# ARM-HAND-FINGER VIDEO GAME INTERACTION

A Thesis

by

DREW ANTHONY LOGSDON

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

December 2011

Major Subject: Computer Science

Arm-Hand-Finger Video Game Interaction

# ARM-HAND-FINGER VIDEO GAME INTERACTION

A Thesis

by

DREW ANTHONY LOGSDON

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

| | |
|---|---|
| Chair of Committee, | Tracy Hammond |
| Committee Members, | Thomas Ioerger |
| | Joshua Bienko |
| Head of Department, | Hank Walker |

December 2011

Major Subject: Computer Science

# ABSTRACT

Arm-Hand-Finger Video Game Interaction. (December 2011)

Drew Anthony Logsdon, B.S., Texas A&M University

Chair of Advisory Committee: Dr. Tracy Hammond

Despite the growing popularity and expansion of video game interaction techniques and research in the area of hand gesture recognition, the application of hand gesture video game interaction using arm, hand, and finger motion has not been extensively explored. Most current gesture-based approaches to video game interaction neglect the use of the fingers for interaction, but inclusion of the fingers will allow for more natural and unique interaction and merits further research.

To implement arm, hand and finger-based interaction for the video game domain, several problems must be solved including gesture recognition, segmentation, hand visualization, and video game interaction that responds to arm, hand, and finger input. Solutions to each of these problems have been implemented.

The potential of this interaction style is illustrated through the introduction of an arm, hand, and finger controlled video game system that responds to players' hand gestures. It includes a finger-gesture recognizer as well as a video game system employing various interaction styles. This consists of a first person shooter game, a driving game, and a menu interaction system. Several users interacted with and played these games, and this form of interaction is especially suitable for real time interaction in

first-person games. This is perhaps the first implementation of its kind for video game interaction. Based on test results, arm, hand, and finger interaction a viable form of interaction that deserves further research.

This implementation bridges the gap between existing gesture interaction methods and more advanced virtual reality techniques. It successfully combines the solutions to each problem mentioned above into a single, working video game system. This type of interaction has proved to be more intuitive than existing gesture controls in many situations and also less complex to implement than a full virtual reality setup. It allows more control by using the hands' natural motion and allows each hand to interact independently. It can also be reliably implemented using today's technology.

This implementation is a base system that can be greatly expanded on. Many possibilities for future work can be applied to this form of interaction.

# ACKNOWLEDGEMENTS

I would like to thank the members of the Sketch Recognition Lab for keeping me sane, giving me ideas, and helping me test those ideas. I thank Manoj Prasad and Pat Robinson for assisting with software development, my parents for helping me get to this point in my life and my career, and my fiancée Jessica McMillin for standing by my side and helping me through difficult times. I also thank my committee: Tracy Hammond, Thomas Ioerger, and Joshua Bienko.

# NOMENCLATURE

| | |
|---|---|
| FPS | First-Person Shooter |
| fps | Frames Per Second |
| PCA | Principle Component Analysis |
| HMM | Hidden Markov Model |

# TABLE OF CONTENTS

Page

# LIST OF FIGURES

FIGURE Page

# 1. INTRODUCTION

Human-computer gesture interaction has existed for several decades and has recently become popular in the video game industry. Some of the original approaches used data-collecting gloves [1] and 3D tracking systems [2]. The most popular and more recent approaches use hand-held controllers [3, 4, 5, 6, 7] or vision-based systems [8, 9, 10, 11, 12] for interaction. However, much of the recent work, especially in the video-game industry, has simplified some of the problems associated with gesture interaction, resulting in systems that do not reach the full potential of gesture interaction.

All of these mainstream gesture controls only focus on orientation and location tracking using handheld controllers and/or optical tracking. While an improvement over traditional handheld controls, the lack of finger tracking in current gesture-based controllers means they can potentially be improved upon to help further immerse gamers in their video games.

Gesture-based controllers afford a wide range of movements, but they still require a handheld controller which uses button presses to perform many actions. Vision-based systems enable controller-free input but are not yet powerful enough to detect fingers as well as the arms in real time, which is important for realistic hand-gesture interaction. Virtual reality approaches have the power to exactly replicate the player's body but are frequently complicated and often tire the player with excessive

---

This thesis follows the style of *IEEE Transactions on Visualization and Computer Graphics*.

hardware. Until vision-based methods become viable for real-time hand-and-finger interaction, some existing virtual reality hardware is used to quickly and accurately capture both hand and finger motion.

Arm, hand, and finger video game interaction has great potential. Throughout this thesis, this type of interaction is referred to as "whole-arm" interaction since every component from the arms to the fingertips is used when interacting. No existing gaming system has provided this type of control while using hand gestures to accomplish in-game goals.

