Wii3D: Extending the Nintendo Wii Remote into 3D

Submitted in partial fulfilment
of the requirements of the degree of

Bachelor of Science (Honours)

of Rhodes University

João Lourenço

Grahamstown, South Africa
29 October 2010

Abstract

The increasing ubiquity of three dimensional output devices suggests a need for three dimensional input devices. A human computer interactive system, entitled Wii3D, was developed to attempt to fill this need. The Wii3D System uses two Nintendo Wii Remotes to track the movement of infra-red lights, attached to the fingers of a user, in each of the camera's viewports. Stereoscopic triangulation is used to resolve the points in three dimensional space. Further tracking in three dimensions is achieved using predictive interpolation. A framework was designed and implemented to enable the recognition of predefined gestures using Finite State Automata and Discrete Hidden Markov Models. The aforementioned tracking data was processed by the gesture recognition module and raised the relevant events when a gesture was recognized. A user study was undertaken in order to investigate the usefulness of the Wii3D System and the general intuitiveness of the proposed interface and gestures when compared with the standard mouse and keyboard interaction technique. The Wii3D System was found to be more intuitive for multitouch applications, but was not as intuitive as the mouse and keyboard for single pointer applications, even though the proposed interaction technique was found adequate for the majority of the gestures implemented.

ACM Computing Classification System Classification

B.4.1 [Data Communications Devices]: Receivers (Bluetooth), Transmitters (Bluetooth)

D.1.3 [Concurrent Programming]: Asynchronous Events

I.2.10 [Vision and Scene Understanding]: Stereoscopic Triangulation

I.4.2 [Compression (Coding)]: Size Functions

I.5.1 [Models]: Deterministic, Neural Networks, Statistical

I.5.2 [Design Methodology]: Classifier Design and Evaluation, Feature Evaluation and Selection, Pattern Analysis

I.5.4 [Applications]: Computer Vision

I.5.5 [Implementation]: Interactive Systems

Declaration

I hereby declare that this thesis is my own unaided work, unless otherwise referenced. It is being submitted for the degree of Bachelor of Science (Honours) in Computer Science from Rhodes University, Grahamstown. It has not been submitted before for any other degree or examination at any other university.

Signed:

João Lourenço
Rhodes University Computer Science Department
Grahamstown
29th October 2010

Acknowledgements

The following individuals' and groups' valuable advice, support and contributions have been fundamental in the completion of this thesis:

The Andrew Mellon Foundation and Rhodes University for their generous scholarship which enabled me to continue my further studies.

Rhodes University and the Department of Computer Science for the opportunity to complete this Honours degree.

Dr Hannah Thinyane for her attentive supervision during the entire year. Her suggestions, knowledge and direction in every aspect of the Wii3D project were invaluable.

Professor Shaun Bangay for his input regarding the stereo vision algorithms. His thorough understanding of three dimensional graphics and its associated algorithms was pivotal in the implementation of the Wii3D system.

Professor Gunther Jager for his input regarding the theory and use of Hidden Markov Models.

Braden King for his help and direction regarding the WiiGlove design and circuitry.

My Family and Friends for supporting me during the trying times this year, and whose continued love and encouragement gave me the strength not to give up.

Christine Louise Marshall for believing in me, loving me, and being the best thing that ever happened to me. Rest in Peace.

I acknowledge the financial and technical support of this project of Telkom SA, Comverse SA, Stortech, Tellabs, Easttel, Bright Ideas Projects 39 and THRIP through the Telkom Centre of Excellence at Rhodes University.

Contents

List of Figures

Chapter 1

Introduction

1.1 Problem Statement

Due to the migration from two dimensional output devices (such as standard monitors) towards three dimensional output devices (such as 3D televisions and Virtual Reality), there is a possibility to migrate from standard input using the mouse and keyboard to other three dimensional devices [22] (such as Microsoft's Kinect, and, hopefully, the Wii3D System). To interact with three dimensional software using two dimensional input devices (such as computer games and 3D modelling programs), software designers are currently using complicated keyboard sequences and mouse combinations to map the input into the third dimension [4, 45]. As such, there is an even greater need to address the issue of the third dimension.

This thesis aims to investigate the feasibility of using multiple Nintendo Wiimotes to support interaction in three dimensional space.

1.2 Research Goals

Several goals were identified during the project's development:

1. Investigate current state of art in three dimensional interaction to determine best practice

2. Design and implement a tracking technique
3. Investigate effectiveness of using Hidden Markov Models for gesture recognition
4. Evaluate the intuitiveness of the interaction technique by a comparison with existing techniques and devices

1.3 Research Motivation

Johnny Chung Lee's project entitled, "Tracking Your Fingers with the Wiimote", users can track their fingers for use in multitouch applications. This project, however, only deals with two dimensional input data. The motivation behind Wii3D is the extension of Johnny Chung Lee's and development of a three dimensional input technique.

1.4 Thesis Outline

This thesis will cover the following topics:

- Literature Review
- Design and Implementation
- User Study
- Conclusion and Future Work

The literature review will touch on the history of human computer interaction. A closer investigation into the areas of vision-based interaction, gesture recognition and some of the hardware used in the Wii3D System. This chapter will conclude with other work that relates to this project.

The design and implementation chapter will investigate the hardware and software requirements. The hardware section will detail both the "off the shelf" products and the custom made ones. The implementation of the software will follow with a look at the operating system, frameworks and libraries used, the algorithms implemented, the problems faced and how they were solved. This section will conclude with an overview of the final product.

The user study chapter will describe the design and methodology of the experiment conducted. The results will then be discussed and will compare and contrast the differences between the Wii3D System traditional mouse and keyboard technique.

The conclusion and future work chapter will summarize the findings and state the conclusions that have been drawn. Suggestions for future work and extensions will conclude the body of the document.

Chapter 2

Literature Review

This literature review takes a brief look at the research into human computer interaction with gloves, followed by an investigation into vision-based interaction, gesture recognition and some of the hardware used in the Wii3D System. A look at related work will then conclude the review.

The Association for Computing Machinery defines human computer interaction as "a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them"[19]. It also states that the design and implementation of a human computer interactive system should draw on the relevant aspects from both humans and computers.

2.1 Human Computer Interaction using Glove Systems

Historically, gloves have been an interesting focus of research for human computer interaction. The first of the gloves started appearing in the late 1970's[32, 36].

The process of tracking a hand generally involves calculating some of the properties of the hand - position, orientation and pose. There are several documented methods for position tracking when using glove-based input[32]:

- Optical tracking (using marker systems or silhouette analysis)
- Magnetic tracking

- Acoustic tracking
- Circuitry tracking

2.1.1 Optical tracking

There are two main ways to achieve optical tracking - markers and silhouettes. The marker system uses multiple cameras to detect the markers. These markers are either infra-red LEDs that can either be constant or flash in a pattern. The silhouette method uses edge detection to extract the silhouette of the gesture. An analysis of the silhouette is used to determine the position and orientation of the elements of the hand[32].

The marker approach hinges on the ability to triangulate the three dimensional position of the markers in real time. The substantial processing power needed to make the necessary calculations at a rate that is sufficient for real time applications makes this approach difficult. The accuracy of the detection of the markers depends on the number of cameras used. However, as the number of cameras increases, the complexity of the calculation increases as there are more components in the linear algebra system[32].

A great deal of work has been done on natural gestures that are free from gloves, allowing the user a more liberated experience with computer interaction[23]. However, the silhouette approach still has several inherent issues[32]:

- The resolution of conventional video cameras is not high enough to capture the detail required for each individual finger and cover the field of view necessary for large motions
- The framerate of conventional video cameras (30 or 60 frames per second i.e. 33.33Hz or 16.67Hz) does not allow for the capture of rapid movements, while other devices, such as the Wii Remote have frequencies of up to 100Hz[39] and Selspot or Optotrak can operate at above 300Hz
- Parts of the hand can occlude other parts - the occlusion problem cannot be solved with a single camera
- Matchmaking (the ability to map objects in a scene to a three dimensional model) is an inexact science that is still in its infancy

These problems detract from the silhouette's freedom and often ensure researchers taking an alternate approach.

2.1.2 Magnetic Tracking

The magnetic tracking approach uses a source device that generates a magnetic field. A sensor reports its position and orientation in relation to the source. Multiple source and multiple sensor configurations allow the tracking to be more accurate. The magnetic systems that have been developed support polling frequencies of up to 100Hz. The primary advantage of this approach is that line of sight is not necessary for tracking purposes. However, objects with much higher magnetic permeability, such as metals, can cause interference in the magnetic field or fields that are generated by the source device or devices. This can lead to inaccuracies with the positions and orientations of the sensors[32].

2.1.3 Acoustic Tracking

Acoustic tracking is achieved by sending ultrasonic sounds from a source device, mounted on points of interest on the hand, and using receivers in the environment to measure the time taken for the sound to reach them. From this data, the tracking system can triangulate the position of the sources. Unfortunately, this approach requires line of sight from the sources to the receivers and acoustically reflective surfaces can cause interference with the system[32].

2.1.4 Circuitry Tracking

This technique uses hard-wired circuitry to detect touches, bends, and inertia etc. changes by monitoring the sensors built into the glove[32].

2.1.5 Glove Systems

There have been many gloves developed for use with a computer, each with its own merits and downfalls for a wide variety of applications [32, 36].

2.1.5.1 Sayre Glove

Richard Sayre postulated that a glove that used flexible tubes (not fibre optics), with a light source at one end and a photocell on the other could measure the extent to which

a finger is bent. The reported voltage across the photocell is correlated to the finger bending[32, 36]. Thomas DeFanti and Daniel Sandin developed such a glove, which is both inexpensive and lightweight[37].

2.1.5.2 MIT LED Glove

The MIT Architecture Machine Group used a camera focused on an LED-studded glove to track limb position for real-time computer graphics animation. This glove, however, was designed and used for motion capture rather than a control device[32].

2.1.5.3 Digital Data Entry Glove

This glove used hard-wired circuitry that consisted of bend, touch and inertial sensors. Although not commercially developed, this system was developed to recognize 80 unique combinations of sensor readings mapped to a subset of the 96 printable ASCII characters from the gestures defined in the Single Hand Manual Alphabet for the American Deaf[14, 32, 36].

2.1.5.4 DataGlove

The DataGlove, developed by Thomas Zimmerman, used optical fibres to measure the angle of the bend in 10 of the finger joints of the hand to give a description with 6 degrees of freedom (position and orientation) of the hand gesture. The glove was constructed from Lycra and optical fibre that ran along the back of the finger joints. Each glove was calibrated at a per-user level, and this calibrated glove would then solve the configurations of the fingers using the analogue attenuation of the light in the optical fibres in the 10 flex sensors. The glove used a magnet that detected the orientation of the hand in three dimensional space. The glove uses a serial cable to transmit the flex and positional information to a computer[7, 32, 36].

This glove had several key advantages over its predecessors - it operated in real-time, did not require line of sight to a camera, and was lightweight and unobtrusive to the user. This glove was made commercially available by VPL Research at a reasonable cost, and resulted in quite widespread use around the world[32]. The accuracy of the glove's flex sensors was rated at 1°, but research showed that it was closer to 5° or 10°[6]. Furthermore, the operating frequency of 30Hz was not sufficient for precision gestures[32].

2.1.5.5 Dexterous Handmaster

This input device, developed by MIT for the control of Dexterous Hand robot, was far more accurate than the data glove[36], with its 20 degrees of freedom measured (4 per finger) by Hall Effect sensors as potentiometers at the joints[32]. However, the accuracy gained by the glove was at the expense of comfort, made of an intricate aluminium exoskeleton that was attached to the joints throughout the hand[36]. The glove accurately measures the bend of each of the joints in the fingers, the relative rotation of the each finger in relation to the hand and the complex motion of the thumb at 200Hz within 1° of flexion[32].

2.1.5.6 Power Glove

Nintendo, inspired by the VPL DataGlove, designed a glove for its gaming consoles that was constructed from moulded plastic and Lycra to allow flexible movement, with one resistive ink sensor per finger for flex detection. The glove used an acoustic unit mounted on the hand to track the glove in three dimensional space to one quarter of an inch using a television mounted acoustic sensor, with further trackers to determine the rotation of the hand[32].

The Power Glove was not particularly accurate, but its crude gesture recognition abilities were sufficient for the gaming application for which it was designed[32, 36].

2.1.5.7 CyberGlove

The CyberGlove was designed to translate American Sign Language into verbal English. It was constructed from 22 thin foil strain gauges sewn into thin fabric. The analogue signal are processed and converted into a digital streaming signal that is sent to a computer using a serial connection. The observed performance of the glove was smooth and stable, while retaining accuracy within 1° of flexion[32].

2.1.5.8 Space Glove

Virtual Entertainment Systems, a company in the development of arcade games, developed a glove for use with its arcade games that measure the flexion of the fingers using

sensors that measure 1 degree of freedom per finger and 2 degrees of freedom on the thumb. This, in conjunction with the magnetic tracker in the back of the glove that tracks the gloves position in three dimensional space, is used for the gaming interface with other Virtual Entertainment Systems (previously W. Industries) arcade games[32].

2.2 Vision-Based Interaction

Gunnar Grape presented a system for computer vision which maps a hierarchy of features using a two dimensional prototype. The prototype models that were used were various projections of the three dimensional objects as a camera views them from a several different locations. The items in the scenes were limited to planar faced, convex objects. The objects that are recognized are then matched to generalizations of the prototype models [13]. This paper deals with many of the concepts of two dimensional vision-based interaction.

2.2.1 Stereoscopic Depth Perception

The ability of an organism to perceive depth is achieved using the composition of multiple perspectives of the same scene. Human vision has two perspectives that are processed by the brain to resolve an estimate of the third dimension[38], which is important, but not essential, to a human's function. Since the mid twentieth century, researchers have documented investigations into depth perception and judgments of distances in the real world[12, 20].

The stereoscopic depth perception problem is solved by deriving the points in three dimensional space relative to some predefined point in the space. In the world of computer vision, the methods for resolving points using stereo vision are[33]:

1. Production of a camera model: the position and orientation of cameras in three dimensional space
2. Position of matching point pairs: loci of corresponding features in the two pictures
3. Computation of the point in three dimensional space for each point pair
4. Presentation of the resultant depth information

A great deal of research into matching areas in stereo images has been done. Marsha Hannah[17] discusses measures of match which are suitable for the stereo vision matching of areas. She goes on to describe several methods for pruning the search space.

2.2.1.1 Triangulation

The linear triangulation of a point in $\mathbb{R}^3$ from two projections, whose views are known, is simple in a non-noisy environment using geometry[18]. Using Figure 2.1, for triangulation in 2 dimensions, let:

- A be Observer 1
- B be Observer 2
- S The observed point
- l The line between A and B
- d The perpendicular distance from the line between A and B and the S
- α The angle between the l and $\overrightarrow{AS}$
- β The angle between l and $\overrightarrow{BS}$

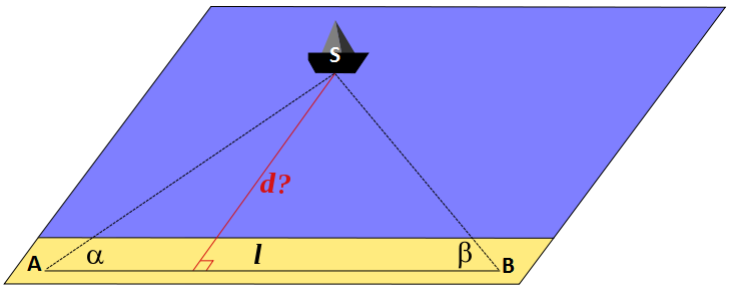


Figure 2.1: Triangulation[42]

Using Pythagoras' theorem:

$$l = \frac{1}{\tan\alpha} + \frac{1}{\tan\beta}$$

Therefore, d can be defined as:

$$d = \frac{l}{\left(\frac{1}{\tan\alpha} + \frac{1}{\tan\beta}\right)} = \frac{l \cdot \sin\alpha \cdot \sin\beta}{\sin(\alpha + \beta)}$$

This can be extended into the third dimension using both horizontal triangulation and vertical triangulation to calculate an estimate of an object's position in $\mathbb{R}^3$.

2.2.2 Matching Point Pairs

Hannah [17] described several ways to find the loci of two corresponding features in the stereo images. In the event of few similar features in each image, it is sufficient to simply do a permutation of pairwise operations on the points and choosing the pairs based on the minimization of the distance between the corresponding points.

2.2.3 Point Tracking

The study by Tziritas[15] shows methods of predictive interpolation by estimating the motion and structure of three dimensional objects from a sequence of images. The study discusses the estimation by using a recursive predictor based on the velocity vectors of a point in the three dimensional space. This predictor is based on a mixture of the previous velocities of the point in question.

In another paper[35], Tziritas goes on to investigate the problem of discontinuity detector which deals with occlusion and algorithmic discontinuities. This detection is attained by checking the errors between the predicted values and the observed values. The detection of such a discontinuity error results in the resetting of the system to start tracking the points from a newly initialized state. The point tracking system is summarized in Figure 2.2.

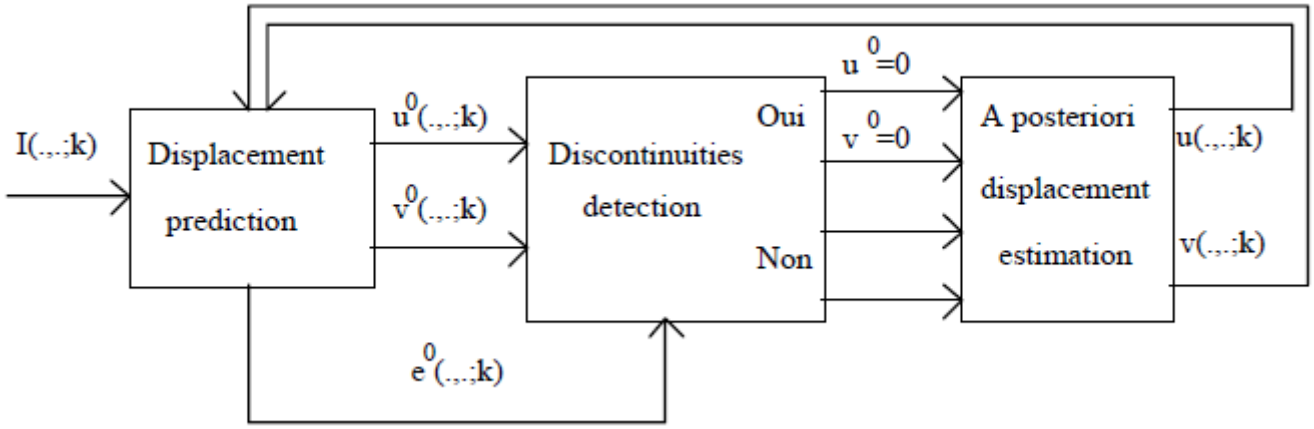


Figure 2.2: Predictive Interpolation

2.3 Gesture Recognition

Humans use gestures, especially hand gestures, for day to day communication. The gestures that humans use are ingrained from childhood, and gestures therefore have an inherently high level of intuitiveness. The use of gestures in computer software gives the user the ability to interact with a computer in a more natural and intuitive fashion.

Fu[10] states that:

> The problem of pattern recognition usually denotes a discrimination or classification of events.

A gesture recognizer uses the spatiotemporal changes as the gesture progresses for its discrimination/classification process[32]. A recognizer generally has three components:

- Encoding - the representation of the gesture
- Classification - the injection of the supported gestures into the recognizer using ideal situations and randomization or by example
- Recognition - the matching of observations to gestures

2.3.1 Encoding

The encoding (representation) of a gesture is important as the optimal transformation can result in a very efficient system. The recognition of shapes in gesture recognition can be

achieved by encoding the data in such a way that the resultant encoding is a pattern that matches a specific gesture. The separation of gestures into categories, or families, allows a system to represent that family of gestures based on common features of the family[10].

2.3.1.1 Size Functions

Frosini[9] proposed a theory that sign language could be recognized by representing the shapes using size functions. The recognition of sign language has been successfully implemented by these representations[25].

A size function is generated by mapping the observations to some measurement system. The importance of choosing the correct measuring system is evident in the possibility of mapping different gestures to the same size function. The possibilities of observed gesture data are inherently infinite and the ability of the classifier to recognize a gesture depends on the size function's ability to map the 'same' gestures to the same encoding. Due to the physical nature of the system, the 'same' gesture can vary, globalized and localized, in its displacement, rotation and scale. These variances need to be taken into account by a process of normalization of the gesture[25]. A simple example of a size function is a mapping of all of the gesture's coordinates' distances from a reference point.

In Figure 2.3, an example of a decomposition of a gesture into a size function is shown. Figure 2.3(a) is the graph of some measuring function, φ. The shaded regions in Figure 2.3(b) and (c) identify the set of points with $\varphi \leq x$ and $\varphi \leq y$ in each of their graphs. In Figure 2.3(d), the darker shaded regions identify the set of points where $\varphi \leq x$ and $\varphi \leq y$ - the union of graphs (b) and (c). Figure 2.3(e) shows the resultant size function for all possible values of x and y. The labelled regions in the size function identify the value of the size function within the underlying region[25].

