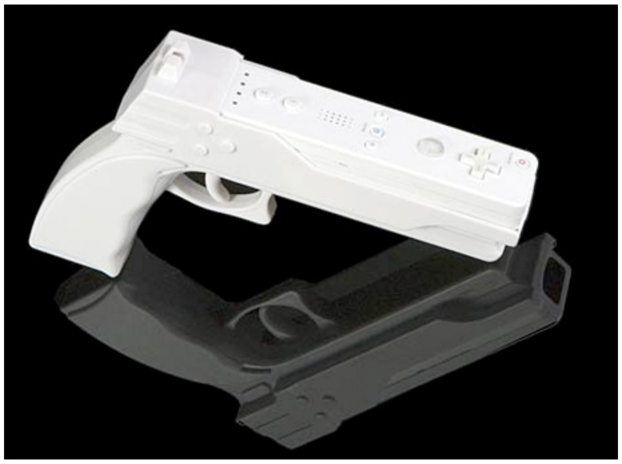

Figure 3.3 Wiimote in a laser gun casing

Props

In addition to a head-tracking device, and the laser gun, the user will also be given what is referred to as a blast shield. This shield will be a real world object that the user can hold while he or she plays. For this reason it will be lightweight and highly portable. The shield is recognized by the Portal system through a reflective infrared tag on its front, as pictured in figure 3.4. The blast shield was implemented specifically because of its ability to easily cross the border from real world object, to digital object. When the user has the shield held up, they will be immune to alien shots; however, they will also be unable to shoot. These choices were made in order to give the game a fair sense of balance. Aliens will be allowed to continue shooting, and remaining player bullets will continue to exist after the shield has been raised. This will allow users to take a shot and retreat as one possible strategy. For feedback purposes, the laser gun will vibrate and make a noise when an alien bullet has hit the user's shield. If the user is not using the shield, and is hit, a more striking vibration and sound will be used.

Artificial Intelligence and Player-tracking

One major difference between the original Space Invaders and the Portal implementation is Artificial Intelligence. In some versions of Space Invaders the aliens are able to target the protoganist based on his or her location. However, knowing the location of

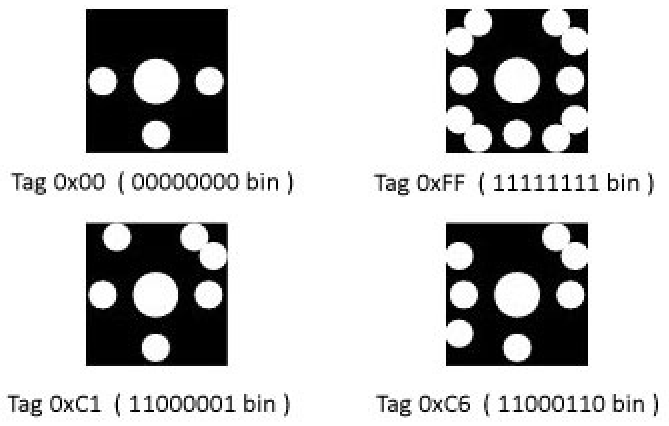
Tag 0x00 (00000000 bin)
Tag 0xFF (11111111 bin)
Tag 0xC1 (11000001 bin)
Tag 0xC6 (11000110 bin)

Figure 3.4 Infrared Tags

the user in the Portal version raises some difficult questions. Does the user really have a location, seeing as he or she does not turly exist in the same world as the aliens? Can we simulate such a location? If we do choose to simulate such a location, how much weight should be given to a precise, accurate representation of player's location? How much weight should be given to issues of balance and gameplay? We have chosen to take a simulated approach, focusing slightly more on balance and fairness than a precise simulation of the player's location. The details of this implementation are discussed in the Software Implementation chapter. Though this implementation may be slightly less realistic, it allows for the aliens to target the user, based on his or her real world location, in a fair and balanced way. We refer to this as player-tracking, and it is made possible through the head-tracking system. This approach attempts to transfer digital objects, the alien bullets, into the real world seamlessly.