Video gaming is a particularly interesting domain not only because of its topical nature but also for its performance demands. Many video games provide real-time interaction between the player and the gaming environment, which itself may contain other players. Such games include first-person shooters (FPS) in which the player experiences the game from an in-game character's point of view. In many of these games the player fires weapons at enemies, objects, or other players and must react quickly to changes in the environment, such as gunfire from other players. In such games, split-second actions can essentially mean the difference between (virtual) life and death. Similarly, physics-based games require real-time interaction between the player and simulated objects. In these and many other genres, the game needs to be able to immediately respond to the player's actions, which means the gesture recognition and interaction approaches need to work quickly to make the game playable.

Glove-based hand gesture recognition has been successfully performed since the 1970's for a variety of tasks, including object identification [2], device control, motion

capture, sign language, 3D object manipulation, 3D drawing/sculpting [13], teleoperation [1], and activity recognition [14]. Data-collecting gloves can quickly and accurately capture finger flexing and abducting, allowing the fingers endless possibilities for interaction. There has not been much work as of yet in the area of finger-gesture input for video games. The possibilities of using a glove and hand tracking system for high-performance video game interaction offer a more complete gaming experience.. This realm of interaction has not yet been extensively explored, and there are many challenges to overcome and great potential for enhancing both the gaming experience and possibly the human-computer interaction experience in general.

To test the viability of finger-gestures in video game interaction, a gesture recognition system as well as a video game system have been implemented. In the game, a player's hands and fingers directly control the game play and menu interaction. The video game system allows unique input and control from each hand. Even though a Flock of Birds sensor provides hand tracking information, the system recognizes the video control gestures solely using hand posture, finger motion, and finger orientation as provided by two CyberGlove sensing gloves. This is done to highlight the power of finger motion and position in recognition. Hand motion is used for visualization, manipulation, and triggering purposes. Manipulation using broad hand movement (without finger postures) might be construed as a gesture in different contexts, however this system does not perform recognition on hand movement, so it is not considered a gesture in the experiments described in this document. Several users played the developed whole-arm controlled games, providing valuable quantitative and qualitative

feedback. This feedback has shown that finger-gestures can be successfully used in high-performance video games, especially in the FPS genre. This implementation bridges a gap between existing popular gesture interaction and virtual reality methods. User experiments have shown that whole-arm interaction is more intuitive than existing gesture gaming and more also more feasible than virtual reality gaming. This work has also helps to identify other game genres that would benefit from whole-arm interaction. Experiments have shown that whole-arm interaction is useful not only in game play, but also in menu navigation. The hand-grasp menu selection was highly praised by users and preferred over existing methods including the Kinect hover, which users complained would make their hands tired. This method of interaction can enhance the gaming experience beyond what is possible with most current gesture-based interaction styles.

The purpose of this thesis is to highlight and explore the initial vision of whole are interaction. Further sections include an overview of related work, a description of the implementation (detailing both the hardware and software design choices), a description of the user study and results, and an illustration of future areas of development.

# 2. RELATED WORK

Video games have become an incredibly popular and expanding industry with thousands of titles developed and millions of copies sold each year. The video game industry is currently worth billions of dollars and new, innovative technologies, both hardware and software, are constantly being developed.

In particular, advancements are being made in human-game interaction. While the idea of video game gesture interaction has been around for several decades (brought to public attention by the Mattel PowerGlove [1]) no commercially successful implementations were introduced until 2006, when the Nintendo Wii was unveiled and released. While not providing truly natural interaction, the Wii proved that gesture controls could work and be used in video games. It was therefore an important stepping stone to more widespread adoption of gesture interaction development. Current developments in video game gesture interaction include motion-sensing controllers (including the Wii as mentioned above), vision-based input using cameras, and glove-based input.

Other types of hand interaction include virtual reality techniques that typically use state of the art, ultra-precise tracking hardware to simulate physical interactions with a virtual world. Such systems typically rely on highly accurate physical calculations to manipulate virtual objects.

Finally, hand gesture recognition has been performed in a variety of other domains. Such domains include robotics and art [3]. Examples of these and all other

systems mentioned previously will now be given along with their benefits and drawbacks.

When using less precise, and less expensive, hardware systems, gesture recognition can be used to substitute for physical calculation. Instead of physically calculating the grip on an object to pick it up, for example, the system can determine that an object is to be picked up by a general grabbing gesture in the close vicinity of the target object. This typically requires much less computational power and can be accomplished with cheaper hardware than the previous approach. This area is potentially a strong growth area of video game interaction as hardware costs come down, processing power continues to increase, and more intelligent systems are developed. The gesture interaction system described in this thesis uses a combination of a hand grasp motion plus location to improve accuracy.