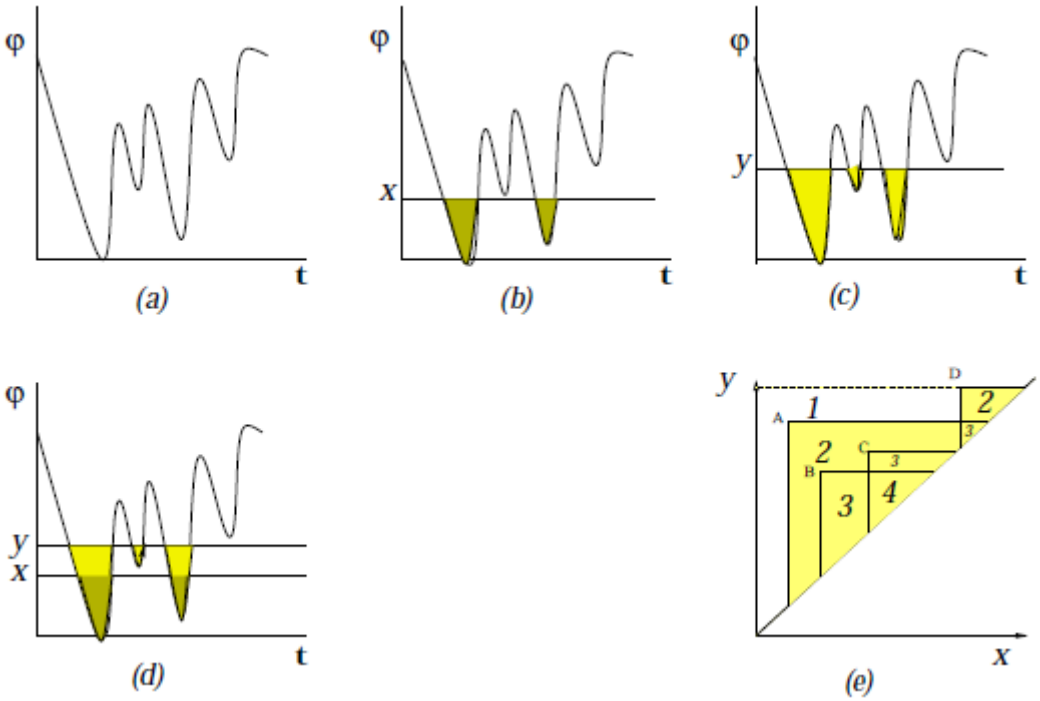


Figure 2.3: Size Function Example

2.3.1.2 Approximate Directional Vectors

The approximate directional vectors are simply the directional vectors between the point at times p_t and p_{t-1} rounded to the predefined principal directions.

2.3.2 Finite State Automata

The use of Finite State Automata for template matching is the simplest approach to recognizing gestures[36]. The gestures are recognized by simply comparing the observed values with the template values, and transitions between the states to a known output results in a gesture being recognized.

2.3.3 Hidden Markov Models

A three dimensional gesture may be recognised and processed using a Hidden Markov Model[22], which is a simple dynamic Bayesian network. This spatiotemporal model,

which is defined below, reduces the 3 dimensional complexity of the hand gesture into a two dimensional problem, and analyses and categorizes these gestures using a state machine. This model has been employed, with success, in speech [31] and handwriting recognition [16].

The HMM can be defined as such[31]:

- states: $S = \{s_1, s_2, \ldots, s_N\}$ where the state at time t is q_t and N is the number of states in the model
- symbols: $V = \{v_1, v_2, \ldots, v_M\}$ where M is the number of distinct observation symbols per state
- probability vector (state transition probability distribution): $A = \{a_{ij}\}$ where $a_{ij} = P(q_{t+1} = s_j \mid q_t = s_i)$ $1 \leq i, j \leq N$
- observation state probability distribution in state j: $b_j(k) = P(v_k \; at \; t \mid q_t = s_j)$ $1 \leq j \leq N$ $1 \leq k \leq M$
- initial state distribution: $\pi = \{\pi_i\}$ where $\pi_i = P(q_1 = s_i)$

The resultant Hidden Markov Model can be visualized as shown in Figure 2.4, where:

- x states
- y possible observations
- a state transition probabilities
- b output probabilities

Sets of Hidden Markov Models are grouped into statistical classifiers. These are statistical methods that map n-feature vector to a point in that same n-space.

2.3.3.1 Learning

The learning task of the Hidden Markov Model is an intractable problem that uses a maximum likelihood approach to determine the best set of state transition and output probabilities, given an output set of sequences. However, the Baum-Welch algorithm is often used to efficiently derive a local maximum likelihood[21].

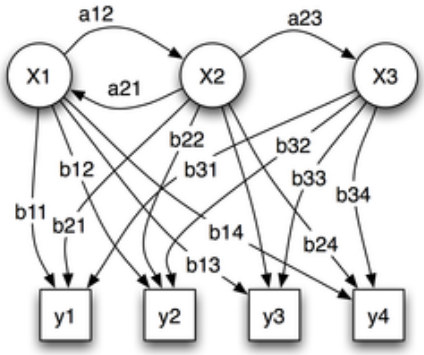


Figure 2.4: Hidden Markov Model[41]

2.3.3.2 Recognition

The recognition process tackles the problem of deciding whether an observed set can be described by the Hidden Markov Model. This is achieved by calculating the probability of the observed set, given the parameters of the model. If this probability is above a predefined threshold, the gesture is recognized as part of the model[21].

2.3.4 Other Methods

2.3.4.1 Artificial Neural Networks

Artificial neural networks are computational models that simulate aspects of a biological neural network. As shown in Figure 2.5, artificial neural networks are built up as a collection of node layers[2]:

- An input layer
- A series of hidden layers
- An output layer

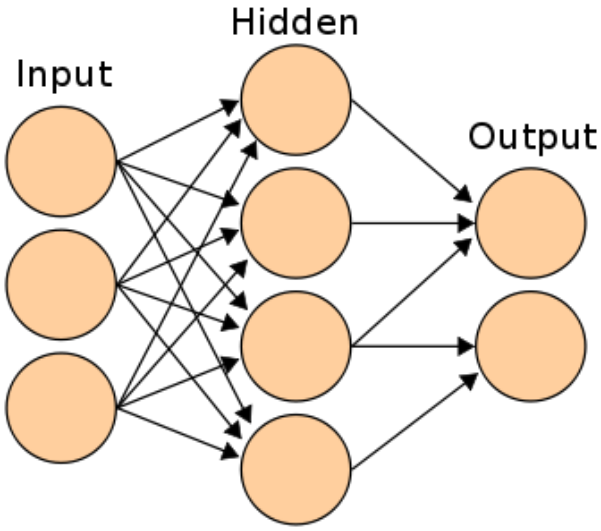


Figure 2.5: Artificial Neural Network[40]

The outputs between states are based on weights are learnt using back propagation through examples. The neural network does not generally perform as well as the more specialized Hidden Markov Models, but this approach has been used with some success in the recognition of sign language gestures in a paper demonstrating the use of size functions used an artificial neural network approach[25].

2.3.4.2 Statistical

Statistical methods of gesture recognition use classifiers, just as the Hidden Markov Models do. Other approaches include:

- Bayesian classifiers
- Hidden Markov Model with Gaussian distributions

Due to the requirement of large training sets and the inflexibility of Hidden Markov Models classification, the Bayesian approach might be preferable, as a study in head gesture recognition showed[43]. The sparse classification model used in that study demonstrated the flexibility of the Bayesian approach.

The Hidden Markov Models can be modified to emit continuous distributions, which would be useful for continuous data rather than transforming the observed values into discrete observations[30, 21].

2.4 Hardware

2.4.1 Nintendo Wii

Although the Nintendo Wii is a proprietary product, and Nintendo has attempted to keep much of its hardware information from the general public, there has been some detailed information released about the products since the game console's inception[39].

2.4.1.1 Technical Specifications

The Nintendo Wii console's hardware has a PowerPC-based "Broadway" CPU, clocked at 729 MHz accompanying an ATI "Hollywood" GPU with a clock speed of 243MHz[3]. The console has 88MB of main memory, of which 24MB is internal 1T-SRAM integrated into graphics package and 64 MB is external GDDR3 SDRAM[5]. It connects to up to 16 Wii Remotes using Bluetooth[27].

The Nintendo Wii Remotes have a 16KB EEPROM chip that allows data and calibration information to be stored, accessed and modified. The Wii Remote camera contains a Charged Coupled Device and an infra-red filter with a hardware resolution of 128x96 pixels, with a reported virtual resolution of 1024x768 pixels at 100Hz[3]. The camera tracks up to four infra-red blobs, whose information, including the blob coordinates and relative sizes, is sent to its paired device using Bluetooth, instead of inefficiently transmitting the data for the entire resolution. Any light that the camera detects whose wavelength is greater 800nm is detected. The camera best detects light sources that have high luminescence and small active areas. The removal of the infra-red filter allows the camera to pick up any high luminescence (bright) objects[44].

2.5 Related Work

Since the release of the Nintendo Wii, there has been a great deal of experimentation into what can be accomplished with the Wii Remotes.

2.5.1 Johnny Chung Lee's Wii Projects

Johnny Chung Lee has developed several applications that use the Nintendo Wii Remotes for interaction with a computer using a Bluetooth connection[24]:

- Tracking Your Fingers with the Wiimote
- Low-Cost Multi-point Interactive Whiteboards Using the Wiimote
- Head Tracking for Desktop VR Displays using the Wii Remote

The projects developed are built using the WiimoteLib library.

2.5.1.1 Tracking Your Fingers with the Wiimote

The tracking of fingers in two dimensions allows a user to have a pointer interface with which interaction with a computer can be achieved by using the Wii Remote's ability to track infra-red blobs, as shown in Figure 2.6.

Figure 2.6: Finger Tracking with the Wiimote

This project uses one Nintendo Wii Remote, an array of infra-red LEDs and infra-red reflectors which are attached to the fingers. The infra-red array is positioned behind the Wii Remote's camera, enabling the users to simple have reflectors attached to their fingers[24].

2.5.1.2 Low-Cost Multi-point Interactive Whiteboards Using the Wiimote

Custom made pens with infra-red lights installed in the tips of the pens are used to track the pens' positions using one Nintendo Wii Remote. This project uses the Wii Remote's two dimensional infra-red blob tracking capabilities. A possible schematic for the pens was suggested, as shown in Figure 2.7[24].

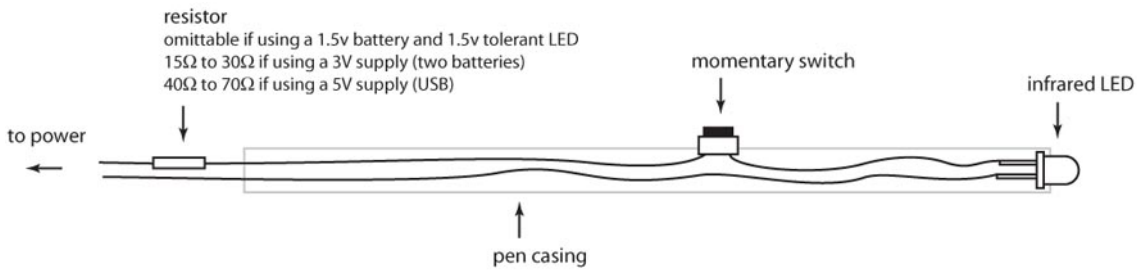


Figure 2.7: Whiteboard Pen

The pen allows a user to use the momentary switch to turn the infra-red light in the pens on or off.

2.5.1.3 Head Tracking for Desktop VR Displays using the Wii Remote

This implementation uses the Wii Sensor bar, mounted on the user's head, to track the user's head. The demonstration application renders targets on the screen based on the position of the Nintendo Wii Remotes, resulting in a more immersive experience, as shown in Figure 2.8.

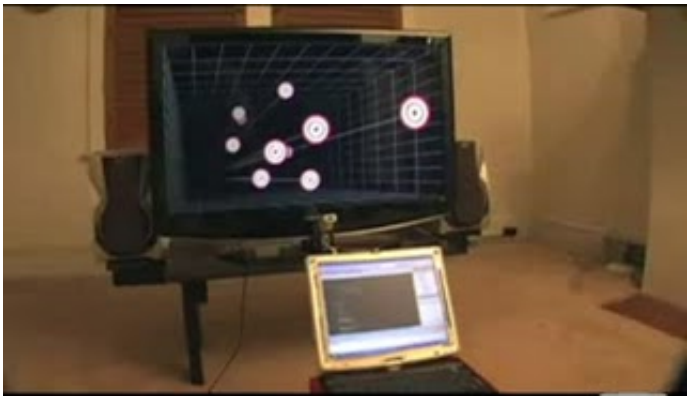


Figure 2.8: Desktop VR

The illusion of depth and space that is created by the rendering engine could add to the quality of computer games and other perspective applications[24].

2.5.2 Design and Implementation of a Hand Tracking Interface using the Nintendo Wii Remote

This project investigated the use of two Nintendo Wii Remotes to track hands in three dimensions for a specific task - molecular visualization applications. The Wii Remote's camera properties were explored, and a hand tracking interface with six degrees of freedom was implemented and tested. The investigator found that this technique was an acceptable method for the visualization of complex molecules, and could be extended to other Computer Aided Design (CAD) applications[44].

2.5.2.1 Camera Details

Wronski[44] experimentally determined the Wii Remote's intrinsic properties necessary for the stereoscopic triangulation calculation by measuring the changes in the horizontal and vertical viewing ranges as the distance from the camera. The results of the experiment are summarized in Figure 2.9

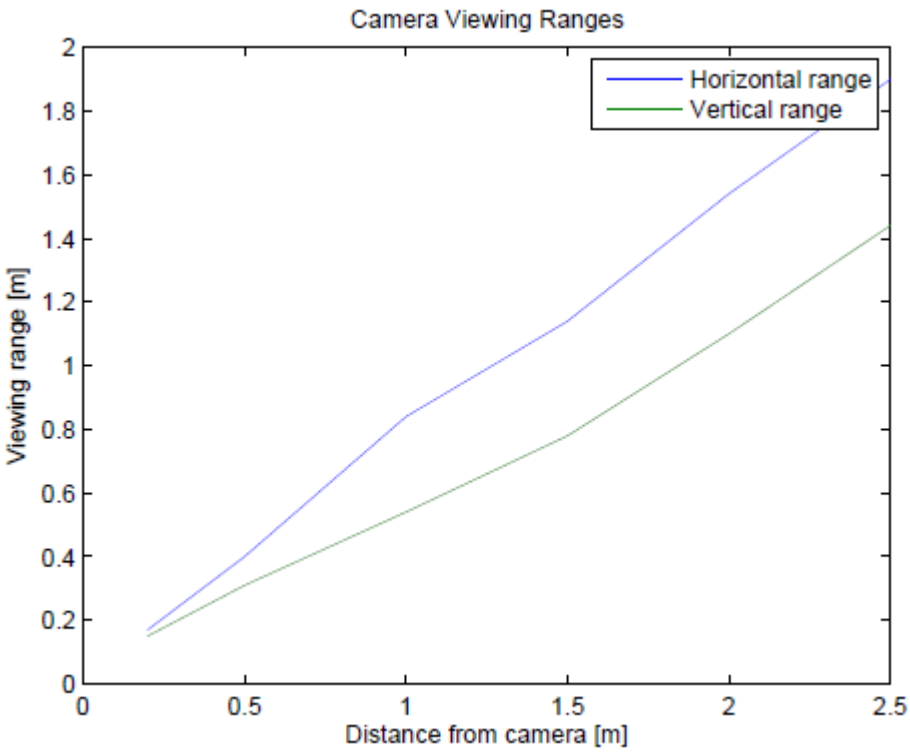


Figure 2.9: Wii Remote Camera Viewing Ranges

Wronski[44] found that viewing angles were 41º horizontally and 31º vertically. The detected near viewing plane and far viewing plane were found to be 10cm and 3m for the infra-red LEDs used in his investigation. Furthermore, he inferred that due to the relatively linear graphs, that there was negligible lens distortion.

2.5.2.2 Implementation

The configuration of the system developed by Wronski entailed a single finger glove, as shown in Figure 2.10, and a layout of Wii Remotes that were placed parallel to each other and 20cm apart[44].

Figure 2.10: Wronski Glove

Wronski's[44] implementation made use of the molecular visualization software package, Avogadro, was used in the investigation due to its easily extensible plugin architecture. The plugin that was developed received tracking data from the two Wii Remotes and modifies the translation, rotation and scale of the viewport in the Avogadro software. He defined several gestures:

- Moving hands closer together/further apart zooms into/out of the molecule
- Moving both hands up, down, left or right simultaneously pans horizontally and vertically
- Moving the left hand down and right hand up, or vice-versa rotates the molecule about z-axis (roll)
- Moving left hand forward and right hand backward, or vice-versa rotates the molecule about y-axis (yaw)
- Moving both hands forward or backward simultaneously rotates the molecule about x-axis (pitch)

2.5.2.3 Outcome

The implementation of the hand tracking interface was found to be an acceptable interface for the manipulation of the Avogadro application viewport. The gestures were successfully recognized and the user experience when using the interaction technique was improved. Wronski investigated the cost of such a setup, and discovered that the hardware, excluding a computer, comes to approximately ZAR2500 (about US$350) [44].

2.6 Summary

There are a multitude of methods for tracking the hand in space, each with its own pros and cons. A great deal of work has been invested in the algorithms involved in hand tracking and their associated approaches to the recognition of gestures. The array of choice in this respect allows researchers and developers to choose an approach that best suits their particular strengths and their systems inherent function and design, while still giving a wide scope in terms of resources required.

The low cost configurations generally yielded lesser quality devices with low accuracy, and those with a higher cost yielded high quality devices with high accuracy. The Wii Remote configuration appears to balance cost, quality and accuracy.

Chapter 3

Design and Implementation

The design and implementation of the Wii3D system is separated into two components - hardware and software.

There are several aspects of the hardware that are discussed. The layout of the Wii remotes will be investigated, in order to see which positions and configurations would work best. The computer requirements and the design and circuitry of the WiiGloves will be described. The software section will deal with the overall design of the implementation, and subsequently investigate how each of the Wii3D System's components were implemented by discussing the algorithms used, with some of the implemented code shown in Appendix B. The Wii3D System's development hinged on the attainment of five main goals:

1. Production of a camera model: the position and orientation of cameras in three dimensional space[33]
2. Position of matching point pairs: loci of corresponding features in the two pictures[33]
3. Computation of the point in three dimensional space for each point pair[33]
4. Tracking the various points in three dimensional space
5. Matching path information to predefined gestures

3.1 Hardware

The Wii3D System hardware configuration included:

- Two Nintendo Wii Remotes
- A Personal Computer with Bluetooth Capabilities
- WiiGloves

3.1.1 Nintendo Wii Remotes

3.1.1.1 Layout of Wiimotes

When designing the proposed layouts of the Nintendo Wii Remotes, several factors were considered:

- Maximization of volume where the infra-red sources would be detected
- Efficiency of stereoscopic triangulation algorithm implemented
- Minimization of error introduced using a particular stereoscopic triangulation algorithm

It is evident from Figure 3.1 that in order to maximize the volume, the Wii Remotes need to be as close together as possible. The increased volume where infra-red sources are detected enables the user to have a larger space in which to interact with the system.

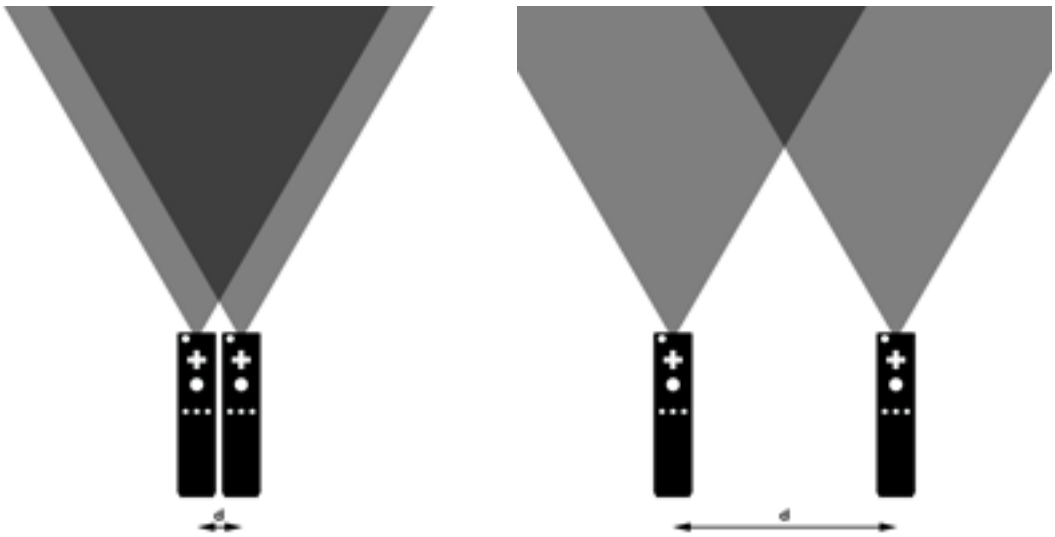


Figure 3.1: Wii Remote Arrangement Maximizing Volume

The efficiency of an algorithm where the cameras are not aligned is decreased substantially. While the volumes of the two arrangements below are similar, the triangulation algorithm for the non-parallel arrangement is not as efficient as that of the parallel arrangement in Figure 3.2.

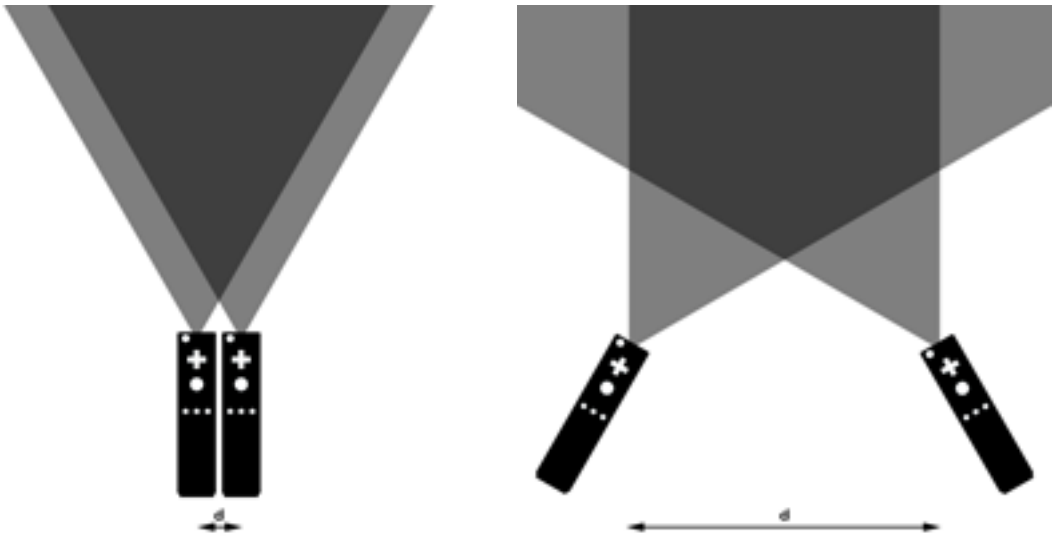


Figure 3.2: Wii Remote Arrangment Maximizing the Efficiency of the Algorithm

3.1.1.2 Wii Remote Camera

The Wii Remote camera's intrinsic properties were measured.