Chapter 4

Hardware Implementation

The hardware for the game must be implemented in a fashion that supports all the games features, which can be broken into three components; head-tracking, Wiimote input, and prop detection.

4.1 Head-tracking

Using the Wii, head-tracking can be implemented rather simply. It relies on both the Wii sensor bar, to emit infrared light, and the Wiimote, to detect said light. Contrary to typical Wii gameplay, the Wiimote remains stationary under the player's screen. This is because head-tracking is more accurate when the camera is stationary [2]. Since the Wiimote requires infrared light in order to detect the player's head location, the Wii sensor bar could be used, adorned atop the user's head, as a source of infrared light. However, this design choice seems uncomfortable, and rather silly. Because any infrared light will do, the user can instead wear a pair of safety goggles (figure 4.1) that house infrared LEDs on the sides [2].

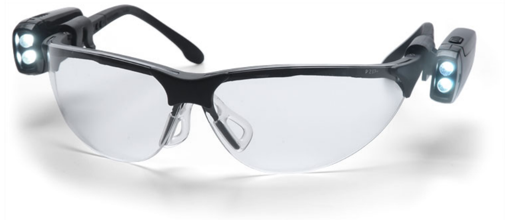

Figure 4.1 Infrared Emitting Goggles

4.2 Wiimote Input

As based on research stated above, users want to feel as if they are using the actual tools that their characters use. Guitar Hero and Rock Band are perfect examples of this phenomenon [1]. Giving the user a gamepad or keyboard to control their alien shooting device would seem not only unnatural generally, but extremely unnatural when paired with the head-tracking system. Thus it should be implemented as if it were an actual shooting device. Fortunately the Wiimote can easily be encased to appear as a laser gun. We believe that this implementation, as opposed to an apparatus free implementation like the Microsoft Natal endorses, will be more entertaining and natural to the user. A physical apparatus, similar to one that an alien destroyer might actually use, will feel more realistic and natural to the user.

We had to make a choice as to whether we implemented the laser gun using the Wiimote's accelerometer, or reyling upon sensed infrared light. Both have their flaws and benefits. The accelerometer, when used, can detect not only X and Y axis movements, but Z axis movements as well. The infrared method must simulate Z axis movement based on infrared light distance as implemented in the head-tracking system. However, the accelerometer is not nearly as accurate as the infrared method in detecting small movements. Because accuracy is most important in the laser gun, we chose to implement the latter method. Z axis changes will be interpreted by using the head-tracking input as a supplement. In other words, each time a shot is fired we will calculate the plane at which the users perspective exists. The trajectory of the shot, therefore, will be a line that passes through that plain and through a point that exists at the X and Y coordinates at which the laser gun points.

4.3 Props

Lastly, the blast shield must be implemented. Though its implementation is physically the most simple, a cardboard shield with a reflective infrared sticker, it is perhaps the most problematic of any of the elements. This is because the blast shield's infrared light will be aimed at the same Wiimote used for head-tracking. The system must be able to tell the shields infrared light from that of the head-tracking goggles. Fortunately, there are two methods that can solve this problem. The first relies on the fact that any one Wiimote can only detect four infrared lights at once, and thus must count and differentiate between said lights. Only two of these four are fired from the head-tracking goggles. If the Wiimote senses more than two, it can assume that the blast shield is up. However, in the event that the user is positioned in such a way that one or both of the infrared LEDs on their goggles are not visible, one cannot know whether that light is coming from the goggles or the blast shield . To avoid this problem, the infrared tag on the shield will reflect four points of light, maxing out the Wiimote's sensing abilities. For added security, the system will be able to recognize the shield's specific infrared tag. Tag recognition is already supported by

C# libraries. Thus, if for some reason less than two lights are being detected, a tag detecting function can be employed. Tag detection will only be necessary when less than two lights are displayed. For this reason, there is not a problem with tag recognition when both infrared goggle lights are visible.