Motion-sensing controller development is one major area of development in video game gesture interaction. There are several commercially available motion-sensing controllers. The Wii's controller, called the Wii Remote [4], or Wiimote, contains an accelerometer and a gyroscope that enable the Wii to replicate the controller's motions in 3D. It also contains an infrared receiver used to determine where the controller is pointing (within a screen equipped with the Wii Sensor Bar). The major drawback of the Wii Remote was a lack of true 3D position detection when it was first released. This problem was addressed by the introduction of an additional gyroscope add-on, called the Wii MotionPlus (which was eventually integrated into the basic controller). However, even with this accessory, the Wii Remote still lacks true absolute

3D tracking and sometimes requires frequent calibration. Additionally, the Wii does not have the ability to accurately identify the grip or hand posture information.

Many gesture researchers have used the Wii Remote or other custom Wii Remote-like controllers for gesture recognition, segmentation, and interaction research, for example [6, 7, 15, 16]. Schlömer [15] outlines a method for recognizing Wii gestures using a Hidden Markov Model approach. The method he uses for gesture recognition using HMMs has become widespread in recent years. Kratz developed a magic spell-casting game called Wiizards [6] that also uses HMMs to recognize spell casting gestures. Before the announcement, release and subsequent popularity explosion of the Wii, Payne developed a similar accelerometer-equipped controller called the 3Motion [7]. It was able to perform gestures in the same way the Wii Remote does.

The Sony PlayStation Move controller [5] expands on the Wii Remote concept by providing absolute positioning (using visual tracking of the handheld controller) in addition to the orientation tracking present in the Wii Remote. While an improvement on the Wii Remote, it is still a controller very much like the Wii Remote, and thus is subject to the same controller-based limitations.

While gestures should allow tasks to be performed more naturally than with buttons, they still impose constraints that can limit the desired natural feelings or even eliminate it if implemented poorly. For example, controller-based systems still require button presses to perform gestures. If the point of gesture interaction is to feel more natural, button presses should only occur when actual buttons are pressed by the in-game characters.

Another major area of development is in vision-based interaction using cameras to track the player. Freeman [17] gives an overview of requirements of vision-based tracking as well as some early approaches and implementations of vision-based tracking and interaction techniques. Several video game companies have released vision-based interaction hardware for their video game systems. The Sony Eye-Toy [18] was one of the first camera attachments for a video game system and allowed players to play games using parts of the body. The Microsoft Kinect [12] is the newest mainstream vision-based implementation for video game systems. The Kinect consists of a pair of infrared cameras and a conventional camera to track movement in 3D. Due to the low resolution (320x240) and required processing power, the tracked human skeleton is limited to 20 joints. The official Microsoft implementation that is used in published, commercially available games uses this skeleton model. The hand and wrist themselves comprise approximately 21 joints, so in the Kinect human body skeleton, the entire hand is treated as one joint. Therefore, no finger tracking is provided in Kinect video games.

To address this limitation, researchers are currently working on methods to allow the Kinect to recognize the hand including the fingers. Ren [9] is developing robust hand recognition using the Kinect sensor. His system recognizes some basic hand poses in cluttered environments. His approach is an important step towards robust visual hand tracking, but it has some limitations. First, it requires the hand to be close to the Kinect's sensors. Second, it requires the dorsal or palmar sides of the hand to be facing the sensor. These problems are described in detail below. Finally, it is not clear if his approach can be used to recognize both hands at once using the Kinect sensor. Whole arm interaction

requires both hands to be used simultaneously as that is more natural than switching between hands for different tasks.

Vision-based finger tracking is being actively researched both with [11] and without [19] visual aids (a colorful glove aids tracking in the case of [11]). Systems powerful enough to track each finger in 3D space require significant computing power and are prone to occlusion problems. In addition to these problems, Wang's approach [11] to recognition using a colored glove still contains too much latency between gesture and visualization to be used effectively in video game interaction. Video game players require very low latency as very fast and complex finger movements might be performed [10].

Other research is being done on visual "naked" hand recognition, i.e., nothing is worn on the hand. De La Gorce's approach [8] to 3D hand pose estimation using a camera may be an important step towards vision-based interaction using cheap, widely available camera hardware. Hamer [20] has recognized the hand while it is holding an object, thus making progress toward solving the occlusion problem. As with Ren's approach, these other hand tracking systems require the hand to be close to the camera.