- The horizontal and vertical fields of view of the Wii Remote's camera
- The limits of the x and y coordinates reported by the Wii Remote

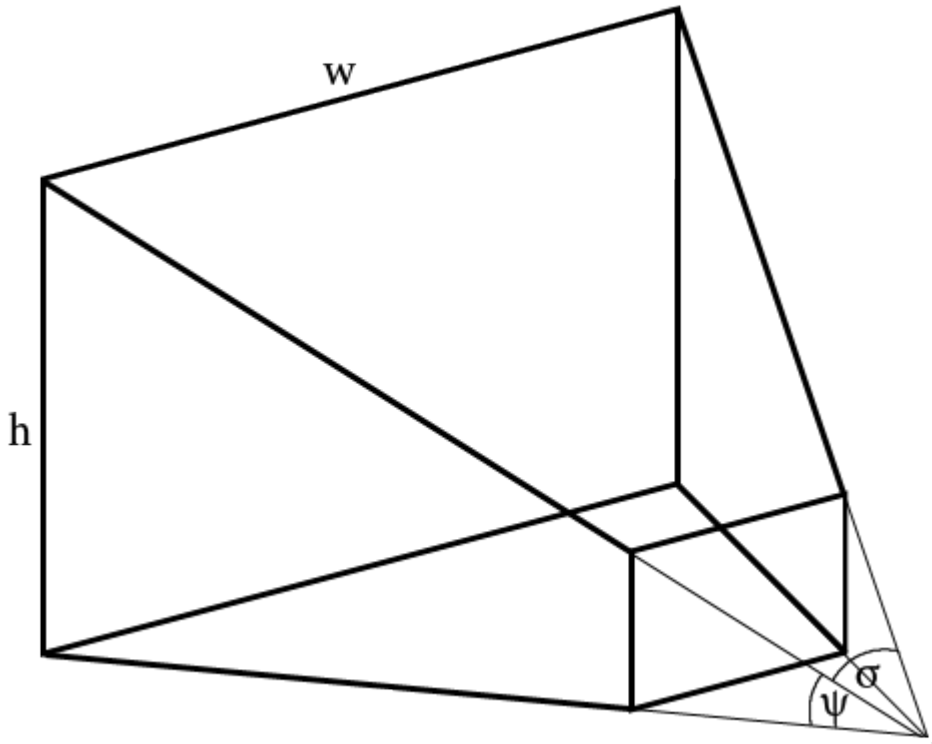


Figure 3.3: Wii Remote Camera's Intrinsic Properties

The horizontal field of view, σ, was found to be 41°, and vertical field of view, ψ, was found to be 31°. These measurements were in agreement with the experiment conducted in the paper entitled, "Design and Implementation of a Hand Tracking Interface using the Nintendo Wii Remote"[44]. For each point, the reported x and y coordinates range from 0 to 1023 and 0 to 767 respectively, and the Wii Remote can track up to four of these points at a frequency of 100Hz[24].

3.1.2 Personal Computer with Bluetooth Capabilities

A personal computer supporting Windows 7 was used for this research. The computer had a Bluetooth dongle that supported the Bluetooth Human Interface Device profile.

3.1.3 WiiGloves

The WiiGloves are custom-made components for use with the Wii3D system. These gloves house the infra-red sources that the Nintendo Wii Remotes track in order to triangulate

each source's coordinates in space. The circuit for each glove is constructed from the following electronic components:

- 1 100Ω Resistor
- 1 Green LED
- 2 Infra-red LEDs
- 1 On/Off Switch
- 1 9V Battery

Ensuring that the current is not too high for the LEDs, the circuit was constructed as shown in Figure 3.4 below.

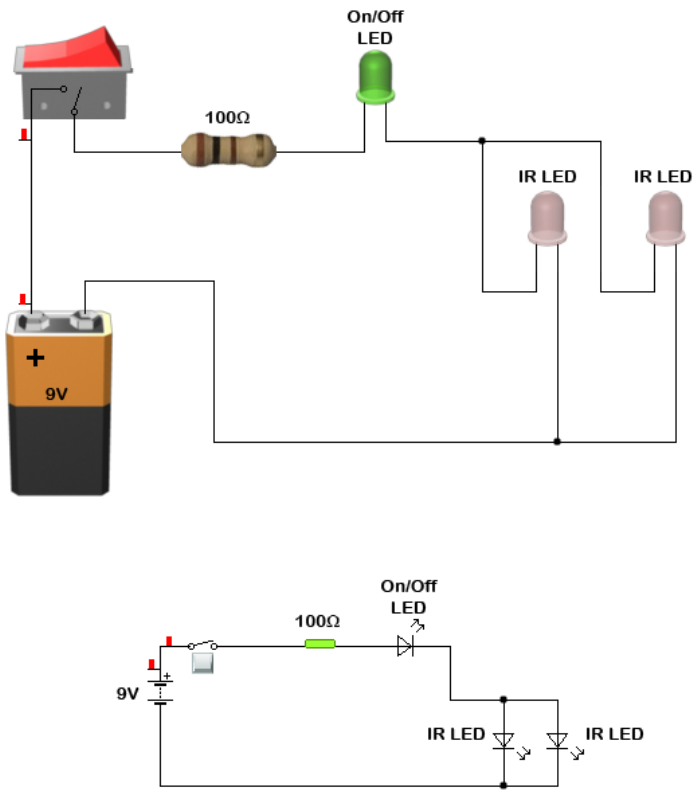


Figure 3.4: WiiGlove Circuit

There are two different configurations for the WiiGlove:

- Two hands with one two infra-red sources on each hand (thumbs and index fingers)

- One hand with four infra-red sources on the thumb, index finger, middle finger and ring finger

The different configurations would give the user more freedom in terms of the way that they perform the different gestures.

3.2 Software

The Wii3D System's design was separated into several components:

- Operating System and Frameworks
- Bluetooth Stack
- Libraries
- Point Tracker
- Gesture Recognizer
- User Interface

Each of the components is discussed to enable a user to set up the Wii3D System and understand how each component fits together.

3.2.1 Operating System and Frameworks

The personal computer was loaded with Windows 7 and the Microsoft Bluetooth driver was installed to run the Bluetooth dongle. The runtime environment selected was the Microsoft .Net Framework 4.0 and was built using Microsoft Visual Studio 2010 as the integrated development environment.

3.2.2 Bluetooth Stack

The Microsoft Bluetooth Stack was used in the implementation of Wii3D. The connection manager made use of a third party library, 32feet.Net, containing the InTheHand.Net assembly that exposes managed interfaces that allow the connection, pairing and management of Bluetooth devices using the stack[8]. It would be possible to use stacks other than the Microsoft Bluetooth Stack (for example BlueSoleil, Widcomm etc.), provided that these Bluetooth drivers and software enable the use of the Human Interface Device profile.

3.2.3 Libraries

Several libraries were used in the implementation of the Wii3D System:

- WiimoteLib 1.7.0.0
- Accord.NET 2.1.1.0
- Math.NET Iridium 2008.8.16.470
- 32feet.NET 2.2.0.0

3.2.3.1 WiimoteLib

The WiimoteLib library is a third party library for managed interaction with connected Nintendo Wii Remotes over Bluetooth[29]. The library exposes two different ways to interact with the Wii Remotes[29]:

- Event driven - every time there is a change in the state of one of the Wii Remotes, an event is fired
- Polling - the Wii Remote is queried at a set interval for its state

The event driven approach would introduce unnecessary complications because of the way that the WiimoteLib built-in events simply monitor a change in the state of the Wii Remotes - the state change is fired each time one of the Wii Remotes detects a change[29]. In order to use this model, whenever an event is fired, the time of the event

would need to be factored into the calculation of three dimensional coordinates of the infra-red sources. This would be accomplished by building a model based on the positions and their associated times and estimating the coordinates from the resulting model. A solution based on this approach was implemented using a linear regression that used the ordinary least squares algorithm. Aside from the inherent overheads generated by the firing of and listening to events, the computational cost of this approach was very high. Furthermore, the propensity of gestures where a resultant model would not be linear or transformable into a linear model made this approach very difficult.

The polling approach ensures that the Wiimote states are synchronized and hence that the stereoscopic estimation of the infra-red sources introduces as little error as possible. The Wii3D system utilises the polling approach and generates its own events from the state changes that it detects.

The decision to use polling rather than events was influenced by three overriding factors. Firstly, the model that was being used for the estimation was not applicable to the problem at hand. Secondly, the overhead created by events which would subsequently fire other events was considered to be too expensive and use resources that would otherwise be put to better use in the gesture recognition component of Wii3D. Finally, and most importantly, when using the polling approach, both of the Wii Remotes' states are received at the same time rather than taking note of the time at which an event was fired and correcting.

3.2.3.2 Accord.NET

The Accord.NET library, built on top of the Aforge.NET library, was used for its statistical modelling capabilities. The discrete Hidden Markov Models present in the library provide the tools for the recognition of the spatiotemporal gestures.

3.2.3.3 MathNET Iridium

The MathNET library was used for its mathematical objects and its associated functions. The stereoscopic triangulation method utilises vectors, matrices, and their associated properties and functions to solve a three dimensional coordinate from two two dimensional coordinates. The MathNet Iridium release exposes the required functionality.

3.2.3.4 32feet.NET

The 32feet.NET library was used for the Windows Bluetooth functionality that it exposes. The Connection Manager allows the user to pair and connect the Nintendo Wii Remotes in preparation for their use in the Wii3D System.

3.2.4 Asynchronous Methods in C#

The C# model for event driven applications allows one to write up asynchronous methods to do background tasks that will fire events upon progress changes and task completion[28]. For example, when using the WebClient user control, one can request a webpage using the code in Listing 3.1.

Listing 3.1: Asynchronous WebClient

```
void DownloadString(string url)
{
  var client = new WebClient();
  client.DownloadStringCompleted +=
     client_DownloadStringCompleted;
  client.DownloadStringAsync(new Uri(url));
}

void client_DownloadStringCompleted(object sender,
   DownloadStringCompletedEventArgs e)
{
  Console.WriteLine("Downloaded: " + e.Result);
}
```

The code in 3.1 creates an instance of the WebClient class, subscribes to its event DownloadStringCompleted, and then calls the asynchronous DownloadStringAsync(Uri uri) method. The advantage of this asynchronous method invocation is that it allows the rest of the GUI to continue to function while the WebClient downloads its data. When the WebClient has finished downloading the contents of the URI, it fires its DownloadStringCompleted event, to which our code is subscribed and will respond to by writing the result to the console.

It is apparent that the use of asynchronous methods is limited to background tasks that will take some time to execute, such as network IO, and tasks that will report their progress or poll an object's status, such as polling an input device's state for changes. The polling paradigm is used extensively in real time applications, such as computer games, whereas the event driven technique is used for other desktop applications.

3.2.4.1 Asynchronous Task Definition

Assume that some time consuming task exists that will need to be run in the background. A basic class, called Program and listed in Listing B.2, has a task defined that should be run in the background.

The DoTask() method will take a long time – it will iterate through a list of integers, printing out each one and pausing for 50ms. If this method were called from a GUI, and the list were sufficiently long, the user interface would hang until the task completed.

3.2.4.2 Converting the Task into an Asynchronous Task

In order to run our task in asynchronously, several changes in the code are necessary.

DoAsync Method

The DoAsync method, shown in Listing B.3, is the asynchronous method that will invoke the worker. The method cannot be called while another operation is under way.

IsBusy Property

The IsBusy property, defined in Listing B.4, is a boolean indicating whether the task is running or not.

DoCompleted Event

The DoCompleted event, defined in Listing B.5, is fired when the operation has completed. The completed event will be fired either after completing normally when the worker completes its work, or after it has been cancelled (the AsyncCompletedEventArgs contain a Cancelled property).

DoProgressChanged Event

The DoProgressChanged event, defined in Listing B.6, is fired when there is a progress update. The worker will fire this event when it has made progress (the ProgressChangedEventArgs contain ProgressPercentage and UserState properties). The UserState property is simply an object that gets passed back from the worker method, and it is up to the developer to choose what to pass back, for example, a status description for a label in the status bar or the object being processed.

CancelDoAsync Method

The CancelDoAsync method, shown in Listing B.7, allows the task to be cancelled by calling this method. This does not cancel the task immediately, but rather marks it for cancellation. The worker will inspect the pDoAsyncContext field for a cancellation flag and will then end gracefully by firing the DoCompleted event.

Note that if there is no asynchronous task underway, this method does nothing.

DoWorker Method and DoWorkerDelegate Delegate

The DoWorker method, shown in Listing B.8, does the actual work that needs to be done. The worker needs to continuously check if the task has been flagged for cancellation, while notifying its subscribed listeners of its progress changes and completion status. The DoWorkerDelegate delegate matches the DoWorker method's signature and is used as the type in other method's parameters.

DoCompletedCallback Method

The DoCompletedCallback method, shown in Listing B.9, is called upon asynchronous task completion. It is responsible for ending the worker, disposing of unused objects and notifying any subscribed listeners of the task's completion.

pDoAsyncContext Field and the DoAsyncContext Class

The pDoAsync field, which is an instance of the DoAsyncContext class, defined in Listing B.10, contains the context for the asynchronous task. The only context necessary for this implementation is the asynchronous method's cancellation information.

pDoSync Field

The pDoSync field, defined in Listing B.11, is used for synchronisation. It ensures that a task is not marked for cancellation during invocation or when the task is completing.

3.2.4.3 Usage

The program shown in Listing 3.2 demonstrates the usage of the changes outlined in Chapter. 50% of the time that this runs, it should list 0 to 99 and then say "Completed", and the rest of the time list 0 to some value less than 99 and then say "Cancelled".

Listing 3.2: Asynchronous Task Usage

```
public static void Main(string[] args)
{
  var prog = new Program();

  //Subscribe to events
  prog.DoCompleted += prog_DoCompleted;
  prog.DoProgressChanged += prog_DoProgressChanged;
  //Start the asynchronous method
  prog.DoAsync();
  Thread.Sleep(new Random().Next(2500, 7500));
  prog.CancelDoAsync();
}

static void prog_DoProgressChanged(object sender, System.
    ComponentModel.ProgressChangedEventArgs e)
{
  Console.WriteLine(e.UserState);
}
```

```
18
19 static void prog_DoCompleted(object sender, System.
     ComponentModel.AsyncCompletedEventArgs e)
20 {
21   Console.WriteLine(e.Cancelled ? "Cancelled" : "Completed");
22   Console.Write("Press enter to exit");
23   Console.ReadLine();
24 }
```

3.2.5 Point Tracker

Point tracking and probabilistic estimation were required to ensure that the system would handle multiple inputs (multitouch) and continue to function even in the event of the loss of points. A loss of points can occur for several reasons:

- An infra-red source can leave the field of view of the Wii Remote
- An infra-red source can move behind another infra-red source
- Two infra-red sources can come together and merge

The point tracker built in to the Wii Remote was found to be insufficient for the Wii3D system. The additional information attained from knowing the permutation of prospective three dimensional coordinates, the two dimensional viewpoint coordinates and their relative velocity vectors allows for a more accurate point tracker by implementing some minimization techniques and a specialised form of predictive interpolation.

3.2.5.1 Stereoscopic Triangulation

The stereoscopic triangulation of a three dimensional coordinate from two two dimensional images from two different viewpoints is achieved using an estimation technique for the near intersection of two rays. In order to achieve this, certain intrinsic properties of the viewpoints are needed[34, 44]:

- Horizontal Field of View

- Vertical Field of View
- Limits of x coordinates
- Limits of y coordinates

As well as information about the layout of the Wii Remotes and the detected infra-red sources[34, 44]:

- The relative positions and orientations of the two cameras
- The two dimensional coordinates of a point from each viewpoint

Using the fields of view, a vector from the camera position to the detected point can be obtained. These vectors can be thought of as rays from the infra-red source to the camera. Any detected point on a ray will report the same two dimensional coordinates.

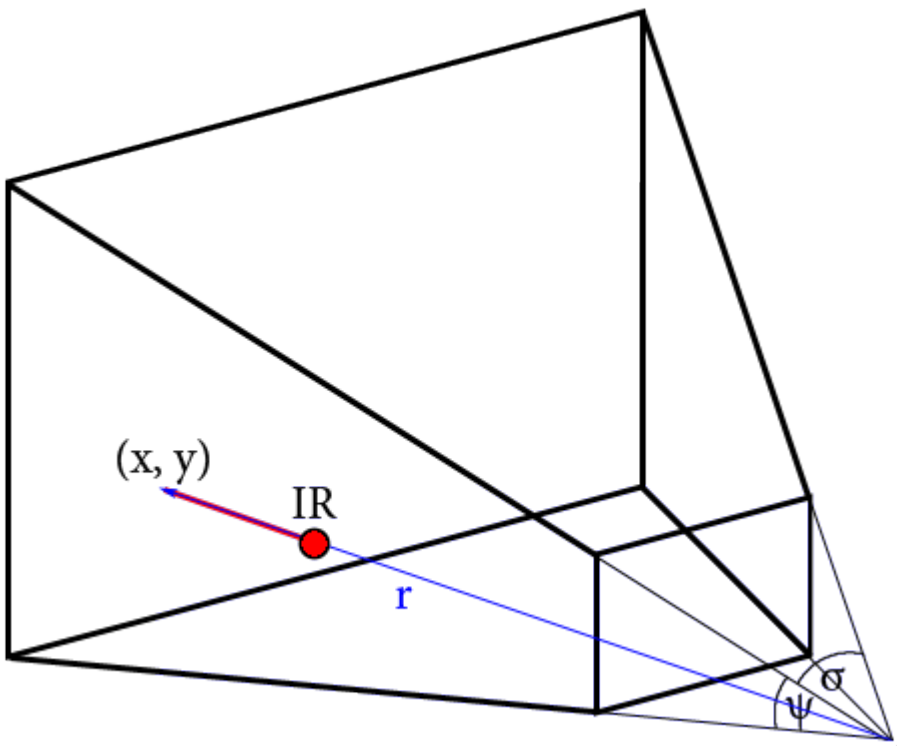


Figure 3.5: Ray Vector

Let:

- σ be the horizontal field of view
- ψ be the vertical field of view
- x be the horizontal component of the two dimensional coordinate
- y be the vertical component of the two dimensional coordinate

Thus, the ray from the camera to the infra-red source can be calculated by the formula:

$$\overrightarrow{r} = [\ (x \cdot \tan\tfrac{\sigma}{2}),\quad (y \cdot \tan\tfrac{\psi}{2}),\quad 1\ ]$$

With the two rays from the two Wii Remote Cameras, the intersection of the two rays can be calculated using linear algebra.