4.4 A Generalized System

While this system seems targeted at our new version of Space Invaders, it is actually highly flexible and can be used for most any game. It is extremely easy to remove one part of the system by simply removing a Wiimote or sensor bar. Moreover, the head-tracking system, tag detection system, and use of the Wiimote as a controller are generalized, and easy to change based on each game's unique needs. By only eliminating a few lines of code, head-tracking can be constricted to only one or two axes. As evidenced by the success of the Wiimote, Wiimote input can also be generalized. Multiple props can be implemented by using multiple, unique tags. The beauty of the Portal system is that it can support any game of any genre. The success of a game is only limited to the imagination of the designer; This is exactly how a console should be implemented.

4.5 Hardware Concerns

Though this system is extremely rewarding and entertaining, it has a number of shortcomings. In games that use a joystick, game pad, or keyboard, unexpected input is limited to a set of buttons or characters. The Portal system; however, must take into account that input to the system can be manipulated in much more strange, unexpected ways. The system must be able to deal with users who turn their back to the camera, bend over to pick something up, or momentarily turn to speak to a friend beside them.

The system must also implicitly know when a user has walked away from the game. If a user has to leave momentarily, the game cannot and should not go on playing without them. This is a security issue as well; cheating can easily occur by stepping out of the Wiimotes field of view. Cheating can also occur by manipulating infrared light. One might place a finger over an LED to instigate some kind of glitch. The same might be done by adding an infrared light to a prop. Issues concerning security must be addressed in the Portal system.

Gameplay concerns do not end at issues of security. Head-tracking poses a controversial problem involving locomotion. Characters in first-person shooters and adventure games are most often found running. Obviously one cannot simulate running using a head-tracking system in the most natural way; the user would either run out of range or run into the TV. Rather, some kind of hot spot system (figure 4.2) must be implemented. That is, standing at certain distance from the TV implies moving forward, another implies moving backward, another side to side, etc. Though this is easy to implement, it fails to acknowledge actions a character should be capable of, such as turning around. Though this is not an issue in Space Invaders, it is one that must be addressed by future game designers.

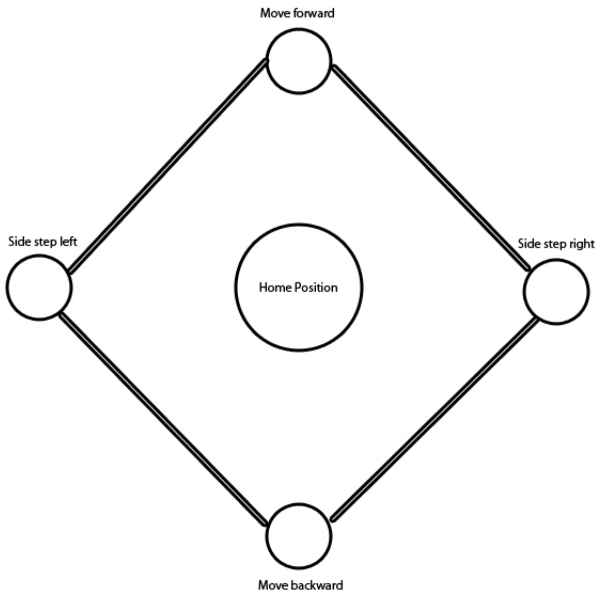

Figure 4.2 Hot Spot System Diagram

Another problem with locomotion lies in the fact that most gamers want to sit on a sofa when they play; a hot spot system would require too much effort [2]. This can be accounted for by using a hot spot system with an adjusted sensitivity level. The user can calibrate the system at the beginning of each game, indicating how far in each direction they'd be willing to lean. This kind of calibration is implemented to some degree in Space Invaders in the beginning of each game when the user's distance and orientation to the screen is detected and accounted for in determining their central location.

Chapter 5

Software Implementation

This section primarily discusses the implementations of the Portal system, as well as implementations of Space Invaders from a software perspective. The hardware is implemented as stated above. Because this system exists on a PC, a bluetooth dongle and software connecting the Wiimotes to the computer are necessary hardware additions. However, other than this minor detail, the hardware is implemented in exactly the manner described above.