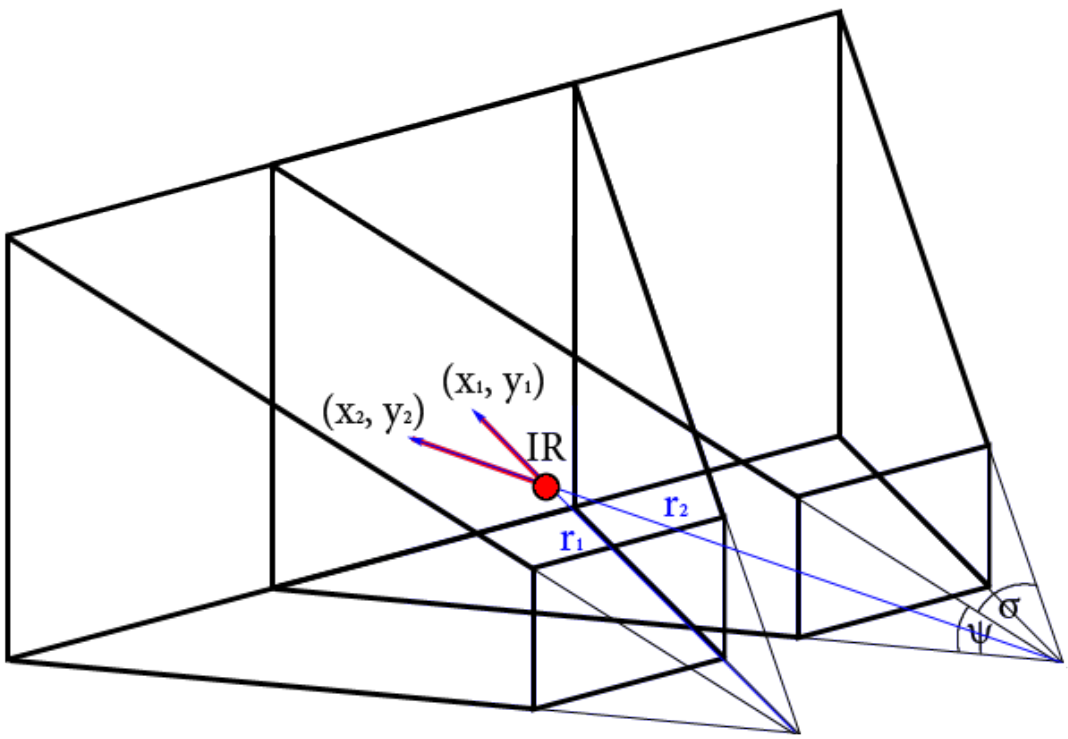


Figure 3.6: Stereoscopic Triangulation

Let the two rays, or the parameterized form of the infra-red source, be defined as:

$$\overrightarrow{r_1} = \overrightarrow{c_1} + \overrightarrow{d_1}s_1$$

$$\overrightarrow{r_2} = \overrightarrow{c_2} + \overrightarrow{d_2}s_2$$

Where:

- $r_1 = \begin{matrix} x_{r1} \\ y_{r1} \\ z_{r1} \end{matrix}$ is the ray vector from the first Wii Remote to the infra-red source
- $r_2 = \begin{matrix} x_{r2} \\ y_{r2} \\ z_{r2} \end{matrix}$ is the ray vector from the second Wii Remote to the infra-red source
- $c_1 = \begin{matrix} x_{c1} \\ y_{c1} \\ z_{c1} \end{matrix} = \begin{matrix} 0 \\ 0 \\ 0 \end{matrix}$ is the position of the first Wii Remote camera
- $c_2 = \begin{matrix} x_{c2} \\ y_{c2} \\ z_{c2} \end{matrix} = \begin{matrix} d \\ 0 \\ 0 \end{matrix}$ is the position of the second Wii Remote camera
- $d_1 = \begin{matrix} x_{d1} \\ y_{d1} \\ z_{d1} \end{matrix}$ is the direction vector from the first Wii Remote to the infra-red source
- $d_2 = \begin{matrix} x_{d2} \\ y_{d2} \\ z_{d2} \end{matrix}$ is the direction vector from the second Wii Remote to the infra-red source
- $s_1 \in \mathbb{R} > 0$ is the scale of the first direction vector, d_1
- $s_2 \in \mathbb{R} > 0$ is the scale of the second direction vector, d_2

In a perfect world, equating the two rays and solving would yield the infra-red source's coordinate. Unfortunately, this is a physical system and there are inaccuracies in the

equipment and measurements[26], including the position and orientation of the Wii Remotes, camera lens distortion and the detection of infra-red sources by the different cameras being out of phase. As such, the two rays would almost never intersect perfectly and the linear system would have no real solution[34, 44]. Therefore estimates of the closest points on each ray to the other ray are necessary.

Let:

$$\hat{s_1} = \frac{\det\begin{pmatrix} o_2 - o_1 & d_2 & d_1 \times d_2 \end{pmatrix}}{|d_1 \times d_2|^2} = \frac{\det\begin{pmatrix} x_{c2} - x_{c1} & x_{d2} & x_{d1} \times x_{d2} \\ y_{c2} - y_{c1} & y_{d2} & y_{d1} \times y_{d2} \\ z_{c2} - z_{c1} & z_{d2} & z_{d1} \times z_{d2} \end{pmatrix}}{|d_1 \times d_2|^2}$$

$$\hat{s_2} = \frac{\det\begin{pmatrix} o_2 - o_1 & d_1 & d_1 \times d_2 \end{pmatrix}}{|d_1 \times d_2|^2} = \frac{\det\begin{pmatrix} x_{c2} - x_{c1} & x_{d1} & x_{d1} \times x_{d2} \\ y_{c2} - y_{c1} & y_{d1} & y_{d1} \times y_{d2} \\ z_{c2} - z_{c1} & z_{d1} & z_{d1} \times z_{d2} \end{pmatrix}}{|d_1 \times d_2|^2}$$

Where:

- $\hat{s_1}$ is the length along the first ray vector
- $\hat{s_2}$ is the length along the second ray vector

Using these lengths along each vector, a final coordinate, κ, can be calculated as the average of the two points, $\hat{r_1}$ and $\hat{r_2}$, given by the distances, $\hat{s_1}$ and $\hat{s_2}$, along their respective rays, r_1 and r_2[44]:

$$\kappa = \frac{\hat{r_1} + \hat{r_2}}{2} = \frac{c_1 + c_2 + d_1 \cdot \hat{s_1} + d_2 \cdot \hat{s_2}}{2}$$

The implementation of this algorithm is shown in Listing B.1.

3.2.5.2 Minimization of Errors by Minimization of the Distance Between $\hat{r_1}$ and $\hat{r_2}$

The points that are reported by the two Wii Remotes are not necessarily sorted in the same order, as shown in Figure 3.7.

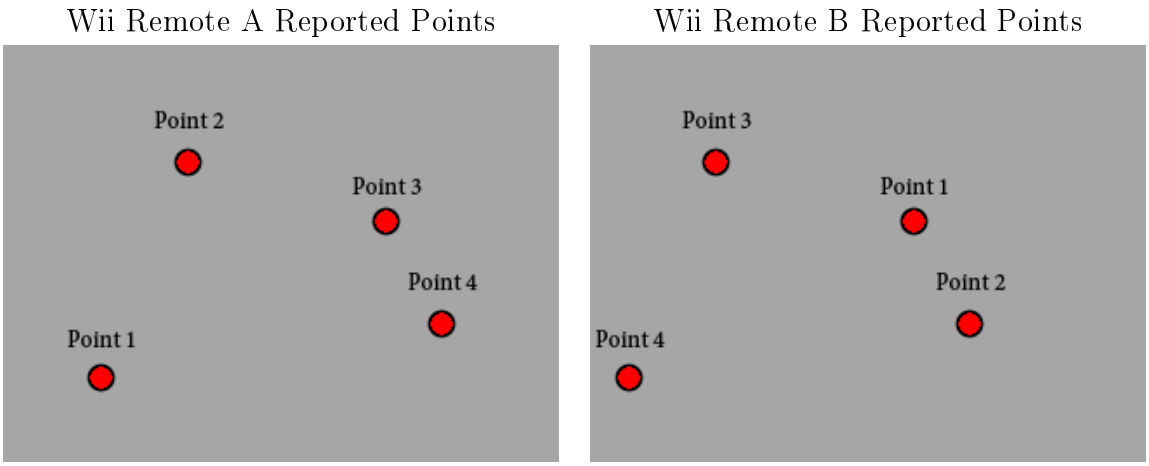


Figure 3.7: Wii Remote Reported Points

Using the algorithm described in Chapter 3.2.5.1, any two points can be used to find a three dimensional point[34]. Therefore, it is necessary to find pairs of points which would most likely yield an intersection[33, 17]. This is achieved by minimizing the distance between $\hat{r_1}$ and $\hat{r_2}$.

A matrix of prospective three dimensional points, K, is generated by taking a pairwise permutation of the two dimensional points and using the stereoscopic triangulation algorithm described in Chapter 3.2.5.1:

$$K = \begin{matrix} \kappa_{0,0} & \kappa_{1,0} & \dots & \kappa_{3,0} \\ \kappa_{0,1} & \kappa_{1,1} & \dots & \kappa_{3,1} \\ \kappa_{0,2} & \kappa_{1,2} & \dots & \kappa_{3,2} \\ \kappa_{0,3} & \kappa_{1,3} & \dots & \kappa_{3,3} \end{matrix}$$

The index, i, of the point, $\kappa_{i,j}$, that yields the smallest distance of the set of points in row j, is chosen as the point that will be used for the final set of points in index i. The distances permissible are limited to being below a predefined threshold value.

3.2.5.3 Minimization of Errors by Predictive Interpolation

The predictive abilities of the tracking algorithm depend on two important factors:

- The instantaneous velocity - the vector difference between the current point and the previous point
- The average velocity - the average of the instantaneous velocities over a period of time

A prediction is made by taking a weighted average of the two velocities and adding this vector to the last point. Applying this to each point, a matrix of errors is calculated using the formula:

$$e_{i,j} = |x_i - E(x_j)|$$

Where:

- i is the observed points index
- j is the expected points index
- $e_{i,j}$ is the error at index [i,j]
- x_i is the prospective observed point (obs_i)
- $E(x_j)$ is the expected point (exp_j)

As such, the following matrix of errors is produced:

$$\begin{matrix} e_{0,0} & e_{1,0} & ... & e_{3,0} \\ e_{0,1} & e_{1,1} & ... & e_{3,1} \\ e_{0,2} & e_{1,2} & ... & e_{3,2} \\ e_{0,3} & e_{1,3} & ... & e_{3,3} \end{matrix} = \begin{matrix} |obs_0 - exp_0| & |obs_1 - exp_0| & ... & |obs_3 - exp_0| \\ |obs_0 - exp_1| & |obs_1 - exp_1| & ... & |obs_3 - exp_1| \\ |obs_0 - exp_2| & |obs_1 - exp_2| & ... & |obs_3 - exp_2| \\ |obs_0 - exp_3| & |obs_1 - exp_3| & ... & |obs_3 - exp_3| \end{matrix}$$

The point with the minimum error for each row is selected as the point for the row index. This allows for duplicate values as coordinates, for example, when a point is clicked. This

assumes that there are four infra-red sources visible at all times (including a duplicate for each click).

In the event of a point actually disappearing from the volume, a maximum error value is imposed on the matrix of errors, based on either a constant or the aforementioned velocities of the infra-red sources. In order to cater for the event where there are not four infra-red sources visible at all times, the same strategy was used with an allowance for the missing points to appear in the next frame.

3.2.5.4 Polling and Asynchronous Events

A slightly modified version of the asynchronous model described in Chapter 3.2.4 was used in the point tracker. At a set interval, the point tracker asynchronously examines the state of the two Wii Remotes and solves for up to four three dimensional points, which are subsequently added to the point buffer. The point buffer is then examined by the gesture recognizer for any recognizable patterns.

3.2.5.5 Point Tracking Ability

The point tracker implemented, although not perfect, was sufficient for the gesture recognition system. The tracking methods used would fail in some cases, for example, when a user had two infra-red sources merge and upon reaching the edge of one of the Wii Remote's viewports, only one leaves and one stays. In this case, the system thinks that there are still two of the pointers when there should only be one. Any problems that the tracker encountered were easily dealt with by removing all of the points from both of the viewports (i.e. hiding the infra-red sources), and move the points back into the volume of gesture recognition.

3.2.6 Gestures

There are two different types of gestures defined:

- A deterministic gesture which tracks movement and uses Finite State Automata to recognize simple gestures

- A non-deterministic gesture which makes use of statistical methods and artificial intelligence to recognize more complex gestures (Hidden Markov Models)

The principal component of the recognizer is the Gesture class. The individual gestures are extensions of this class. Any gesture can be defined using the finite state automata approach or the Hidden Markov Model approach.

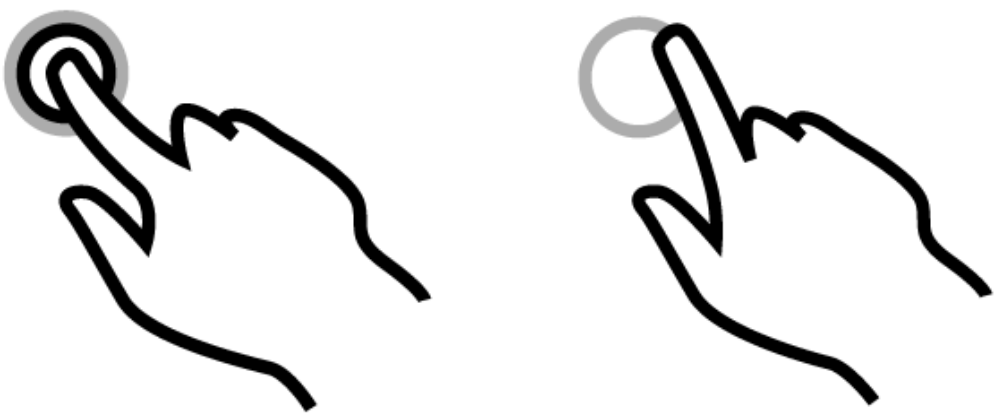
There are predefined gestures implemented in the Wii3D System:

- Click
- Pan/Scroll
- Zoom
- Rotate
- Circle

The images for the gestures are sourced from GestureWorks[11].

3.2.6.1 Click Gesture

The click gesture is defined as a finger going down and forward, and then up and backward, as shown in Figure 3.8. This gesture was defined using a Finite State Automaton.



Finger Down Finger Up

Figure 3.8: Click Gesture

3.2.6.2 Pan/Scroll Gesture

The pan/scroll gesture is defined as two fingers that are close together moving in similar directions, as shown in Figure 3.9. This gesture was defined using a Finite State Automaton.

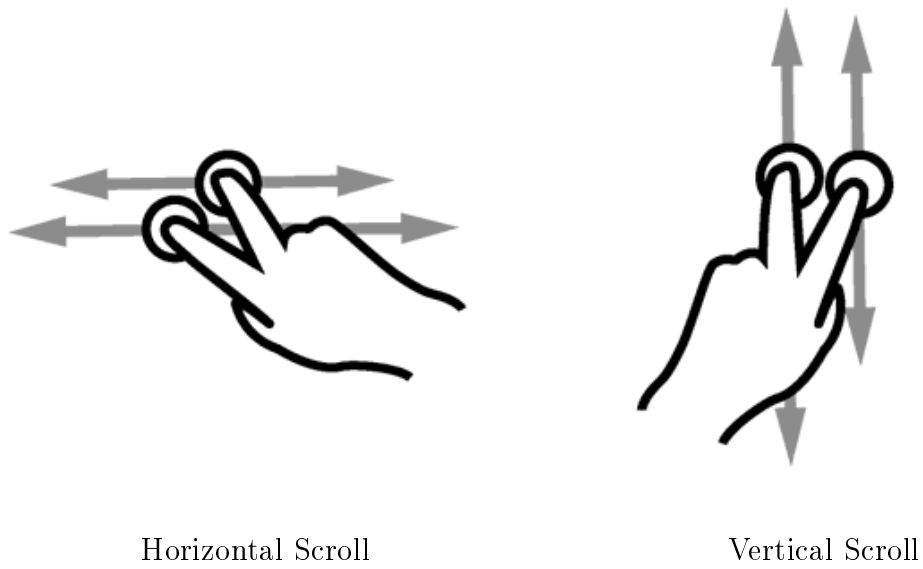
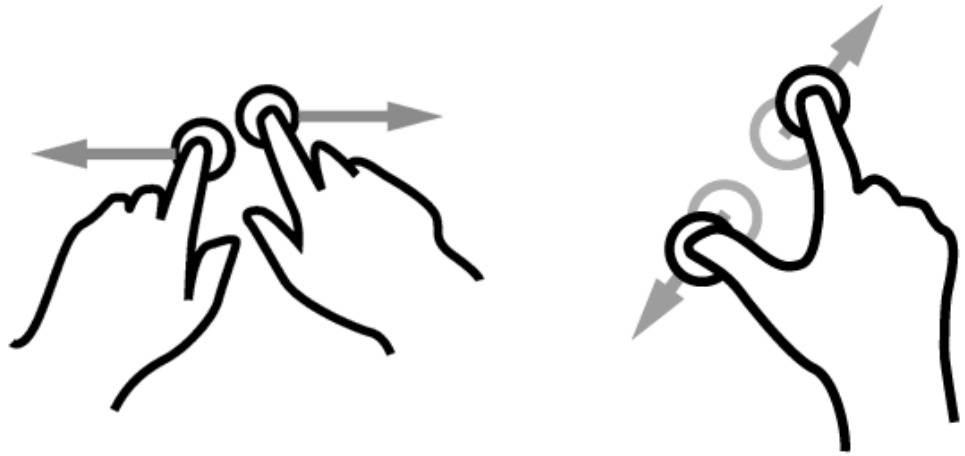


Figure 3.9: Pan/Scroll Gesture

3.2.6.3 Zoom Gesture

The zoom gesture is defined as two fingers moving in opposite directions with the distance between them increasing, as shown in Figure 3.10. This gesture was defined using a Finite State Automaton.



Two Finger Zoom In Pinch Zoom In

Figure 3.10: Zoom Gesture

3.2.6.4 Rotate Gesture

The rotate gesture is defined as two fingers rotating about a pivot. This was accomplished by making sure that the two fingers are moving in opposite directions with the distance between them staying the relatively constant, as shown in Figure 3.11. This gesture was defined using a Finite State Automaton.

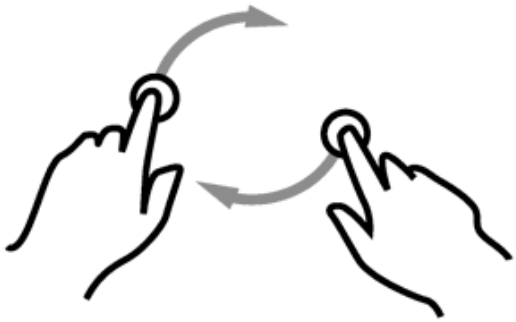


Figure 3.11: Rotate Gesture

3.2.6.5 Circle Gesture

The circle gesture is defined as a finger rotating 360 degrees about a pivot, as shown in Figure 3.12. This gesture was defined using a Hidden Markov Model.

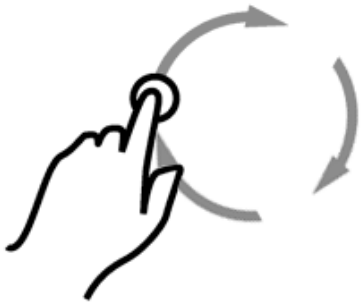
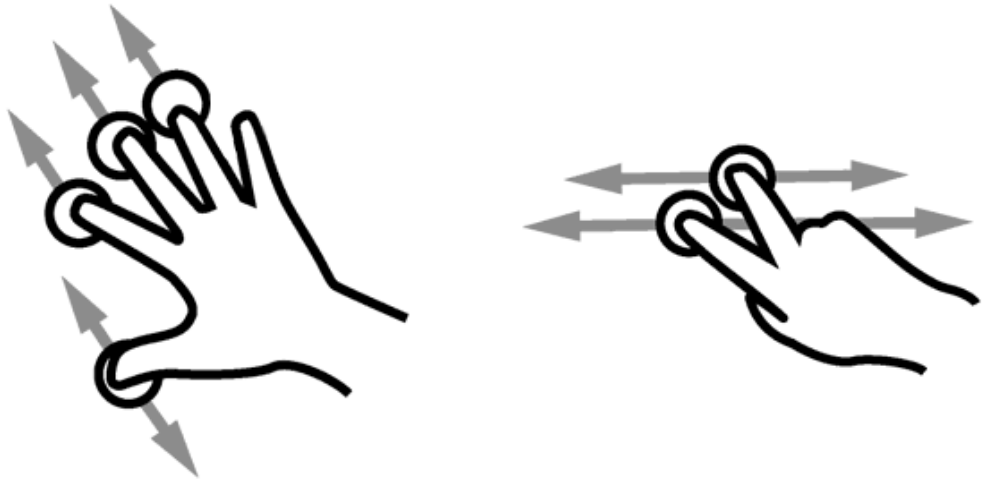


Figure 3.12: Circle Gesture

3.2.6.6 Multitouch Gestures

The Wii3D System detects up to four points, and the number of points used for each gesture has been linked to the intensity of the gesture. For example, a pan gesture using four fingers, as shown in Figure 3.13 will have more impetus than a pan gesture using just two fingers. Due to the pairwise recognition of gestures described in Chapter 3.2.7, this is taken care of by the Wii3D System.



Multitouch Pan Normal Pan

Figure 3.13: Multitouch Pan/Scroll

3.2.7 Gesture Recognition

There are three types of gestures that are defined in the Wii3D System:

- Pointer movement
- Finite State Automaton gesture
- Discrete Hidden Markov Model gesture

The pointer movement event is fired at the same rate as the Wii Remotes are queried for their coordinates. Once the Wii3D System processes the points and yields its resolved three dimensional coordinates, the event is fired and any subscribed components can then process the events.

3.2.7.1 Encoding

The Wii3D System uses two ways to encode the three dimensional coordinate data in preparation for recognition:

- Size functions

- Approximate directional vectors

The size functions are used for the Hidden Markov Model classification model. The resultant size function graph is further encoded to use only integral values, bounded within predefined maximum and minimum values. This doubly encoded array can then be used as an input for the Discrete Hidden Markov Model.

The approximate directional vectors are calculated by finding the closest unit vector of the principal directions and using this vector's integral identification value. The principal directions have been defined as Up, Down, Left, Right, Back and Forward. Using simple vector algebra, the angles between the vector and each of the principal axes are calculated, and the approximated direction vector is chosen as the unit vector with which the observed vector has the smallest angle.

3.2.7.2 Finite State Automata

The Finite State Automata gestures are processed using a deterministic recognition method for template matching. The encoding used for this method of recognition is the approximate directional vector approach. Using the gestures defined in Chapter 3.2.6, the templates are defined and exactly matched against the encodings. The direction vectors, however, are not sufficient for this approach on their own. However, there is more information that is needed. Size of the different gestures is required to ensure that the gestures are above a predefined minimum, which are achieved by examining the sizes of the vectors before they are processed for the directional vectors.

3.2.7.3 Hidden Markov Models

The Hidden Markov Models are taught to the Wii3D System at application start-up. Generation of sample data is done randomly about an ideal model. Gestures of different lengths with random sizes of each component are generated. For example, when using the approximate directional vectors, uniformly distributed lengths of the gesture are taken about a mean length, which is then divided into sections that are uniformly distributed.

An example of the generation of sample inputs for a circle gesture is:

- 3-6 left vectors

- 3-6 down vectors
- 3-6 right vectors
- 3-6 up vectors

By taking large samples of the randomly generated gestures, the Wii3D System uses the Accord.NET framework to build a model by example data.

Under the assumption that hand gestures with four points that need to be followed do not vary from person to person as much as speech or handwriting does, as there are much fewer degrees of freedom[1], the Hidden Markov Model will therefore be implemented instead of the Artificial Neural Network approach. Furthermore, the number of states the model will be limited, which removes the requirement for excessively large training sets.

3.2.8 User Interface

The user interface was designed to be a harness that displays the status of the Wii3D System and reports any events that the Wii3D System detects. The system is used by:

- Starting up the application
- Connecting two Wii Remotes
- Polling for any events

3.2.8.1 Start-Up

Upon start-up, the window shown in Figure 3.14 is presented to the user.

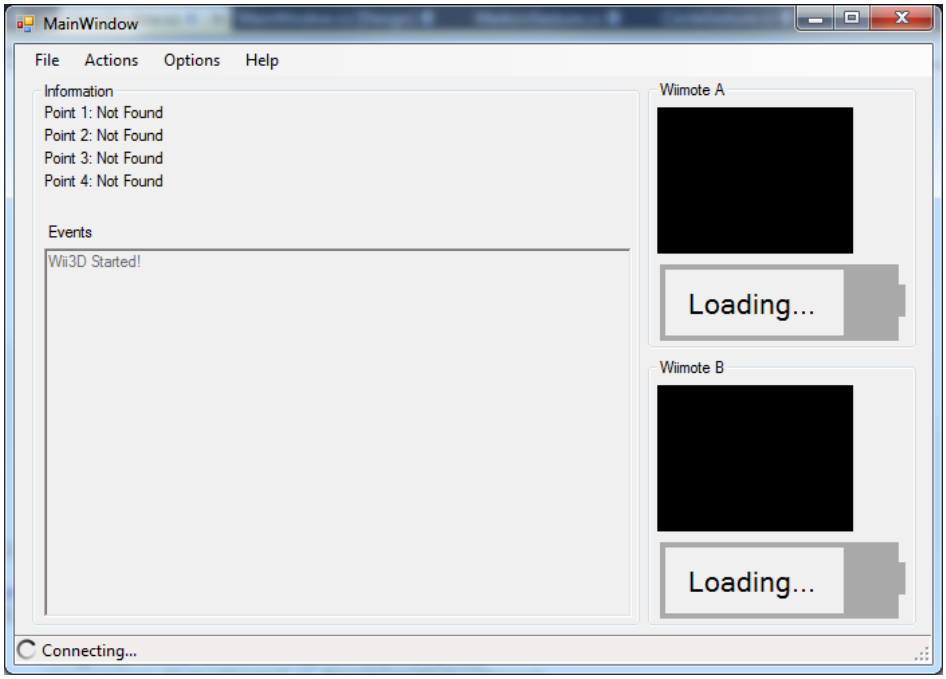


Figure 3.14: Start-Up - Connecting

The Wii3D System will attempt to initialize by connecting to the Wii Remotes. If the connection is unsuccessful, there are two possible errors that could be the cause, as shown in Figures 3.15 and 3.16.

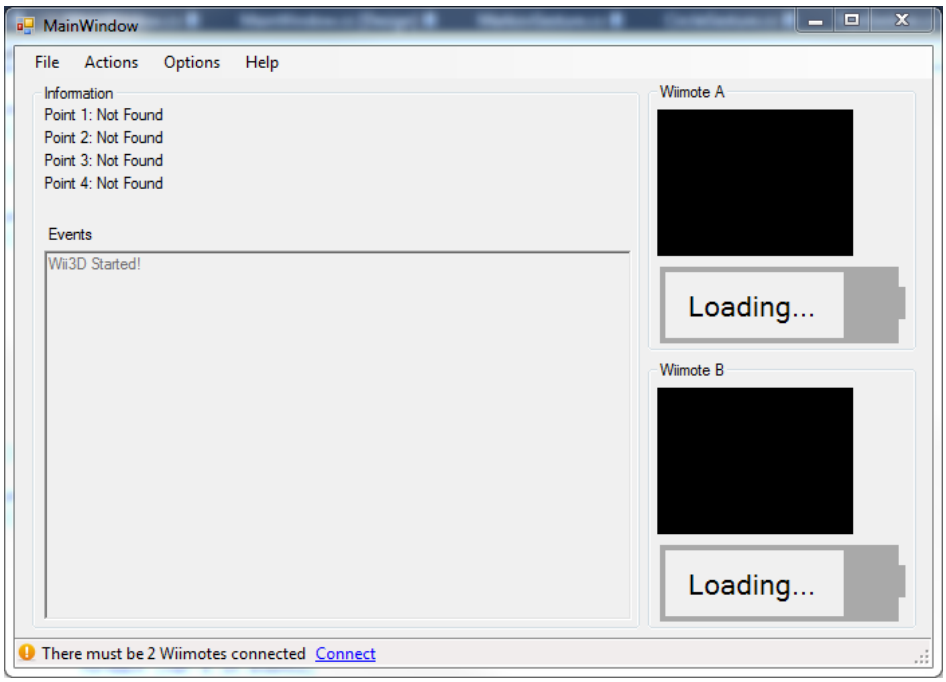


Figure 3.15: Connection Errors - Number of Wiimotes Connected

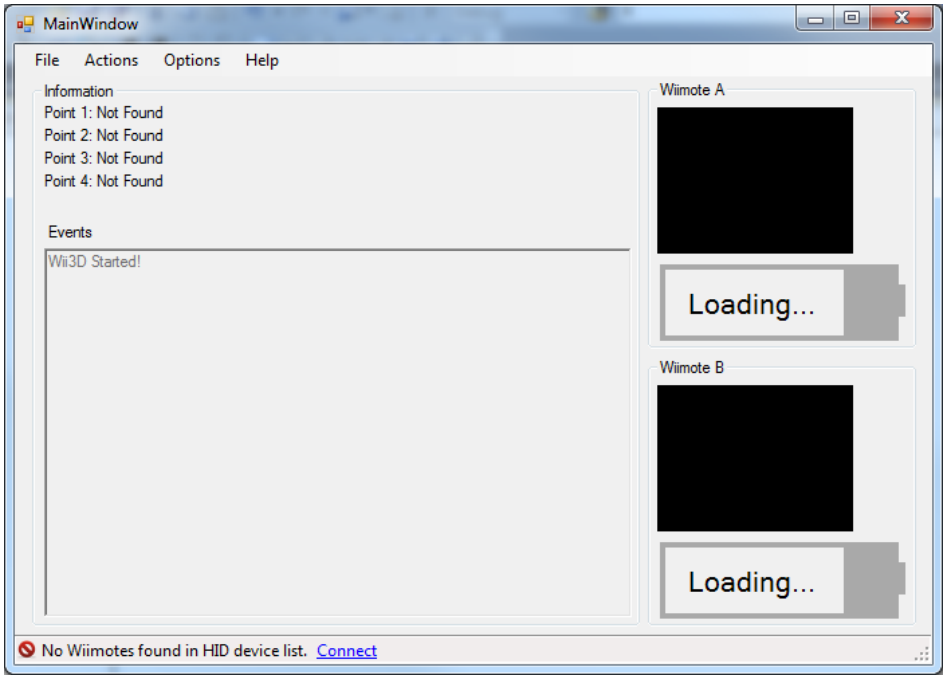


Figure 3.16: Connection Errors - No Wiimotes Found in the HID Device List

The connection errors result in the 'Connect' link being displayed in the status strip.

3.2.8.2 Bluetooth Connection

When the 'Connect' link is clicked, the Connection Manager, shown in Figure 3.17 is launched.

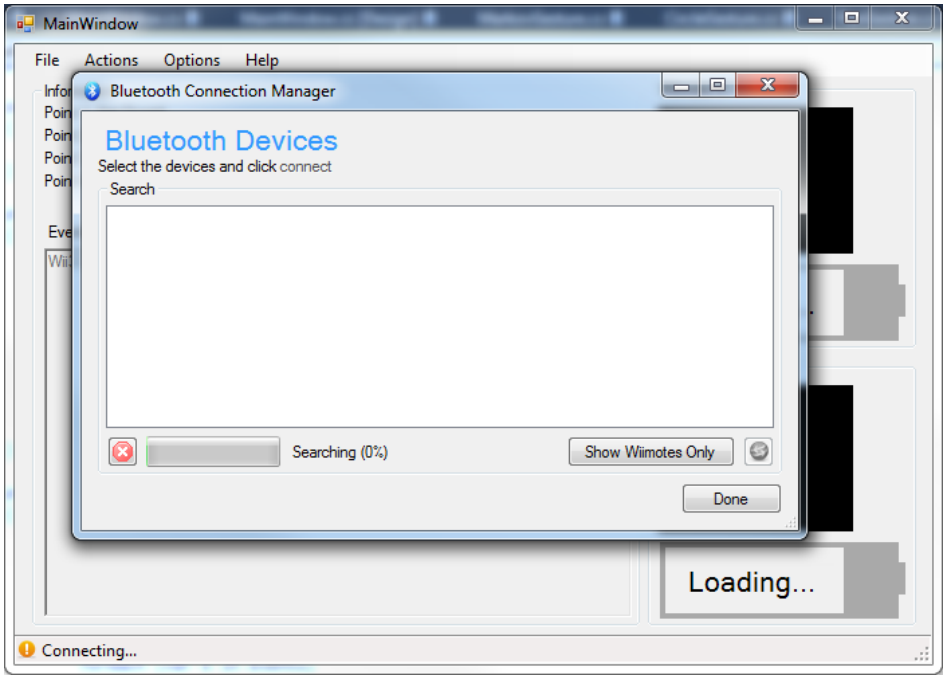


Figure 3.17: Connection Manager - Searching

The manager asynchronously searches for any Bluetooth devices in range, shown in Figure 3.18. The selection of any devices will enable the 'connect' buttons.

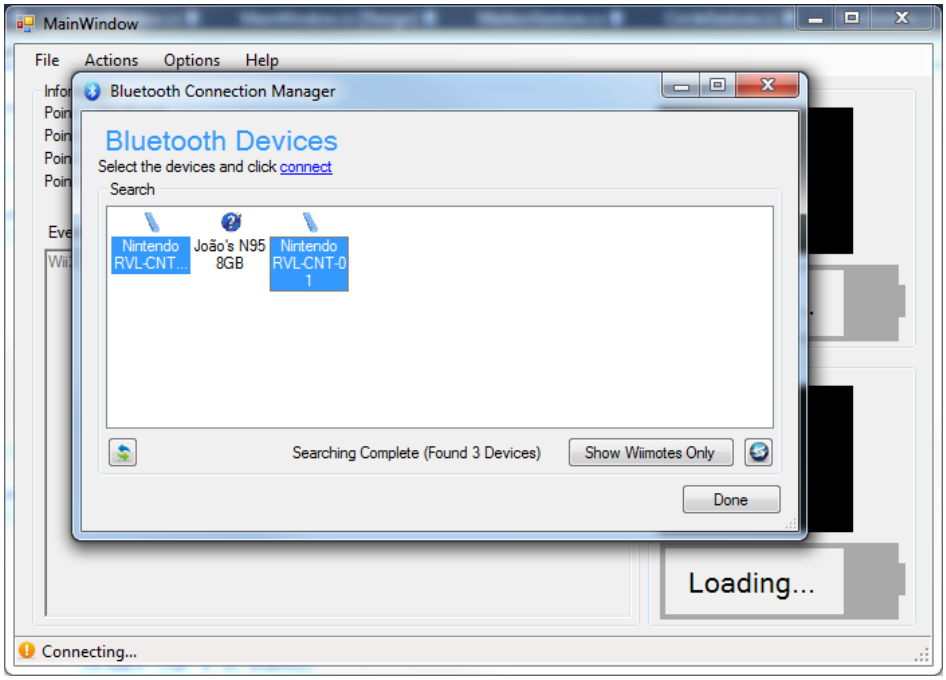


Figure 3.18: Connection Manager - Devices

Once one of the connect buttons is clicked, an information box telling the user that he/she should wait until the devices have finished installing.

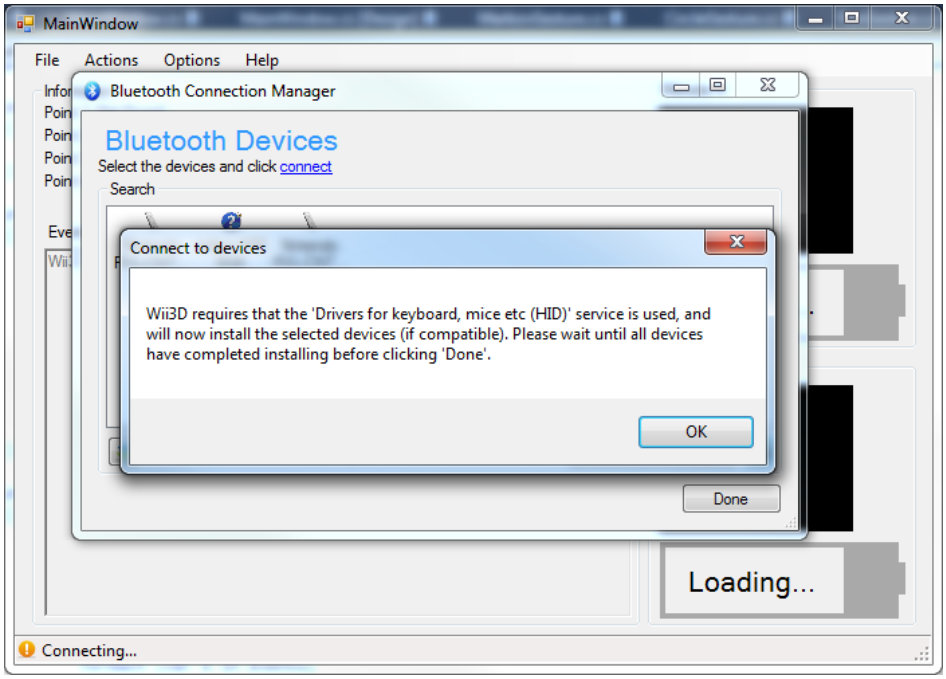


Figure 3.19: Connection Manager - Device Installation

Once the user clicks 'OK', the Wii3D System will initialize the driver installation of the Wii Remotes and the installation progress will be updated in the Windows tray. Once the installation of both of the devices has been completed, the user may then click 'Done'.

3.2.8.3 Polling and Events

When the Wii Remotes have been connected successfully, the status strip will reflect this as shown in Figure 3.20.

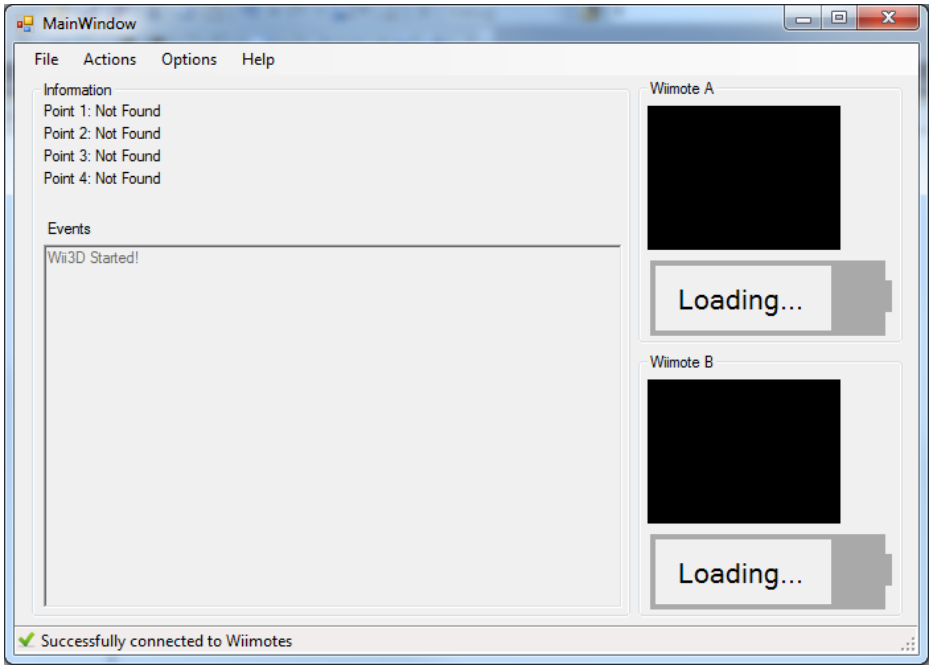


Figure 3.20: Successful Connection

The Wii3D System does not start polling automatically. The user needs to click the 'Actions' button in the menu strip and then click on the 'Start Polling' menu item, as shown in Figure 3.21.

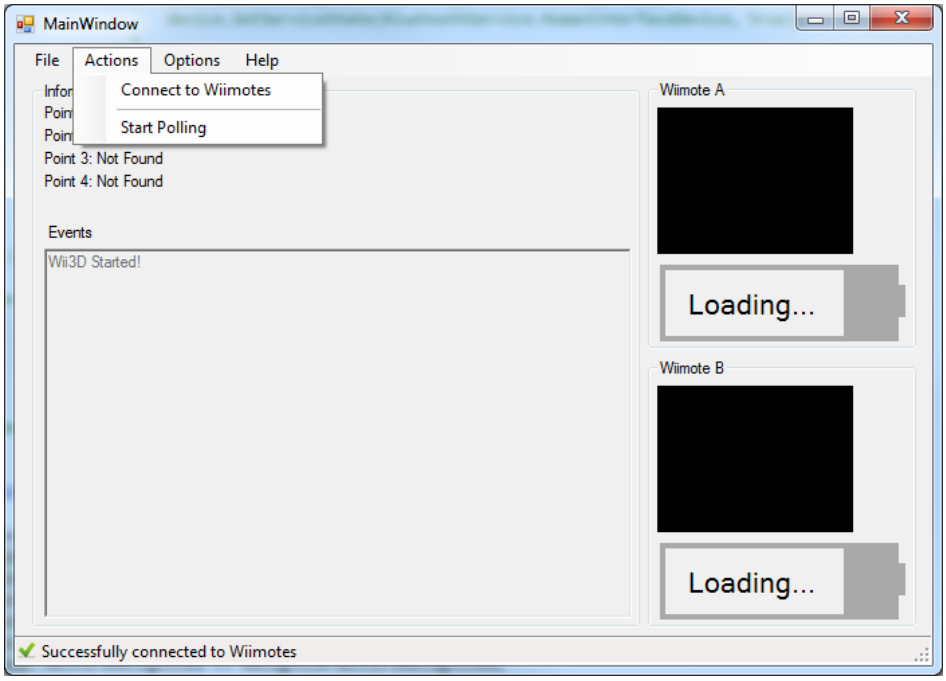


Figure 3.21: Start Polling

Once the polling is started, the Wii3D system will display the battery status for each of the Wii Remotes, alongside the two dimensional points reported by the devices, as shown in Figure 3.22. The three dimensional points that are triangulated using the stereoscopic triangulation algorithms described in Chapter 3.2.5.1 are displayed.

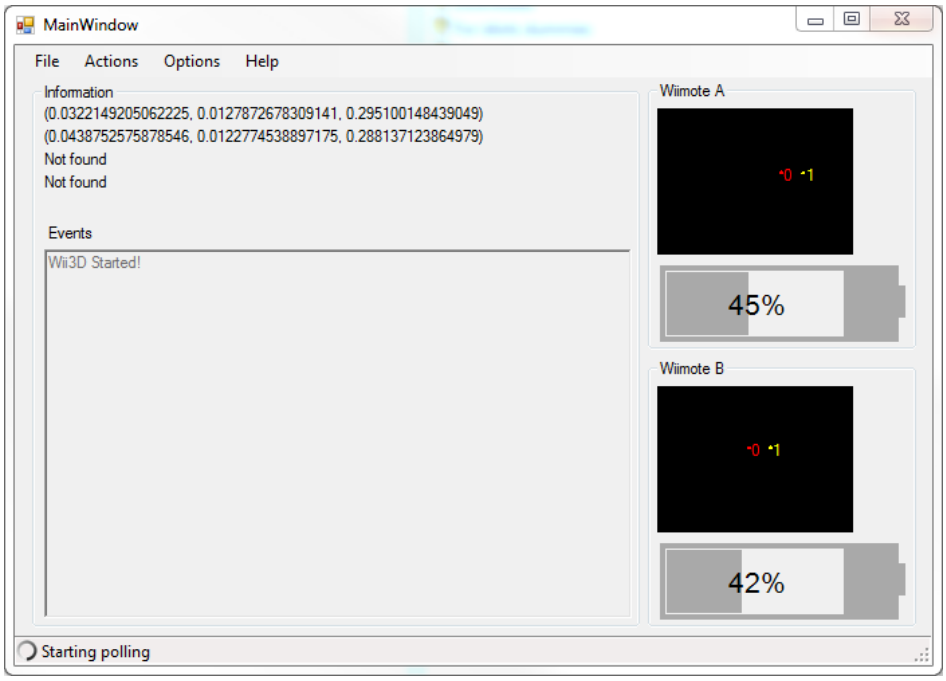


Figure 3.22: Polling

Any gestures that are recognized are displayed in the 'Events' box, as shown in Figure 3.23.