5.1 Head-tracking

The head-tracking system begins by calibrating the user's position in relation to the stationary Wiimote. This position will be known as the center position, or home position. The user indicates their preference of home position by pressing the the B button on the Wiimote used as their laser gun. The math used to calculate this angle was provided by Johnny Lee.

```
else if(ws.ButtonState.B){
   double angle = Math.Acos(.5/headDist) - Math.PI/2; //angle of head
   if (!cameraIsAboveScreen) angle = -angle;
   CameraVerticalAngle = (float) ((angle-relativeVerticalAngle)); // camera angle
}
```

In addition to calculating the head's relative angle to the screen, this code also takes into account whether the Wiimote is above or below the screen.

A function called `ParseHTData` is responsible for taking in information about the location of the infrared lights, and converting it to meaningful data regarding head location and angle. The function essentially moves through each infrared light found and normalizes the values of said infrared light's X and Y coordinates. This ensures that the lights exist in the same coordinate system as the rest of the game's objects. All raw positions are still preserved.

Once all of the light data has been parsed, the code calculates the data to solve for important information about the head's position. If there are exactly two lights visible, the program assumes these lights are those from the head-tracking goggles. The distance between the two points of light is used to solve for the head angle (its X location, Y location, and distance). This math was also provided by Johnny Lee. The projection is then transformed based on these variables:

```
if (remote.WiimoteState.IRState.IRSensors[0].Found)
   {
    wiimotePointsNormalized[0].x = 1.0f - remote.WiimoteState.IRState.IRSensors[0]
    .RawPosition.X / 768.0f;
    wiimotePointsNormalized[0].y = remote.WiimoteState.IRState.IRSensors[0]
    .RawPosition.Y / 768.0f;
    firstPoint.x = remote.WiimoteState.IRState.IRSensors[0].RawPosition.X;
    firstPoint.y = remote.WiimoteState.IRState.IRSensors[0].RawPosition.Y;
    numvisible = 1;
    }

...

device.Transform.Projection = Matrix.PerspectiveOffCenterLH(nearPlane*
    (-.5f * screenAspect + headX)/headDist,
    nearPlane*(.5f * screenAspect + headX)/headDist,
    nearPlane*(-.5f - headY)/headDist,
    nearPlane*(.5f - headY)/headDist,
    nearPlane, 100);
```

5.2 Gestural Input

Another function similar to that of the`ParseHTData`fuction, called `ParseCursorData`, is used to interpret laser gun movements and laser gun shots. Simple event processing based on button input is used to implement shooting, pausing, etc. As stated above, the X and Y locations of the Wiimote relative to the screen are solved for by processing the locations of the infrared lights:

```
if (remote.WiimoteState.IRState.IRSensors[0].Found)
  {
    wiimotePointsNormalized[1].x = (remote.WiimoteState.IRState.IRSensors[0]
    .RawPosition.X / 768.0f)-.6475f;
    wiimotePointsNormalized[1].y = (1.0f - remote.WiimoteState.IRState.IRSensors[0]
    .RawPosition.Y / 768.0f)-.492f;
    bulletX = (remote.WiimoteState.IRState.IRSensors[0].RawPosition.X / 768.0f)
     - .6475f;
    bulletY =  (1.0f - remote.WiimoteState.IRState.IRSensors[0].RawPosition.Y / 768.0f)
    -.492f;
     wiiCursor1.isDown = true;
...
if (wiiCursor1.isDown)
    wiiCursor1.setDown(wiimotePointsNormalized[0].x, wiimotePointsNormalized[0].y);
wiiCursor1.wasDown = wiiCursor1.isDown;
```

Essentially, these X and Y values are the normalized points of infrared light modified by a constant that adjusts for screenwidth and height. Additionally, the crosshair is mapped onto the Aliens' current Z location, implementing a kind of auto-aim system for the user. These coordinates are then used when transforming the location of the cursor.

```
Matrix trans = Matrix.Translation(new Vector3(X,Y,(-3.75f+fSpeed)));
```