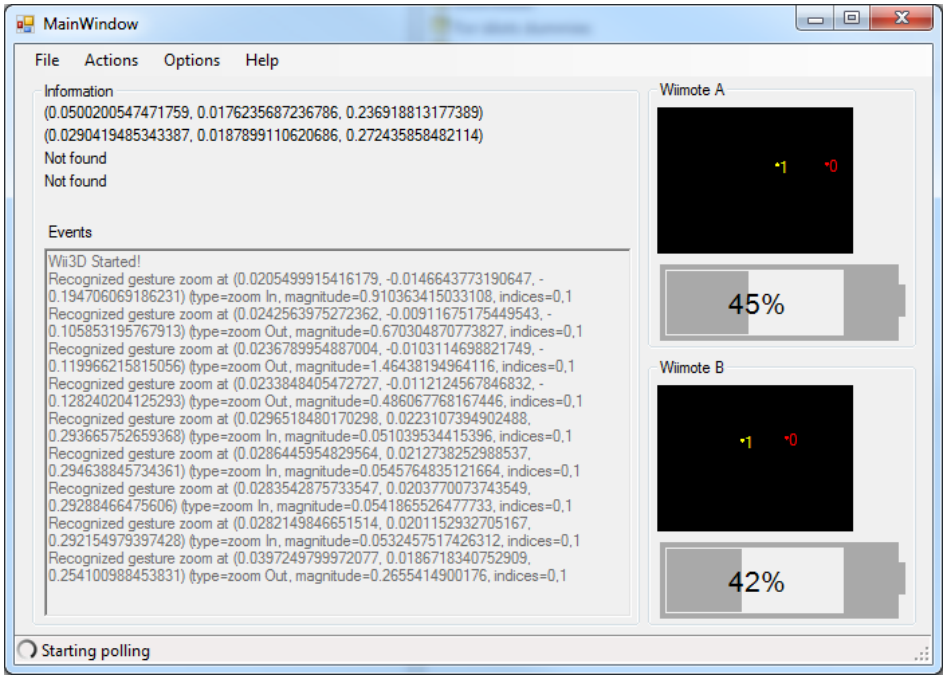


Figure 3.23: Events

3.2.8.4 State Transitions

The user interface has several states and transitions, shown in Figure 3.24.

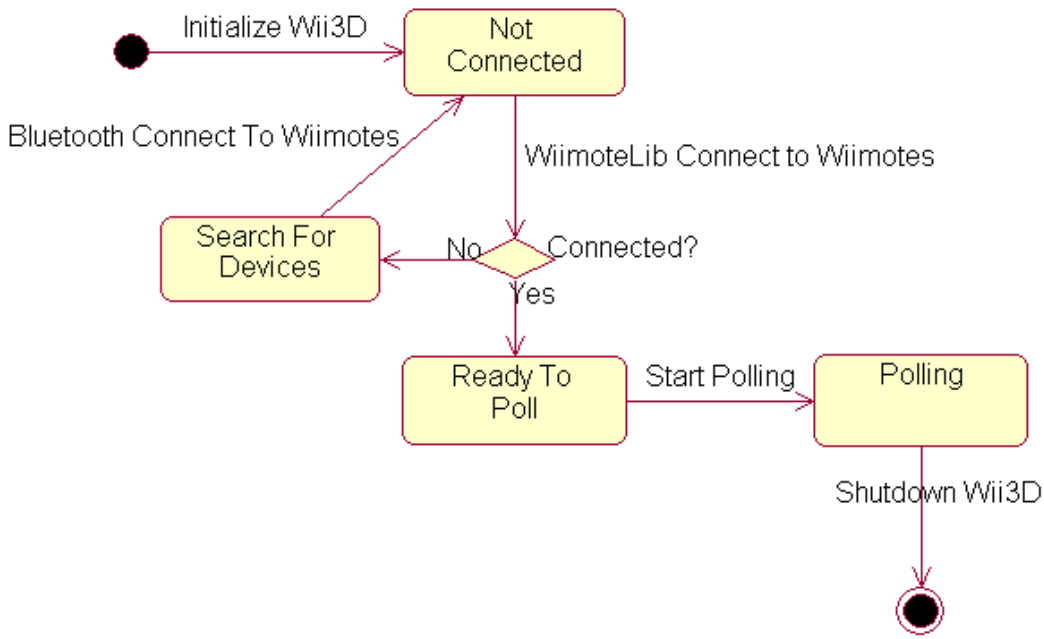


Figure 3.24: User Interface State Transition

3.3 Summary

Once the Wii3D System hardware is set up, by placing two Nintendo Wii Remotes next to each other with their lines of sight parallel, the user wears the WiiGlove or WiiGloves, depending on the chosen configuration, and then turns the Wii Remotes on.

The Wii3D System functionality is achieved in several steps. Once the Wii Remotes are connected, they are queried at regular intervals for their two dimensional viewport coordinates, which are then used to resolve their three dimensional coordinates, which are grouped into pairs by matching points from each viewport using distance minimization. The resulting three dimensional points are matched to their paths using predictive interpolation, and the resulting paths are inspected by the gesture recognizer to see if any gestures match the paths.

Chapter 4

User Study

The goal of the user study was to evaluate the intuitiveness of the interaction technique by a comparison with existing techniques and devices.

4.1 Goal of the Experiment

The experiment has two goals:

- To investigate whether the Wii3D System is a useful system
- To investigate whether the Wii3D System could be used as an alternative to the mouse and keyboard

4.2 Design and Methodology

The 20 participants selected for the study were approached and asked if they would be interested in testing the Wii3D System. The participants were sampled from the Rhodes University student body, and their consent forms and questionnaires were destroyed once their data had been captured. None of the captured records were linked to the individual student.

Once the participant agreed to be a part of the research, he/she was allocated a time slot of 15 minutes where he/she would be expected to carry out several tasks. Each task was

supervised by the principal researcher, and, upon completion of the task, a short section of the questionnaire was answered.

The proposed design and methodology for this user study was passed by the Rhodes University Ethical Standards Committee.

4.2.1 Consent

In order to take part in the user study, a participant was required to complete the consent form, shown in Appendix A, Figure A.1.

4.2.2 Introductory Questions

The users were asked two questions before any tasks were started:

1. How long have you been using a mouse and keyboard for?
2. How often do you use a mouse and keyboard?

These questions were asked in order to gain some insight into the possible groupings of the different participants by their experience with the accepted mouse and keyboard method of interaction with a computer.

4.2.3 Tasks

Once the consent form was completed, the participant was given a brief outline of the Wii3D System and walked through the tasks that needed to be completed. None of the participants had any experience with the Wii3D System, and were required to have some experience with a mouse and keyboard. The users had to rate the intuitiveness of the methods used to complete each task, and could add optional comments if they chose.

4.2.3.1 Task 1 - Pointer Movement

The participants were asked to complete three sub-tasks for the pointer movement task, as shown in Appendix A in Figure A.2. These tasks were designed to make comparisons between mouse and keyboard and Wii3D System pointer movement. Both single and multiple pointer movement were investigated.

Single pointer movement was investigated by comparing the use of the mouse pointer with a single Wii3D System pointer, dual pointer movement by comparing the mouse and the keyboard combination "WASD" with two Wii3D System pointers, and, quadruple pointer movement by adding "TFGH" and "IJKL" with two additional Wii3D System pointers. The participants were required to move the pointers over targets as shown in Figure 4.1.

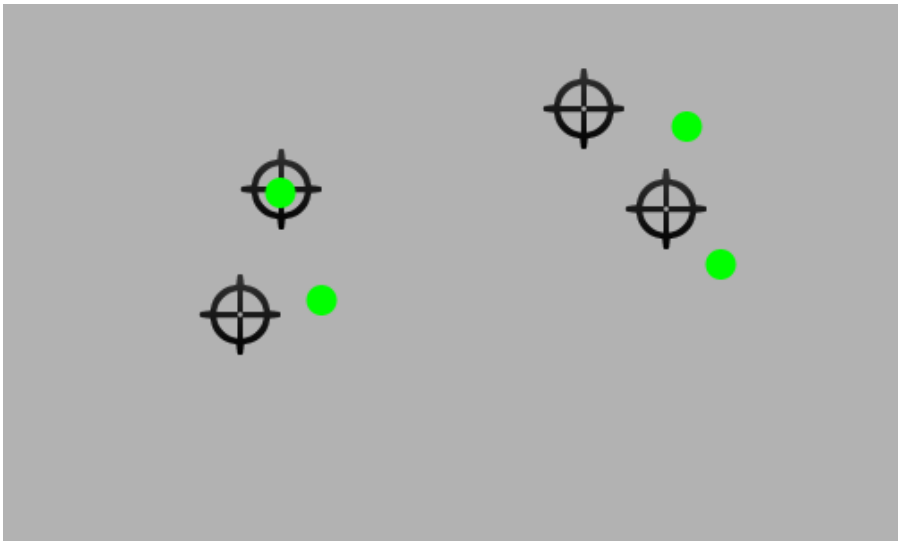


Figure 4.1: Pointer Movement Targets

4.2.3.2 Task 2 - Clicking

The participants were asked to complete two sub-tasks for the clicking task, as shown in Appendix A in Figure A.2. These tasks were designed to make comparisons between mouse and keyboard and Wii3D System clicking. Both single and double clicks were investigated.

Single pointer clicking was investigated by comparing the use of a mouse pointer and mouse left click with a single Wii3D System pointer and a click gesture, as described in Chapter 3.2.6.1. Dual clicking was investigated by comparing the mouse pointer and a

mouse left click, the keyboard combination "WASD" and the "Enter" key with two Wii3D System pointers and their respective click gestures. As in the movement task, participants were required to move the pointers over the targets, and then make the click gesture.

4.2.3.3 Task 3 - Panning/Scrolling

The participants were asked to complete two sub-tasks for the panning/scrolling task, as shown in Appendix A in Figure A.2. These tasks were designed to make comparisons between mouse and Wii3D System scrolling.

The users were required to scroll right and then left to pan/scroll an image to the right and left - they used the mouse's left/right scroller and the Wii3D System's pan/scroll gestures described in Chapter 3.2.6.2. The participants panned/scrolled the image as shown in Figure 4.2.

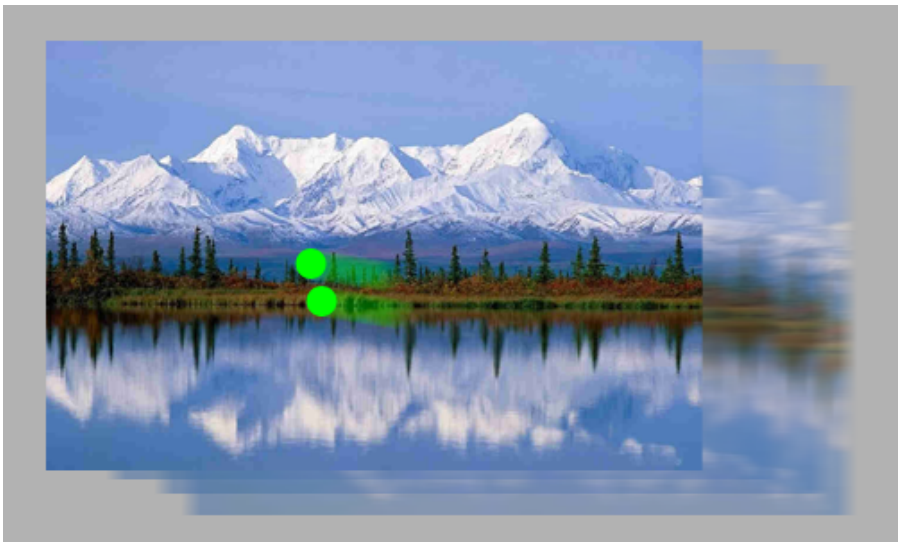


Figure 4.2: Panning/Scrolling an Image

4.2.3.4 Task 4 - Zooming

The participants were asked to complete two sub-tasks for the zooming task, as shown in Appendix A in Figure A.3. These tasks were designed to make comparisons between mouse and Wii3D System zooming.

The users were required to zoom in and out of an image - they used the mouse's up/-down scroller and the Wii3D System's zoom gestures described in Chapter 3.2.6.3. The participants zoomed the image in and out as shown in Figure 4.3.

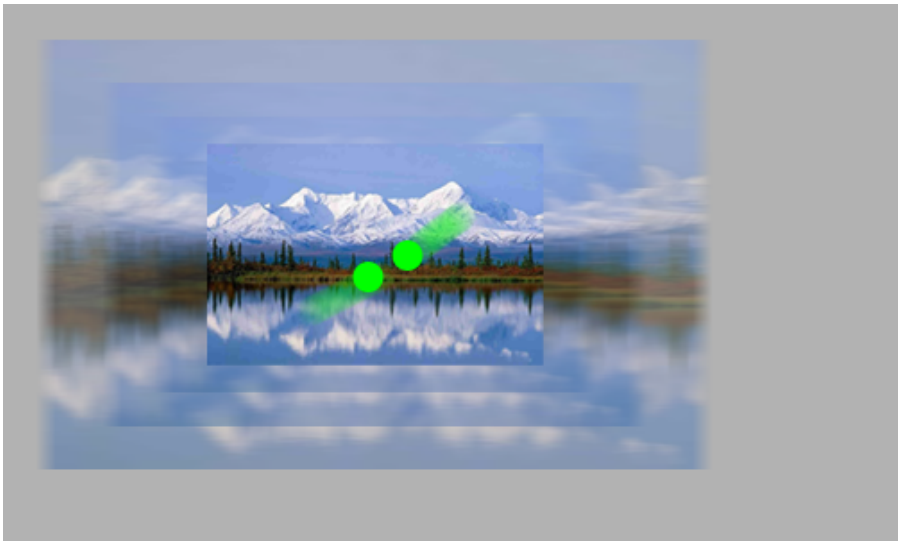


Figure 4.3: Zooming an Image

4.2.3.5 Task 5 - Rotating

The participants were asked to complete four sub-tasks for the rotating task, as shown in Appendix A in Figure A.2. These tasks were designed to make comparisons between mouse and Wii3D System rotating.

The users were required to rotate an image to the clockwise and counter-clockwise about the Z and X axes - they used the image's handles, mouse motion and clicks and the Wii3D System's rotating gestures described in Chapter 3.2.6.4. The participants rotated the image as shown in Figures 4.4 and 4.5.

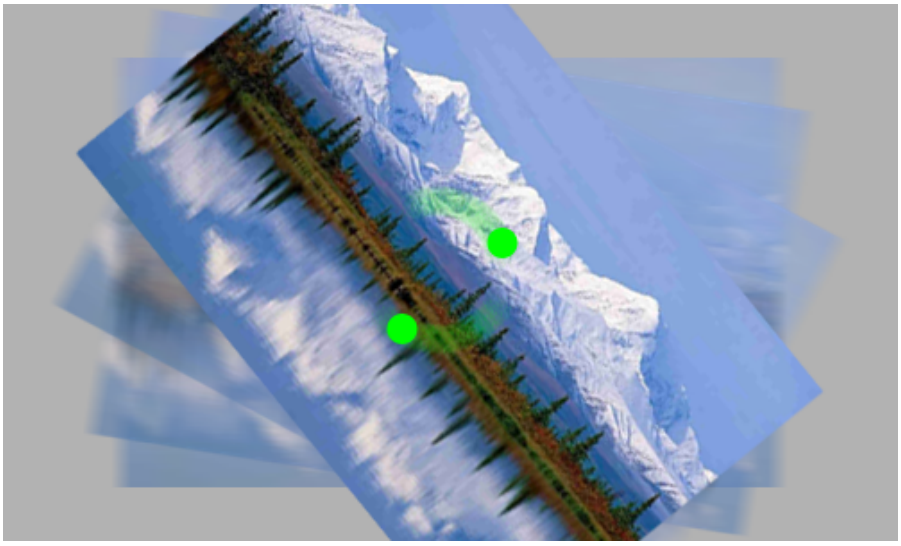


Figure 4.4: Rotating an Image About the Z Axis

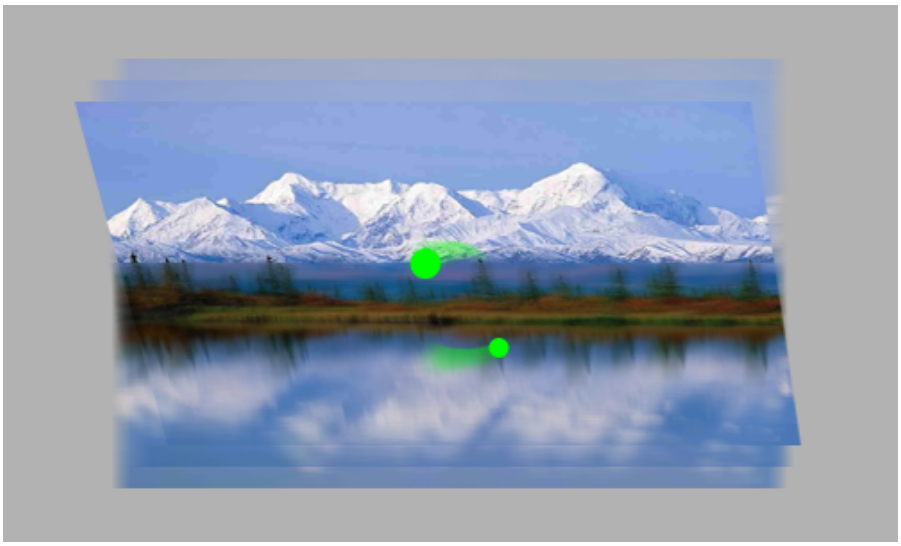


Figure 4.5: Rotating an Image About the X Axis

4.2.3.6 Task 6 - Complex Gesture and 3D Interaction

The participants were asked to complete two sub-tasks for the complex gesture and 3D interaction task, as shown in Appendix A in Figure A.2. These tasks were designed to gauge the efficiency and accuracy of the Hidden Markov Model and the 3D interaction effectiveness.

The users were required to draw two circles - one that was parallel to the screen's plane

and one that was perpendicular to the screen's plane - using a single Wii3D System pointer.

4.2.4 Overview Questions

Once the tasks were completed and the relevant questionnaire sections were filled out, the participants were required to answer several overview questions:

1. Would you use the Wii3D System as an alternative to the mouse and keyboard? Why or why not?
2. If no, do you believe that the Wii3D System would be useful to someone else? Why?
3. Would you add/remove/change anything in the Wii3D System? Why?
4. You have undoubtedly had more experience using a mouse and keyboard than this type of system. What tasks would you find Wii3D more suitable for than a mouse and keyboard?

These questions were to gain a feel for what the participants thought of the Wii3D System in general, and whether they thought that this kind of system would be useful to themselves or others.

4.3 Results

4.3.1 Preliminary Observations

There were several issues that were observed during the experiment:

- Confusion between rotation and zoom gestures
- People are more used to mouse and keyboard
- Hidden Markov Models not performing efficiently
- Hidden Markov Models only successfully recognizing a circle gesture 50% of the time

4.3.2 Study's Principal Results

All of the participants of this user study stated that they would not use the Wii3D System as the primary input device for a computer. There were several primary reasons given for these negative responses:

- Lack of typing support
- Gestures are not always recognized
- Accuracy of the Wii3D System pointers was not as good as a mouse

The majority of the participants stated that they thought that the Wii3D System would be useful to someone else, or that there were specific tasks for which the Wii3D System would be useful. Each participant provided the same answer for Question O.2 and O.4, therefore requiring only one graph, shown in Figure 4.6. The Wii3D System was found to be statistically useful ($\chi^2 = 6.05$, $df = 1$, $p = 0.01391$).

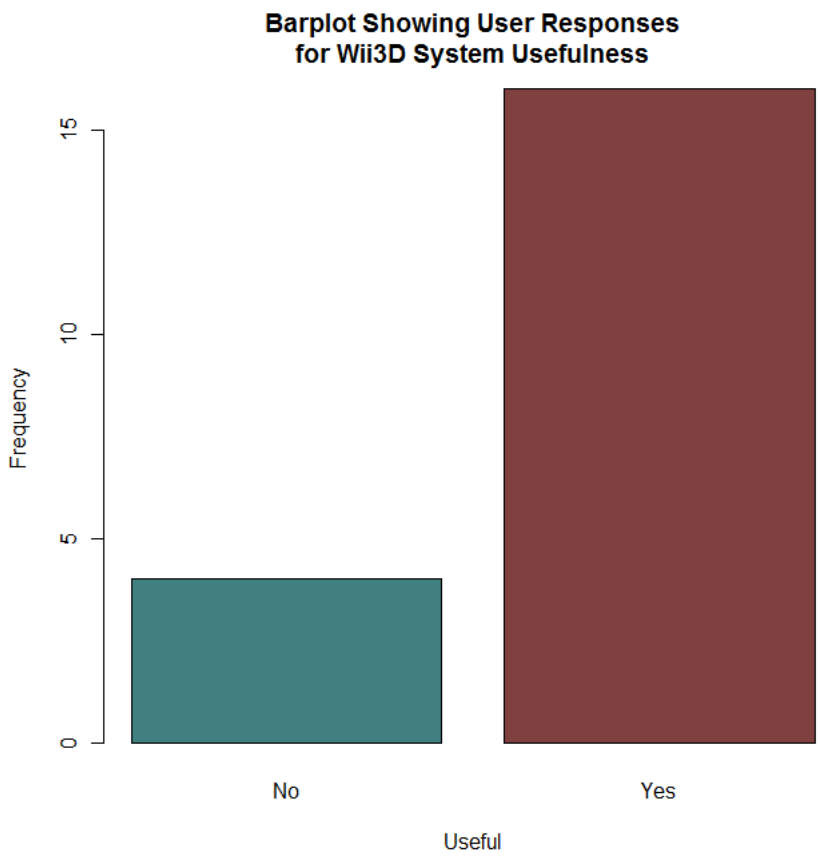


Figure 4.6: Wii3D Usefulness

The statistical tests that follow rate the intuitiveness of the proposed gesture by grouping the results into two sections:

- Very unintuitive, unintuitive and average
- Intuitive and very unintuitive

The tests measure the significance of differences in proportions between the two methods of input with levels of significance set at 5%.

4.3.2.1 Task 1 - Pointer Movement

The barplots shown in Figure 4.7 show that the participants find the single pointer movement more intuitive when using the mouse than when using the Wii3D System ($\chi^2 = 3.6571$, $df = 1$, $p = 0.02791$). It also shows that the participants find that there was no significant difference between dual pointer movement ($\chi^2 = 0.9375$, $df = 1$, $p = 0.3329$). Quadruple pointer movement appears to be more intuitive when using the Wii3D System than when using the mouse and keyboard combination, although the statistical test shows that the difference is not significant ($\chi^2 = 0.5263$, $df = 1$, $p = 0.2341$). However, if the grouping of the results is changed to be average or above performance as acceptable, the Wii3D System is found to be more intuitive ($\chi^2 = 16.4103$, $df = 1$, $p = 2.550 \times 10^{-5}$).