5.3 Props

Props are also dealt with through the `ParseHTData` function as they are detected by the same stationary Wiimote that parses head-tracking data. The code determines whether or not there are four infrared lights. If so, then the prop is in use. To ensure that there isn't an error, a tag detection tool, built into many C# libraries, could be used. This could also be used to differentiate between multiple props. In the event that the shield is detected, a translucent blue plane will mask the user's perspective. The user will be unable to shoot and move, though they may prepare to shoot by adjusting the point at which their laser gun is aimed. This is implemented quite simply:

```
if (remote.WiimoteState.IRState.IRSensors[2].Found)
{
    if (remote.WiimoteState.IRState.IRSensors[3].Found)
       shieldUp = true;
    else
       shieldUp = false;
}
else shieldUp = false;
```

5.4 Space Invaders

The space invaders implementation is very straight-forward. Aliens are stored as an array of booleans. If an array index is true, then that alien is alive; otherwise, it is dead. The location of the aliens is determined by using their array index multiplied by a distance constant based on alien size and distance from other aliens on both the X and Y axes:

```
//Draws aliens from top right hand corner on
for (int i = 0; i < 5; i++)
  { startPoint.Translate(-.2975f, .3f, -3.0f);
     transZ.Translate(0, 0, fSpeed);
     transY.Translate(0f, -.15f * i, 0f);
     for (int j = 0; j < 5; j++)
         { transX.Translate(.15f * j, 0f, 0f);
           dev.Transform.World = transX * transY * transZ * startPoint * backNForth;
            if (aliens[i, j] == true) {
                if (collDetect(bnfSpeed, i, j)) {
                    aliens[i, j] = false;
                }
                else drawMesh.DrawSubset(0);
            }
      }
 }
```

Aliens will be able to shoot bullets based on the player's location as determined by head-tracking. Player bullets are fired from a starting point based on their location as determined by the head-tracking system. The goal location of the bullet is a point in space indicated by the X and Y input from the laser gun Wiimote and current Z coordinate of the aliens. For example, if the user is ducked down on the right side of the screen, their bullet will fire upwards and to the left, ultimately aimed at where their cursor points. This implemented rather simply using a slight auto-aim feature specifically aiding aim on the Z axis. The game will assume that the player's goal Z point is the point at which the aliens exist at the moment the bullet is fired, and that the desired X and Y coordinates of this goal point are where the cursor was pointed at that moment as well. The starting point from which the bullet is fired is based on data from the head-tracking system, indicating where in the digital space the user exists. Once a trajectory between said points has been calculated, a constant for speed is implemented to ensure that all bullets move at the same rate.

```
public void createBullet(string type)
{
   for (int i = 0; i < 100; i++ )
      {
         if (bullets[i].inPlay == false)
         {
             bullets[i].inPlay = true;
             bullets[i].type = type;

             bullets[i].goalX = bulletX; //as declared by wiimote pointer
             bullets[i].goalY = bulletY;
             bullets[i].goalZ = -3.75f + fSpeed;
```

```
            bullets[i].xTranslate = headX;
            bullets[i].yTranslate = headY;
            bullets[i].zTranslate = headDist;

            //change factor
            cf = bullets[i].zTranslate - bullets[i].goalZ;
            cf = .1f / cf;

            bullets[i].changeX = -(bullets[i].xTranslate - bullets[i].goalX) * cf;
            bullets[i].changeY = -(bullets[i].yTranslate - bullets[i].goalY) * cf;
            bullets[i].changeZ = -.1f;
            break;
            }

        }
}
```

Bullets are implemented using a struct. The bullets must not only know where they are, but also the trajectory they must follow, and what kind of bullet they are (alien or protagonist) . Thus a struct must be used. Once a bullet has collided with an alien, the protagonist, or exceeded a certain X, Y, or Z boundary, it will disappear.

Chapter 6

Testing

Testing the Portal system manifests itself as a particularly interesting problem. This is because we have issues of both interface design and game design at play simultaneously. In other words, it may not be entirely clear whether the user has problems with the game's interface, for example head-tracking, the interface in conjunction with Space Invaders, or the game Space Invaders itself. To avoid confusing these areas, a duplicate of the Space Invaders game was produced to be as similar as possible to the Portal implementation. Users were then asked to play both games and compare their experiences.