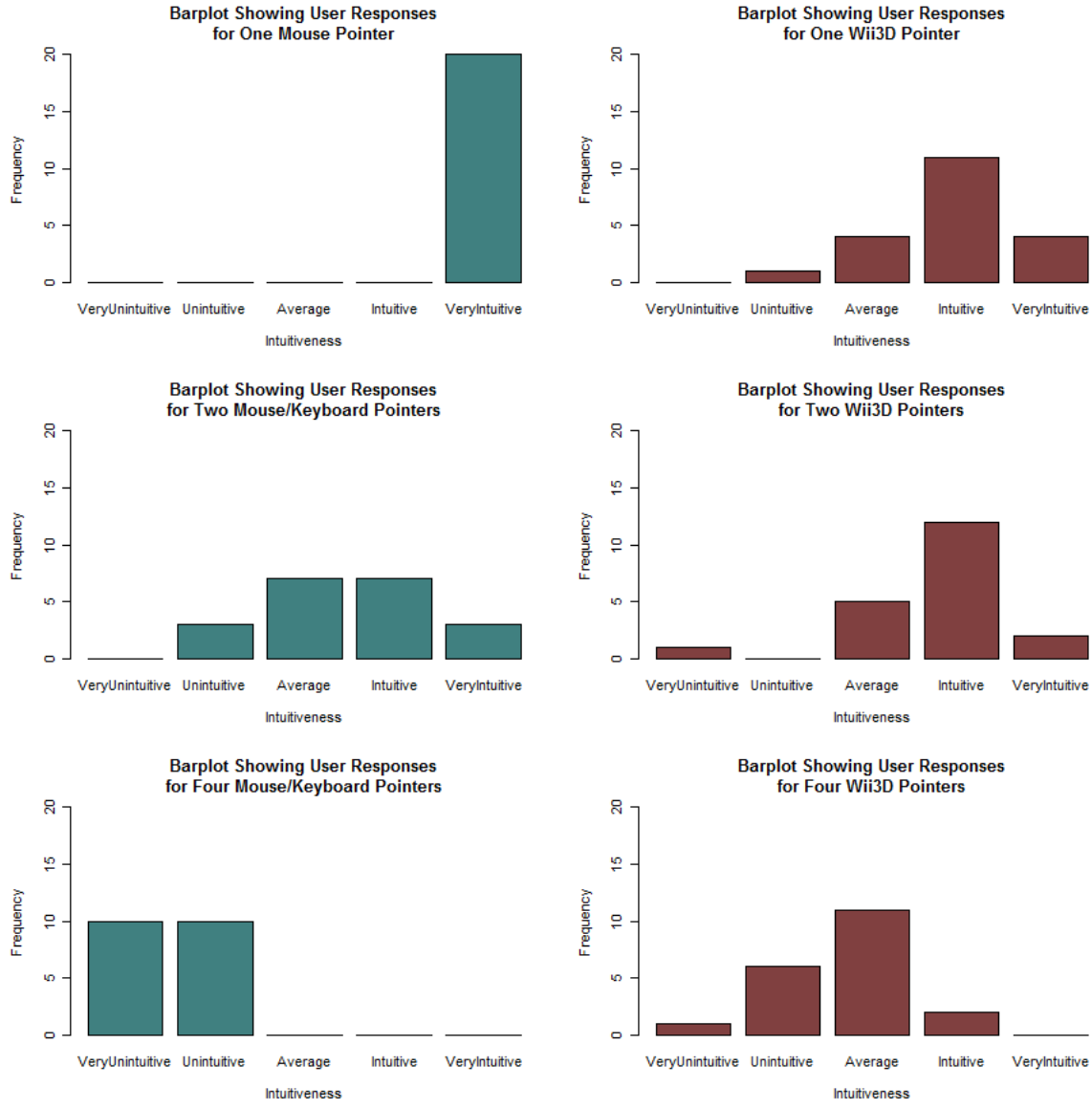


Figure 4.7: Movement Results

4.3.2.2 Task 2 - Clicking

Figure 4.8 shows that a single click was more intuitive with a mouse than with the Wii3D System ($\chi^2 = 4.902$, $df = 1$, $p = 0.01341$). When participants were faced with having to click multiple targets, the Wii3D System responses were found to be insignificantly different to the mouse and keyboard combination ($\chi^2 = 0$, $df = 1$, $p = 1$).

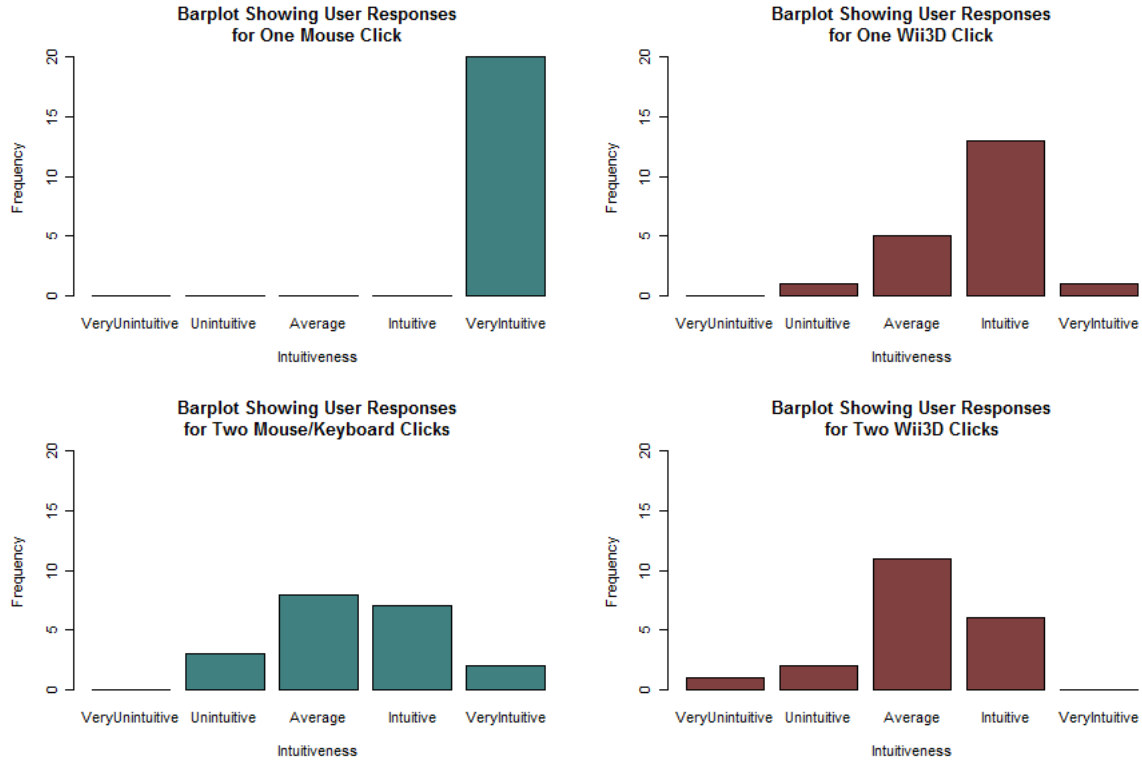


Figure 4.8: Clicking Results

4.3.2.3 Task 3 - Panning/Scrolling

The panning and scrolling preferred method was the mouse, as shown in Figure 4.9 ($\chi^2 = 9.1756$, $df = 1$, $p = 0.001226$). The Wii3D results, however, were still positive as the majority of the responses rated average or above.

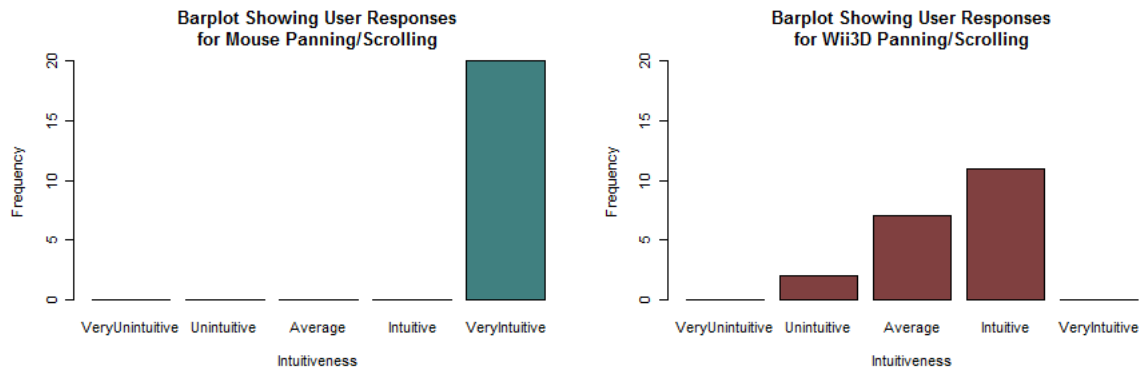


Figure 4.9: Panning/Scrolling Results

4.3.2.4 Task 4 - Zooming

The participants found this gesture the most intuitive out of all of the gestures. There was no significant difference between the mouse technique and the Wii3D System method ($\chi^2 = 1.4414$, $df = 1$, $p = 0.2299$), all of the observations were average or above, as reflected in Figure 4.10.

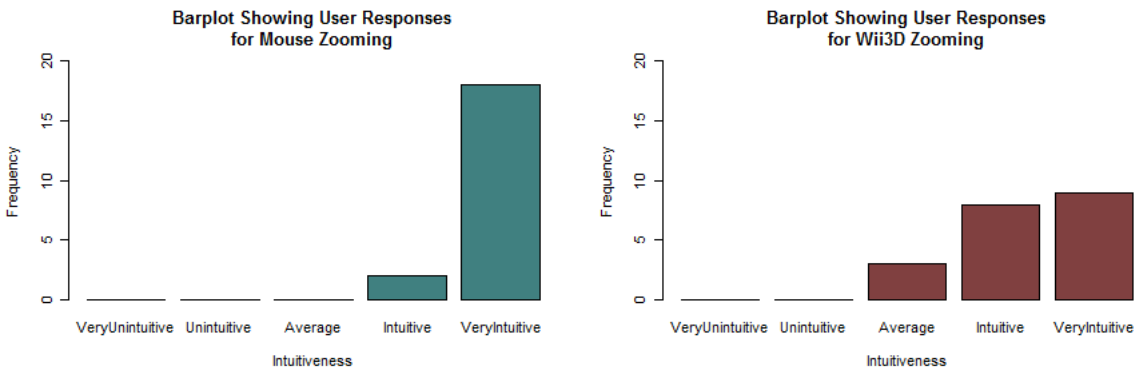


Figure 4.10: Zooming Results

4.3.2.5 Task 5 - Rotating

The feedback from the participants for the rotation tasks, as shown in Figure 4.11, suggest that the mouse input method is slightly more intuitive than the Wii3D System rotation gestures in two dimensions ($\chi^2 = 12.6042$, $df = 1$, $p = 0.0001924$), but insignificantly different in three dimensions ($\chi^2 = 0$, $df = 1$, $p = 1$). This could be attributed to several participants' rotational gestures being recognized as zoom gestures instead.

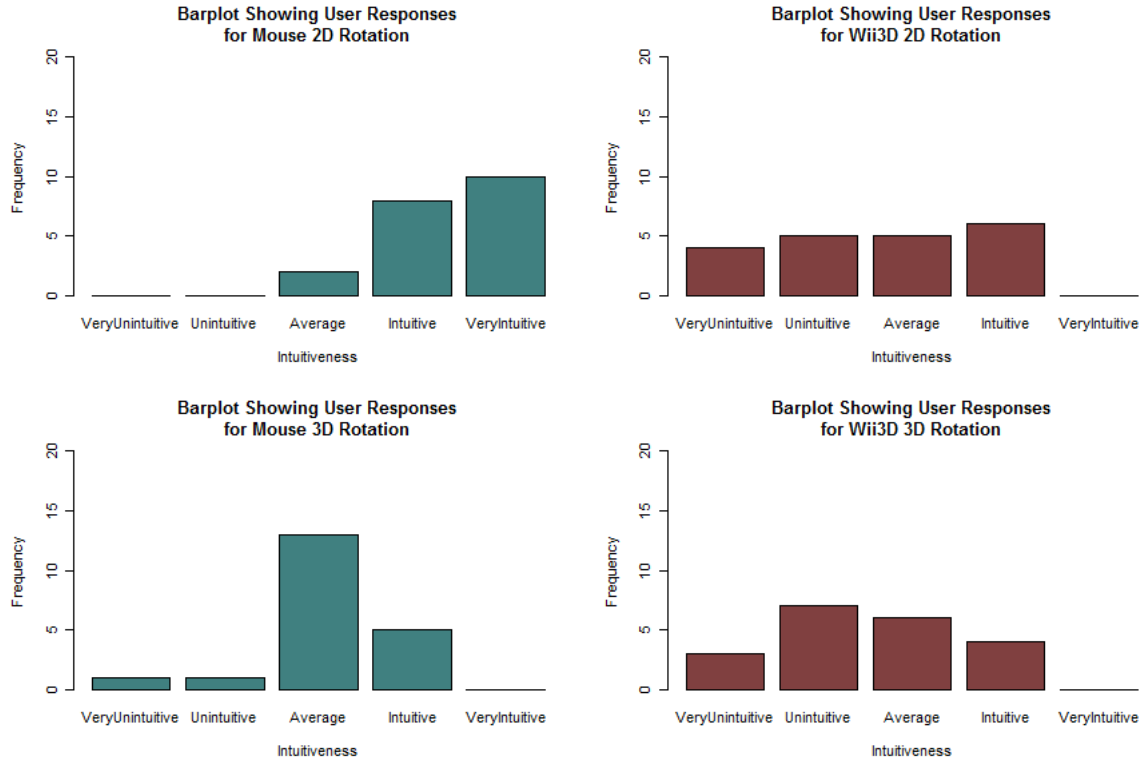


Figure 4.11: Rotating Results

4.3.2.6 Task 6 - Complex Gesture and 3D Interaction

Figure 4.12 below appears to show a very discouraging result - the users found that the complex gestures are unintuitive ($\chi^2 = 6.05$, $df = 1$, $p = 0.01391$). Unfortunately, the circle gestures were recognized using Hidden Markov Models, and the gesture was correctly identified from the model 50% of the time. This result is reflected in the ten participants who rated this task as "Very Unintuitive". If these results were removed, the results would show that the technique is more intuitive than the graph in Figure 4.12 suggests ($\chi^2 = 0.1$, $df = 1$, $p = 0.7518$).

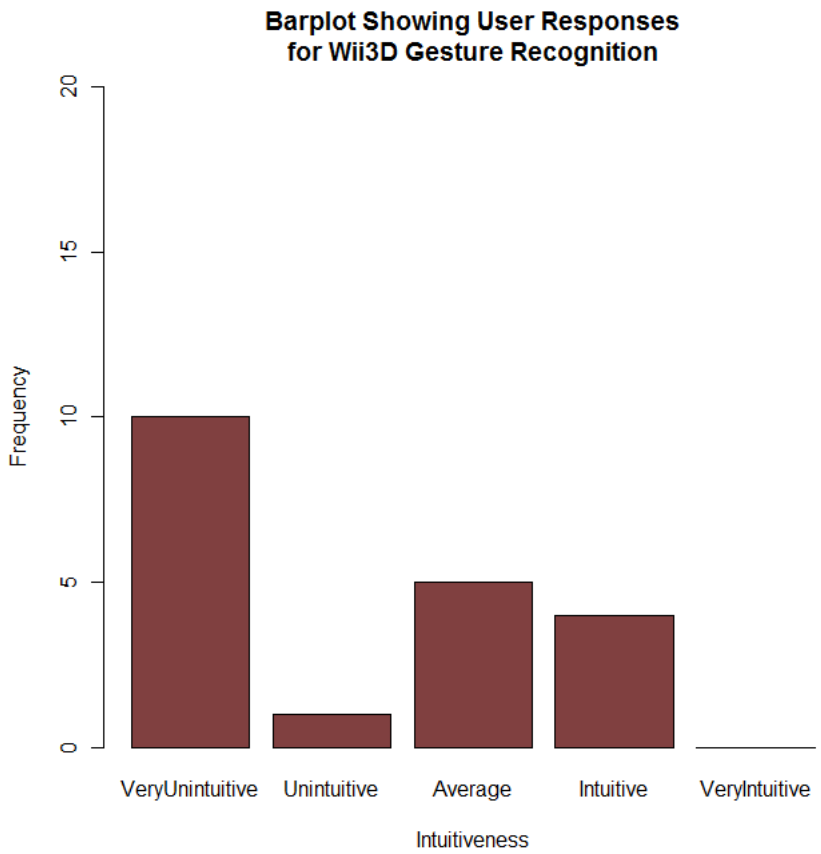


Figure 4.12: Complex Gesture and 3D Interaction Results

4.3.3 Discussion

The general user position was that the Wii3D System would not work as a replacement for the standard mouse and keyboard. One of the participants wrote, "It's a really nice program that I could get used to! How would I type letters if I had to replace my keyboard?", and another user went so far as to comment, "This system might as well be a gimmick that will be marketed for gaming consoles and will go out of fashion".

The majority of the participants felt that the Wii3D System would be useful for some applications - one of the users said, "Wii3D would work well if some two handed gestures were made for driving games. Holding your hands up and moving them to steer the car and changing gears would be fun".

The implemented point tracker was found to fail in some situations. In extreme circumstances, such as when two infra-red sources merge into one, and one of them then leaves

the field of view without being detected, the tracker can be fooled into detecting multiple points when there should only be one.

The participants' feedback yielded two main areas that required attention:

- The users thought that the Wii3D System would be more useful for gaming and other three dimensional interactions
- The users want typing functionality

Chapter 5

Conclusions and Future Work

5.1 Improvements

There are several improvements and optimizations that have been identified over the course of the investigation.

5.1.1 Multiple Wii Remotes

The current configuration of the Wii3D System only uses two Wii Remotes. Additional Wii Remotes could be used to increase the accuracy of the stereoscopic triangulation. This would also aid in dealing with the loss of points for the different perspectives.

5.1.2 Point Tracker

The point tracker described in Chapter 3.2.5 can be modified to incorporate acceleration and factor this observed property into the predictive interpolation. The velocities and accelerations of the two dimensional viewports may also be used in the calculation.

The development of a point tracker had some success, but the tracker can fail in some situations. It is highly probable that there are several other situations where the predictive interpolator could be confused, and more investigation into this component would be prudent before incorporating this system into a general user interface.

5.1.3 Dynamic Hidden Markov Models

The Hidden Markov Models implemented in the Wii3D System are generated at application start-up. Extending the functionality of the system to incorporate object serialization and learning to the Models would allow the user to define his/her own gestures by example, rather than relying solely on the hard-coded gestures which require programming experience and a good understanding of Hidden Markov Models.

The Accord.NET library uses discrete Hidden Markov Models. Due to the continuous nature of the point coordinates in a gesture, a custom model could be developed to incorporate Gaussian mixture models[30, 22] in the Markov chains to form a Hidden Markov Model.

5.2 Additions

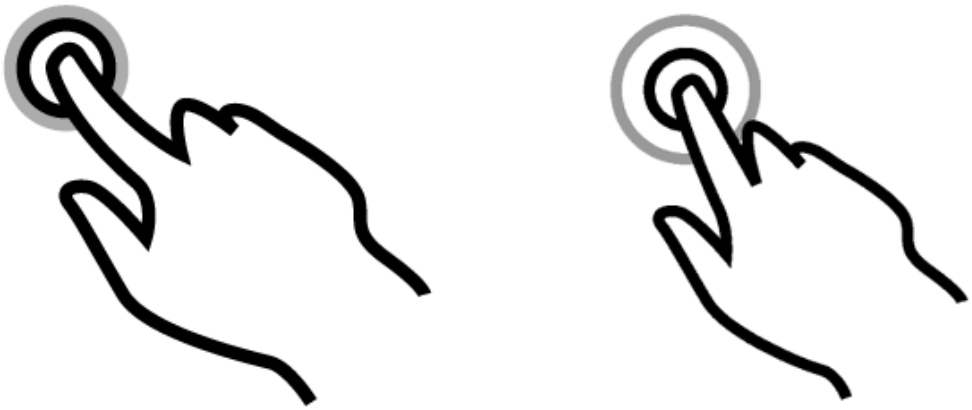
5.2.1 Gestures

There are several gestures that could be used in addition to those already defined.

- Double Click
- Flick
- Scaling an Object

The images for the gestures are sourced from GestureWorks[11].

The double-click gesture could be used in the same way that a mouse double click is used.



Click 1 Click 2

Figure 5.1: Double Click

The flick gesture could be used for page turns, and would be especially useful for document readers.

One Finger Flick Two Finger Flick

Figure 5.2: Swipe

A gesture to scale selected objects would be useful for 3D vector based programs such as Autodesk Maya or 3DStudio Max. The gesture could be defined as placing two fingers on an object and moving third towards or away from this pivot to scale the object using the third finger's direction vector.

Scale X Scale Y

Figure 5.3: Scaling an Object

5.2.2 Typing

A typing interface could be used using the methods described in [16]. In order to create the necessary two dimensional gesture from the three dimensional paths that the Wii3D System uses, there are several techniques that could be used:

- A linear regression two decompose the three dimensional gesture into a closest fit two dimensional plane
- A removal of one of the principal dimensions (e.g. the Z plane)
- The use of a single Wii Remote's two dimensional coordinates

The linear regression would allow the user to make letter gestures in any orientation because the adjustment of the axes to become parallel to the gesture's plane and the subsequent reduction of one of the unused principal axes results in the best fit two dimensional gesture. Each letter gesture would need to be made such that it fits inside an approximate plane. The decision whether to use the gesture as a letter gesture could be ascertained by the statistical significance of the regression.

The removal of one the axes would limit a user to making gestures such that each letter gesture would fit inside the XY plane. Similarly, the use of only one of the Wii Remote's

reported coordinates would limit the gesture such that its containing plane is almost perpendicular to that particular Wii Remote's line of sight.

5.2.3 Gesture Recognition Methods

The Wii3D System makes use of Finite State Automata and Hidden Markov Models for gesture recognition. Artificial neural networks could be implemented in the system to add another dimension to the recognition.

5.3 Final Thoughts

The investigation into the current state of human computer interaction, especially the vision based and three dimensional interaction, suggested an implementation of a three dimensional interaction technique that is easily accessible to the general public. The Nintendo Wii Remotes fill this task, and the components for gloves designed are easily sourced and built. Although the Wii3D System would not be suitable as a replacement for the traditional mouse and keyboard interaction, it would work well as an alternative input device for specialized applications, especially for gaming and other three dimensional interactive applications. The use of the Wii3D System could work well as a replacement of the default mouse pointing system, rather than the replacement of the both the mouse and keyboard.

Bibliography

[1] P. Badin and A. Serrurier. Three dimensional linear modeling of tongue: Articulatory data and models. In D. Demolin H. C. Yehia and R. Laboissière, editors, *Proceedings of the 7th International Seminar on Speech Production*, pages 395–402. ISSP7, 2006.

[2] C. M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, 1995.

[3] Matt Casamassina. Revolution's horsepower. Website, March 2006.

[4] Michael Chen, S. Joy Mountford, and Abigail Sellen. A study in interactive 3-d rotation using 2-d control devices. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, volume 22, pages 121–129, New York, NY, USA, 1988. ACM.

[5] Hiroki Eda. Ps3 vs wii, comparisons of core lsi chip areas. Website, November 2006.

[6] S. Wise et al. Evaluation of a fiber optic glove for semi-automated goniometric measurements. *Journal of Rehabilitation Research an Development*, 27(4):411–424, 1990.

[7] T. G. Zimmerman et al. A hand gesture interface device. In *Proceedings of Human Factors in COmputing Systems and Graphics Interface*, pages 189–192, New York, April 1987. ACM Press.

[8] Peter Foot. 32feet.net - personal area networking for .net. Website, July 2010.

[9] P. Frosini. Measuring shapes by size functions. In *Proceedings of SPIE on Intelligent Robotics and Computer Vision*, volume 1607, pages 3–26, Boston, Mass., 1991.

[10] K. S. Fu and T. S. Yu. *Statistical Pattern Classication using Contextual Information.* Recognition and Image Processing Series. Research Studies Press, 1980.

[11] GestureWorks. Gestureworks supported gestures. Website, October 2010.

[12] A. S. Gilinsky. Perceived size and distance in visual space. *Psychological Review*, 58:460–482, 1951.

[13] Gunnar Rutger Grape. Model based (intermediate-level) computer vision. Technical Report AD0763673, Stanford University, May 1973.

[14] G. J. Grimes. Digital data entry glove interface device, November 1983.

[15] G.Tziritas. Estimation of motion and structure of 3d objects from a sequence of images. In *1st IEEE Conference on Computer Vision*, pages 693–697, London, June 1987.

[16] J. Y. Ha, S. C. Oh, J. H. Kim, and Y. B. Kwon. Unconstrained handwritten word recognition with interconnected hidden markov models. *IWFHR*, 3:455–460, May 1993.

[17] Marsha Jo Hannah. Computer matching of areas in stereo images. Technical Report AD0786720, University of Stanford, July 1974.

[18] Richard I. Hartley and Peter Sturm. Triangulation. *Computer Vision and Image Understanding*, 68(2):146–157, November 1997.

[19] Card Carey Gasen Mantei Perlman Strong Hewett, Baecker and Verplank. Acm sigchi curricula for human-computer interaction. Website, July 2009.

[20] J. W. Philbeck S. S. Fukusima J. M. Loomis, J. A. Da Silva. Visual perception of location and distance. *Current Directions in Psychological Science*, 5:72–77, 1996.

[21] Yangsheng Xu Jie Yang. Hidden markov model for gesture recognition. Technical Report CMU-RI-TR-94-10, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, May 1994.

[22] In-Cheol Kim and Sung-Il Chien. Analysis of 3d hand trajectory gestures using stroke-based composite hidden markov models. *Applied Intelligence*, 15(2):131–143, 2001.

[23] M. W. Krueger. *Artificial Reality*. Addison-Wesley, 2nd edition, 1990.

[24] Johnny Chung Lee. Johnny chung lee - projects - wii. Website, February 2009.

[25] D. Ziou M. Handouyahia and S. Wang. Sign language recognition using moment-based size functions. In *Proceedings of the International Conference on Vision Interfaces*, pages 210–216. Ker, CRC Press, 1999.

[26] D. Murray and A. Basu. Motion tracking with an active camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16:449–459, 1994.

[27] Nintendo. *Wii operations manual.* Nintendo.

[28] CSharp Online.NET. Csharp delegates and events - asynchronous method calls. Website, June 2006.

[29] Brian Peek. Managed library for nintendo's wiimote. Website, July 2009.

[30] Valery A. Pterushin. Hidden markov models: Fundamentals and applications (part 2 - discrete and continuous hidden markov models). *Online Symposium for Electronic Engineers*, 1:1, 2000.

[31] Lawrence R. Rabiner. *Readings in Speech Recognition*, chapter 18, pages 267–296. Morgan Kaufmann Publishers Inc., 1990.

[32] David J. Sturman and David Zeltzer. A survey of glove-based input. *IEEE Computer Graphics and Applications*, 14(1):30–39, January/February 1994.

[33] Clark Thompson. Depth perception in stereo computer vision. Technical Report STAN-CS-7 5-5 2 1, Stanford University, 1975.

[34] Eric Haines Tomas Möller and Naty Hoffman. *Realtime Rendering, Third Edition.* A. K. Peters, Ltd., Natick, MA, USA, third edition, 2008.

[35] G. Tziritas. Displacement estimation for image predictive coding and frame motion-adaptive interpolation. In *Proceedings of SPIE Visual Communications and Image Processing*, pages 936–941, 1988.

[36] Richard Watson. A survey of gesture recognition techniques. Technical Report TCD-CS-93-11, Trinity College, Dublin, July 1993.

[37] P. Wellner. Interacting with paper on the digital desk. *Communications of the ACM*, 36(7):87–96, July 1993.

[38] C. Wheatstone. XXXVI. Contributions to the physiology of vision. Part the First. On the some remarkable, and hitherto unobserved, phænomena of binocular vision. *Philosophical Magazine Series 4*, 3(18):241–267, 1852.

[39] Wiire.org. Main page - wiire. Website, May 2007.

[40] Wikipedia. Artificial neural network. Website, October 2010.

[41] Wikipedia. Hidden markov models. Website, October 2010.

[42] Wikipedia. Triangulation. Website, October 2010.

[43] Shu-Fai Wong and Roberto Cipolla. Continuous gesture recognition using a sparse bayesian classifier. In *ICPR '06: Proceedings of the 18th International Conference on Pattern Recognition*, pages 1084–1087, Washington, DC, USA, 2006. IEEE Computer Society.

[44] Michal Piotr Wronski. Design and implementation of a hand tracking interface using the nintendo wii remote. Master's thesis, University of Cape Town, 2008.

[45] Robert C. Zeleznik, Andrew S. Forsberg, and Paul S. Strauss. Two pointer input for 3d interaction. In *Proceedings of the 1997 Symposium on Interactive 3D Graphics*, pages 115–120, 1997.

Appendix A

User Study

Each participant was required to fill out a consent form, shown in Figure A.1 before carrying out the user study tasks, shown in Figures A.2 and A.3. Each participant was required to fill out a questionnaire, shown in Figures A.4, A.5 and A.6 after each of the tasks was carried out.

Wii3D User Study | 2010

Consent Form

I have read and/or I understand the contents of the information sheet and understand that I have been invited to participate, that my agreeing is fully voluntary, and that I can withdraw at any time.

Name (Will be blacked out):	Signed: Date:__/__/2010
Witness: João Lourenço	Signed: Date:__/__/2010

1 of 6 | Page

Figure A.1: User Study Consent Form

Wii3D User Study – Instruction Sheet

Welcome to the Wii3D user study! Thank you very much for your cooperation in my research.

Task 1 – Moving the Pointers

Mouse and Keyboard

1. There is a single target on the screen. Move one of the pointers over the target with the mouse.
2. There are two targets on the screen. Move one of the pointers over a target with the mouse, and the other pointer over the other target with the keyboard (WASD).
3. There are four targets on the screen. Move a pointer over each of these targets (mouse, WASD, TFGH IJKL).

Wii3D System

4. There is a single target on the screen. Move one of the pointers over this target.
5. There are two targets on the screen. Move a pointer over each of these targets.
6. There are four targets on the screen. Move a pointer over each of these targets.

Please fill out the section entitled "Task 1 – Moving the Pointers" in the questionnaire.

Task 2 – Clicking

Mouse and Keyboard

1. There is a single target on the screen. Click on it with the mouse.
2. There are two targets on the screen. Click on one of them with the mouse, and select the other by moving the other pointer with the keyboard and hitting enter. Hitting 'enter' and the mouse click should be at the same time.

Wii3D System

3. There is a single target on the screen. Click on it (move forward then backward).
4. There are two targets on the screen. Click on them at the same time.

Please fill out the section entitled "Task 2 – Clicking" in the questionnaire.

Task 3 – Panning/Scrolling

Mouse and Keyboard

1. Scroll right using the scroll-bar provided.
2. Scroll left using the scroll-bar provided.

Wii3D System

3. Move two fingers, together in similar directions, to scroll right.
4. Move two fingers, together in similar directions, to scroll left.

Please fill out the section entitled "Task 3 – Panning/Scrolling" in the questionnaire.

Figure A.2: User Study Tasks Page 1

Task 4 – Zooming

Mouse and Keyboard

1. There is an image on the screen. Zoom in using the mouse's scroll.
2. Zoom out.

Wii3D System

3. There is an image on the screen. Zoom in (move fingers away from each other in opposite directions parallel to the screen).
4. Zoom out (move fingers towards each other).

Please fill out the section entitled "Task 4 – Zooming" in the questionnaire.

Task 5 – Rotating

Mouse and Keyboard

1. Rotate the image 90 degrees clockwise (click and hold the rotation handle and move the mouse).
2. Rotate the image 90 degrees counter-clockwise.
3. Rotate the image 90 degrees, top coming to the front (hold down 'alt' and click and hold the rotation handle and move the mouse).
4. Rotate the image back to the original position.

Wii3D System

5. Rotate the image 90 degrees clockwise (rotate fingers about a pivot).
6. Rotate the image 90 degrees counter-clockwise.
7. Rotate the image 90 degrees, top coming to the front.
8. Rotate the image back to the original position.

Please fill out the section entitled "Task 5 - Rotating" in the questionnaire.

Task 6 – Complex Gesture and 3D Interaction

1. Use one finger to draw a circle that is parallel to the screen's plane.
2. Use one finger to draw a circle that is perpendicular to the screen's plane.

Please fill out the section entitled "Task 6 – Complex Gesture and 3D Interaction" in the questionnaire.

Task 7 – Overview

Please take 5 minutes to think about whether you would use this system and fill out the section entitled "Task 7 – Overview" in the questionnaire.

Figure A.3: User Study Tasks Page 2

Questionnaire

- Please place an 'x' in the box that you choose.
- Please only give one option per question
- Please rate your answers from 1 (very unintuitive) to 5 (very intuitive)

Introductory Questions

How often do you use a mouse and keyboard?

How long have you been using a mouse and keyboard for?

Task 1 – Moving the Pointers

	Question	1	2	3	4	5
1.1	How did you find moving the single pointer with the mouse?					
1.2	How did you find moving the two pointers with the mouse and keyboard?					
1.3	How did you find moving the four pointers with the mouse and keyboard?					
1.4	How did you find moving the single pointer with the Wii3D System?					
1.5	How did you find moving the two pointers with the Wii3D System?					
1.6	How did you find moving the four pointers with the Wii3D System?					

Comments:

Task 2 – Clicking

	Question	1	2	3	4	5
2.1	How did you find clicking the target with the mouse?					
2.2	How did you find clicking the two targets with the mouse and keyboard?					
2.3	How did you find clicking the target with the Wii3D System?					
2.4	How did you find clicking the two targets with the Wii3D System?					

Comments:

Task 3 – Panning/Scrolling

	Question	1	2	3	4	5
3.1	How did you find panning and scrolling with the mouse?					
3.2	How did you find panning and scrolling using the Wii3D System?					

Comments:

Figure A.4: User Study Questionnaire Page 1

Task 4 – Zooming

	Question	1	2	3	4	5
4.1	How did you find zooming in and out with the mouse?					
4.2	How did you find zooming in and out with the Wii3D System?					

Comments:

Task 5 – Rotating

	Question	1	2	3	4	5
5.1	How did you find rotating the image with the mouse in 2D?					
5.2	How did you find rotating the image with the mouse in 3D?					
5.3	How did you find rotating the image with the Wii3D System in 2D?					
5.4	How did you find rotating the image with the Wii3D System in 3D?					

Comments:

Task 6 – Complex Gesture and 3D Interaction

	Question	1	2	3	4	5
6.1	How did you find making the gestures?					

Comments:

Task 7 – Overview

Would you use the Wii3D System as an alternative to the mouse and keyboard? Why or why not?

If no, do you believe that the Wii3D System would be useful to someone else? Why?

Figure A.5: User Study Questionnaire Page 2

Would you add/remove/change anything in the Wii3D System? Why?

You have undoubtedly had more experience using a mouse and keyboard than this type of system. What tasks would you find Wii3D more suitable for than a mouse and keyboard?

THE END

Thank you very much!

Figure A.6: User Study Questionnaire Page 3

Appendix B

Code Listings

Listing B.1: Stereoscopic Triangulation of a 3D Coordinate from Two 2D Coordinates

```
//Constants
private const double MaximumDistance = 0.01; //1cm

//Static variables
private static readonly double TanHoriz = Math.Tan(Math.PI *
    (41.0 / 180.0) / 2); //Horizontal FOV = 41 degrees
private static readonly double TanVert = Math.Tan(Math.PI *
    (31.0 / 180.0) / 2); //Vertical FOV = 31 degrees

public Point3D Stereo(Point2D a, Point2D b, Vector
    wiimoteAPosition, Vector wiimoteBPosition)
{
  if (a.Found && b.Found)
  {
    //Calculate the rays from the Wiimotes to the detected
        point
    var aRay = new Vector(new[] { a.X*TanHoriz, a.Y*TanVert,
        1 });
    var bRay = new Vector(new[] { b.X*TanHoriz, b.Y*TanVert,
        1 });

    //Calculate the matrix determinants
    var rayCrossProduct = aRay.CrossMultiply(bRay);
```

```
18      var matrixA = new Matrix(new double[][] {
           wiimoteBPosition - wiimoteAPosition, bRay,
           rayCrossProduct });
19      matrixA.Transpose();
20      var matrixB = new Matrix(new double[][] {
           wiimoteBPosition - wiimoteAPosition, bRay,
           rayCrossProduct });
21      matrixB.Transpose();
22
23      //Calculate estimate scale values for the rays
24      var crossLengthSquared = rayCrossProduct.SquaredNorm();
25      var sA = matrixA.Determinant()/crossLengthSquared;
26      var sB = matrixB.Determinant()/crossLengthSquared;
27
28      //Calculate the points along each ray
29      var rA = wiimoteAPosition + aRay*sA;
30      var rB = wiimoteBPosition + bRay*sB;
31
32      //Check the distance
33      var distance = (rA - rB).SquaredNorm();
34      if (distance <= MaximumDistance)
35      {
36        var point = (rA + rB) / 2;
37        return new Point3D(point[0], -point[1], -point[2]);
38      }
39    }
40    return Point3D.NotFound;
41  }
```

Listing B.2: Synchronous Program

```
1  class Program
2  {
3    public void DoTask()
4    {
5      //A time-consuming task
6      for (var i = 0; i < 100; i++)
7      {
```

```
8         Console.WriteLine(i);
9         Thread.Sleep(50);
10      }
11    }
12
13    public static void Main(string[] args)
14    {
15      var prog = new Program();
16      prog.DoTask();
17    }
18  }
```

Listing B.3: Asynchronous Invoker

```
1  public void DoAsync()
2  {
3    DoWorkerDelegate worker = DoWorker; //Create the worker
4    AsyncCallback completedCallback = DoCompletedCallback; //
       Create the async callback
5    //Lock the sync object to prepare for the async operation
6    lock (pDoSync)
7    {
8      if (DoIsBusy)
9      {
10       //If the async operation is already underway, throw an
           error
11       throw new InvalidOperationException("Currently
           performing 'Do' operation");
12     }
13
14     var async = System.ComponentModel.AsyncOperationManager.
         CreateOperation(null); //Create the async operation
15     var context = new DoAsyncContext(); //Create the async
         context
16     bool cancelled; //Create a variable to contain whether
         the operation has been cancelled
17     worker.BeginInvoke(async, context, out cancelled,
         completedCallback, async); //Start the worker
```

```
18      DoIsBusy = true; //Set the async operation to busy
19      pDoAsyncContext = context; //Set the async context to the
            created context
20    }
21  }
```

Listing B.4: IsBusy Property

```
1 public bool DoIsBusy { get; set; }
```

Listing B.5: DoCompleted Event

```
1 public event System.ComponentModel.AsyncCompletedEventHandler
      DoCompleted;
2
3 protected virtual void OnDoCompleted(System.ComponentModel.
     AsyncCompletedEventArgs e)
4 {
5   if (DoCompleted != null)
6     DoCompleted(this, e);
7 }
```

Listing B.6: DoProgressChanged Event

```
1 public event System.ComponentModel.AsyncCompletedEventHandler
      DoCompleted;
2
3 protected virtual void OnDoCompleted(System.ComponentModel.
     AsyncCompletedEventArgs e)
4 {
5   if (DoCompleted != null)
6     DoCompleted(this, e);
7 }
```

Listing B.7: CancelDoAsync Method

```
1 public void CancelDoAsync()
2 {
3   //Lock the sync object to prepare for the cancellation of
        the async operation
```

```
4   lock (pDoSync)
5   {
6     //Check if the context is not null
7     if (pDoAsyncContext != null)
8     {
9       //The operation exists, flag the operation for
          cancellation
10      pDoAsyncContext.Cancel();
11    }
12  }
13  //Set the async operation to not busy
14  DoIsBusy = false;
15 }
```

Listing B.8: DoWorker Method and DoWorkerDelegate

```
1 private void DoWorker(System.ComponentModel.AsyncOperation
    async, DoAsyncContext asyncContext, out bool cancelled)
2 {
3   cancelled = false; //Assume that the operation has not been
      cancelled
4   //TODO: Do action
5   for (var i = 0; i < 10; i++)
6   {
7     //Create the progress event arguments
8     var progressArgs = new System.ComponentModel.
        ProgressChangedEventArgs(
9       (int)((100 * i) / 10.0), //TODO: Calculate the
          percentage completion of the task
10      i //TODO: Pass an object back
11    );
12    Thread.Sleep(50); //Operation takes time
13    //Notify any subscribed listeners of the async operation'
        s progress change
14    async.Post(
15      e => OnDoProgressChanged((System.ComponentModel.
          ProgressChangedEventArgs)e),
16      progressArgs
```

```
17      );
18      //Check if the async operation has been flagged for
           cancellation
19      if (asyncContext.IsCancelling)
20      {
21        //Set the state of the async operation to cancelled
22        cancelled = true;
23        return;
24      }
25    }
26  }
27
28  private delegate void DoWorkerDelegate(System.ComponentModel.
       AsyncOperation async, DoAsyncContext asyncContext, out
       bool cancelled);
```

Listing B.9: DoCompletedCallback Method

```
1  private void DoCompletedCallback(IAsyncResult ar)
2  {
3    var worker = (DoWorkerDelegate)((System.Runtime.Remoting.
        Messaging.AsyncResult)ar).AsyncDelegate; //Get the
        worker from the result
4    var async = (System.ComponentModel.AsyncOperation)ar.
        AsyncState; //Get the async state from the result
5    bool cancelled; //Create a variable to contain whether the
        operation has been cancelled
6    worker.EndInvoke(out cancelled, ar); //Stop the worker
7    //Lock the sync object to prepare for async operation
        completion
8    lock (pDoSync)
9    {
10     DoIsBusy = false; //Set the async operation to not busy
11     pDoAsyncContext = null; //Remove the reference to the
          async operation's context
12   }
13   //Create the completed event arguments
```

```
14   var completedArgs = new System.ComponentModel.
       AsyncCompletedEventArgs(null, cancelled, null);
15   //Notify any subscribed listeners of the async operation's
       completion
16   async.PostOperationCompleted(
17     e => OnDoCompleted((System.ComponentModel.
         AsyncCompletedEventArgs)e),
18     completedArgs
19   );
20 }
```

Listing B.10: pDoAsyncContext Field and the DoAsyncContext Class

```
1 private DoAsyncContext pDoAsyncContext = null;
2
3 internal class DoAsyncContext
4 {
5   private readonly object sync = new object();
6
7   private bool isCancelling = false;
8   public bool IsCancelling
9   {
10    get { return isCancelling; }
11  }
12
13  public void Cancel()
14  {
15    lock (sync) { isCancelling = true; }
16  }
17 }
```

Listing B.11: pDoSync Field

```
1 private readonly object pDoSync = new object();
```

Appendix C

CD Contents

The accompanying compact disc contains the following items on it:

1. Project Proposal
2. Literature Review
3. Project Poster
4. Final Presentation
5. Thesis (this document)
6. Electronic References
7. Data used in the project
8. Additional software