6.1 Non-Portal Space Invaders

It is important that the Non-Portal Space Invaders implementation be as similar to that of the Portal implementation as possible. For this reason, we implemented the Non-Portal version as a first-person experience. However, rather than using head-tracking to control the locomotive movements of the protagonist, the user must use the arrow keys on the keyboard to do so. The 'b' key controls the blast shield, and the space bar controls shooting. Perspective control, utilized by turning one's head in the Portal implementation, is not possible without the use of a head-tracking system or joystick, and is thus eliminated from the Non-Portal implementation. Additionally, the cross-hair with which the user aims their laser gun stays stationary at the center of the screen in this version. Instead, the user must rely on keyboard input to control their aim.

6.2 Comparison

Users were asked to play each version of the Space Invaders implementation, until completion, three times in a row. Half the group was asked to play the Non-Portal implementation first, and the other half to play the Portal implementation first. Once the games were completed, each user was then given a survey on their experiences. The order of these survey's questions were randomized for control purposes.

6.2.1 Survey

The survey is as follows:

1. Which game did you enjoy more?
2. Which game did you find more natural to control?

Answer the following questions on this scale:

1: Strongly Agree 2: Agree 3: Neutral 4: Disagree 5: Strongly Disagree

3. Controlling my character was more enjoyable using the keyboard
4. Shooting was more enjoyable using the Wiimote
5. I enjoyed using the head-tracking system
6. Using the blast shield was more enjoyable using the keyboard
7. I would like to play more games using Portal
8. I would prefer to play games using Portal
9. I would prefer to play games using traditional interfaces like keyboards and gamepads
10. What video game systems do you own?
11. How often do you play video games?
12. General comments

6.3 Concerns

While traditional areas of concern involving surveys are still applicable to the Portal survey, there are a few issues unique to this area of research that must be acknowledged. Firstly, the two games are not exactly alike. We were not able to make every feature of the Portal-based implementation available on the Non-Portal implementation. This is not only because some features were impossible to duplicate without special physical devices, but also because doing so would complicate the game to a point that it would no longer be enjoyable. On that same note, because the interfaces are so vastly different from one another, balance in the Non-Portal implementation had to be refined to ensure that it was as difficult as the Portal implementation. However, one's opinion of difficulty across games is a subjective, and cannot be accounted for precisely.

Our sample size, 21 students, had very little diversity in the way of age and gender. Almost all participants were college-aged women, however, different ethnicities were fairly well represented . This must be taken into account when analyzing the test data, as most games are targeted at a primarily male audience. Lastly, there were technical difficulties during the first 5 tests conducted. These participants played the game in a room in which there was much infrared interference, causing the head-tracking system to malfunction.

6.4 Results

The survey was conducted on 21 participants. The results of the survey clearly indicated that users preferred the Portal version of Space invaders in terms of overall enjoyment of the game, and naturalness of the interface.

Question 1: Which game did you enjoy more? [1]		
Answer	Portal Interface	Keyboard Interface
Raw Data	18	2
Percentage	90%	10%

Table 6.1 Question 1

Question 2: Which game did you find more natural to control?		
Answer	Portal Interface	Keyboard Interface
Raw Data	16	6
Percentage	72%	18%

Table 6.2 Question 2

Question 3: Controlling my character was more enjoyable using the keyboard					
Answer	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
Raw Data	2	6	2	6	5
Average	3.28				
Mean	Agree and disagree				

Table 6.3 Question 3

Question 4: Shooting was more enjoyable using the Wiimote					
Answer	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
Raw Data	8	8	2	2	1
Average	2.04				
Mean	Strongly Agree and Agree				

Table 6.4 Question 4

Question 5: I enjoyed using the head-tracking system					
Answer	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
Raw Data	7	10	2	1	1
Average	2.0				
Mean	Agree				

Table 6.5 Question 5

Question 6: Using the blast shield was more enjoyable using the keyboard					
Answer	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
Raw Data	4	4	3	5	5
Average	3.1				
Mean	Disagree and Strongly Disagree				

Table 6.6 Question 6

Question 7: I would like to play more games using Portal					
Answer	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
Raw Data	8	9	2	1	1
Average	1.95				
Mean	Agree				

Table 6.7 Question 7

Question 8: I would prefer to play games using Portal					
Answer	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
Raw Data	7	6	6	1	1
Average	2.1				
Mean	Agree and Neutral				

Table 6.8 Question 8

Question 9: I would prefer to play games using interfaces like keyboards and gamepads					
Answer	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
Raw Data	2	1	8	7	3
Average	3.38				
Mean	Agree and Disagree				

Table 6.9 Question 9

6.5 Analysis

Questions one and two, particularly when taking into account the fact that almost a quarter of the users experienced technical difficulties while playing, signify that Portal was a great success. However, it is more helpful to know, how, exactly, this success was achieved?

According to the data, the users most enjoyed the head-tracking and shooting aspects of the Portal system, and only felt neutrally about the infrared props. However, we believe that this lack of acceptance of the prop was not due its likeability, but rather flaws in the testing process. Firstly, the prop was only implemented as a Wii sensor bar, and not as a real toy shield. Secondly, players rarely used the blast shield in either implementation. This is because we chose not to penalize the player for being hit in an attempt to avoid players having strong negative feelings towards a version they were less skilled at playing. However, choosing to do this, and not having a fully prepared prop, we feel affected the data.

It is also important to note that though the players enjoyed the Portal system, and generally stated that they would prefer it to regular interfaces, they did not wish to do away entirely with games that use traditional interface systems. Rather, they felt neutrally about them. It seems that though immersive user interfaces are desired, there is still room for gamepad-based systems in gamers lives.

Chapter 7

Future Work

The Portal system is envisioned as a comprehensive and complicated gaming system, and thus has room to grow in many ways. This section will discuss the ways it might advance in the areas of both hardware and software.

7.1 Hardware

Though head-tracking is somewhat successful using the current system, more can be done to make it more natural to the user. For example, if the field of view of the Wiimote camera were more than 45 degrees, one could get more accuracy closer and farther from the screen, thus aiding in the interfacing of locomotive movements. Additionally, the head-tracking system implemented is not stereoscopic, and thus does not create truly three-dimensional displays. Polarized glasses could make this possible. It would also be useful for the Wiimote to detect more than four infrared lights. This would open vast possibilities involving infrared tags. Lastly, in an ideal world, the infrared goggles, numerous sensor bars, and Wiimotes neccessary to implement this interface would come standard in one comprehensive console package, available for multiplayer use.

7.2 Software

It would be useful for tag identification and head-tracking to be built in a way that is more friendly to future programmers. Receiving Wiimote input, however, already exists in a programmer friendly API. For this research, we had to do extensive modifications to the head-tracking system implemented by Johnny Lee, and work from the ground up on seemingly simple tasks such as cursor control. These kinds of programs should come standard in a game platform. An API for these kinds of implementations would prompt more game programmers, and other kinds of programmers, to use the system. This API could also deal with security flaws as discussed in the sections above.

Eventually, it would be interesting to see Artificial Intelligence, beyond that used and described in this work, implemented based on these systems. This could extend as far as taking a laser scan of the players room, and using it in the game. A fellow protagonist could, theoretically, advise the player to hide behind an object to the players left, for example.

Chapter 8

Conclusion

If nothing else, the Portal interface represents a step in the right direction in game interface design. It is highly flexible in its hardware, and allows for traditional game interfaces to exist, while simultaneously functioning as a platform where novel interfaces like head-tracking, gesture-based input, and real-world object detection can be implemented as well. These features function not only has beneficial to games themselves, but also to console advertisement and the creation of merchandise (e.g. collectable props). While this may seem trivial, the marketing appeal of games, or a game's hook, is often integral to its success [4].

As discussed in Chapter 6, we consider the results of our tests to be a success. Not only did the subjects enjoy the Portal system on the whole, but they enjoyed Portal's components individually as well. This is key to Portals success as not all games can function using all of Portals components. Rather, developers should be allowed to choose which functionalities of Portal they wish to use, knowing that these functionalities are effective both in conjunction with one another, and on their own.

Current trends in the gaming industry are not only strong and successful, but have been moving in this direction for over a decade. Users are actively seeking virtual reality environments in their interactions with electronic devices. This is not only made evident by the success of games and systems referred to throughout this thesis, but also by the successes of devices such as the iPod touch, iPhone, and the Microsoft Surface. While games and applications themselves may not have changed much in content over the years, interfaces have evolved drastically, consistently trying to make interacting with computers increasingly natural the user. It is this natural gaming environment, that also functions as a source of entertainment, that Portal seeks to achieve.

Bibliography

[1] Jared N. Bott , James G. Crowley , Joseph J. LaViola, Jr., ”Exploring 3D gestural interfaces for music creation in video games”, *Proceedings of the 4th International Conference on Foundations of Digital Games*, April 26-30, 2009, Orlando, Florida.

[2] Johnny Chung Lee, “Hacking the Nintendo Wii Remote,” *IEEE Pervasive Computing*, vol. 7, no. 3, pp. 39-45, July-Sept. 2008, doi:10.1109/MPRV.2008.53.

[3] Michael Zyda, “From Visual Simulation to Virtual Reality to Games,” *Computer*, vol. 38, no. 9, pp. 25-32, Sept. 2005.

[4] Oxland, Kevin. Gameplay and Design. Essex, England: Pearson Education Limited, 2004.

[5] Shuo Wang and Xiaocao Xiong and Yan Xu and Chao Wang and Weiwei Zhang and Xiaofeng Dai and Dongmei Zhang, “Face-tracking as an augmented input in video games: enhancing presence, role-playing and control”, *Proceedings of the SIGCHI conference on Human Factors in computing systems*, April 22-27, 2006, Montral, Qubec, Canada.

[6] Microsoft Project Natal, “Project Natal”, http://www.xbox.com/en-US/live/projectnatal/, Accessed March 8th, 2010.

[7] Sreeram Sreedharan and Edmund S. Zurita and Beryl Plimmer, “3D input for 3D worlds”, *Proceedings of the 19th Australasian conference on Computer-Human Interaction: Entertaining User Interfaces*, November 28-30, 2007, Adelaide, Australia.

[8] Klaus Petersen and Jorge Solis and Atsuo Takanishi, “Development of a Real-Time Gesture Interface for Hands-Free Musical Performance Control”, *Proceedings of the International Computer Music Conference*, August 2008, Belfast.

[9] B.D. Allen, G. Bishop, and G. Welch, “Tracking: Beyond 15 Minutes of Thought,” *Proceedings of the 28th Ann. Conf. Computer Graphics and Interactive Techniques (Siggraph 01)*, ACM Press, 2001.

[10] Torben Schou and Henry J. Gardner, "A Wii remote, a game engine, five sensor bars and a virtual reality theatre", *Proceedings of the 19th Australasian conference on Computer-Human Interaction: Entertaining User Interfaces*, November 28-30, 2007, Adelaide, Australia

[11] Microsoft, "Managed Library for Nintendo's Wiimote", blog, http://blogs.msdn.com/coding4fun/archive/2007/03/14/1879033.aspx, Accessed February 15th, 2010.

[12] Wikipedia, "Space Invaders", http://en.wikipedia.org/wiki/Space_Invaders, Accessed March 18th, 2010.

[13] Wikipedia, "Project Natal", http://en.wikipedia.org/wiki/Project_Natal, Accessed April 3rd, 2010.

[14] Brian Peek, "A Compendium of Random Uselessness", blog,http://www.brianpeek.com/blog/, Accessed February 15th, 2010.

[15] Sony, "Sony Move", http://us.playstation.com/ps3/playstation-move/index.htm, Accessed April 5th, 2010.

[16] CNN, "Sony Unveils Move, Its PS3 Controller", http://www.cnn.com/2010/TECH/03/11/cnet.sony.move/index.html, Accessed April 5th, 2010.

[17] Wikipedia, "Rumble Pak", http://en.wikipedia.org/wiki/Rumble_Pak, Accessed April 5th, 2010.