For this implementation, a full-body view is required. The view used by current Kinect games is ideal since it can view the entire human body and therefore capture total arm movement. The related work approaches mentioned above all require the hand to be close to the sensor, and are thus not ideal for interactive high motion FPS games. Alternatively, the approach described in this thesis allows the hands to attain any physically possible orientation. Additionally, this approach will work with any complete

model of the hand, however it is obtained, vision or otherwise. Many visual approaches will fail if the hand majorly occludes, or blocks, itself. Such a possible pose might involve the index finger pointing directly at the screen or one hand completely occluding itself.

For vision-based systems, poor resolution often imposes excessive noise or simplified body detection, both of which can hinder the interaction through low precision and incorrect recognition. Because vision-based tracking requires high computing power, it is not yet adequate for tracking the motion of the fingers and the hands in real time.

Some researchers have created systems that allow for video game interaction using the fingers. Sugiura et al. developed Walky [21], a system to control bipedal movement using finger input. It works by capturing finger gestures using a multitouch screen (an iPhone in their demonstration) and translating the movements to a bipedal actor (which can be a video game character, robot, etc.) which will then move its feet according to the paths defined by the finger gestures. This approach is novel and uses the fingers as direct input to a game, but it does not fully capture the movements of the entire finger. The gestures are also not as natural in the sense that the fingers are performing the movements of the actor's legs rather than the actor's fingers.

Komura [22] also implemented interaction by using the fingers to control a virtual character's legs. The user, wearing a data glove, moves her fingers as if they are legs and her hand is a character. The virtual character's legs move according to the user's finger motions. This method of interaction was used to control a virtual character

moving around a space and jumping over obstacles. This form of interaction does use the fingers in a gaming setting, but like Walky, the fingers are not used to control virtual fingers. Instead, they control a character's legs. Therefore, it suffers the same drawbacks as Walky.

Yet another area of development in video game interaction centers on using data gloves to more directly capture hand movements, especially the angles of the fingers. Gesture recognition work using gloves has been developed for various domains [1]. For example, Paulson [14] analyzed static hand poses to identify common office activities. He used a nearest neighbor algorithm to match hand poses with the average values of a few training examples. A follow-up study was performed in [23] and showed that simple activities can be recognized using simple classifiers. These ideas and a similar nearest neighbor implementation were initially considered and tested for video game interaction. More complex gesture recognition algorithms were required for successful and immersive video game interaction.

The Peregrine gaming glove [24] is one of the very few glove inputs designed for video games. It does not provide any finger tracking, however. Instead, it is dotted with buttons across the fingers and palm to allow more natural button pressing for interaction.

These are a small sample of some of the existing applications of glove-based interaction. While people have used gloves (containing finger tracking) for many different domains, very little whole-hand gesture interaction (including the fingers) has been explored in the video game domain in which the player's hand motions are directly mapped to virtual character hand motions.

Virtual reality has seen direct mapping of the user's hands for some time. Many applications use motion tracking hardware (usually using a sensing glove such as the CyberGlove) to exactly mimic the user's hand in a virtual world. This is a very natural form of interaction since the movements in the real world correspond to the exact actions in the virtual world. Virtual reality approaches generally do not apply gesture recognition in their interaction techniques. Rather, they typically define some type of hand model, such as [25], to determine properties that are then used to aid interaction. Many approaches also define some form of contact point calculation, such as [26], to enable precise virtual object manipulation. These approaches are appropriate for ultra-precise virtual reality applications but may be too complex for video game developers and applications.

Gesture recognition has also been performed within the virtual reality domain. Aleotti [26] performed hand grasp recognition in a virtual environment using hand data collected from a glove. He analyzed contact points and contact normals on simple primitive shapes to recognize hand grasp from a set of grasp types. This method could be problematic for more complex scenes and objects found in modern-day games and may be too complicated for game consoles and/or developers since using this method for gesture recognition would require complex contact and grasp analysis over time. LaViola performed [27] whole-hand input in a virtual environment using a combination of hand gestures and speech input to design a room using a stereoscopic display. This is a step closer to the interaciton implemented for this thesis, though some gestures are not natural, such as using pulling gestures for navigating through the 3D space or using a

pinching gesture to grab items instead of a natural grab gesture. It does allow for independent input with each hand.

Hand gesture recognition has been performed in other domains as well, particularly in robots. In fact, Aleotti's method goes on to be used in robotic programming by demonstration (PbD). Iba [28] used simple pointing and waving hand gestures to operate a mobile robot. His interaction consisted of a few simple gestures, such as waving the hand left or right, making a flat-palmed stop gesture, and making a finger-pointing go gesture. He trained a simple HMM to recognize his gestures using motion data collected from a data glove and tracking hardware. Bernardin [29] also performed hand gesture recognition using data from a sensing glove and tactile sensors. He trained an HMM using this data to recognize a set of hand gestures. Like Aleotti, this approach is used in a robotic PbD.

Schkolne [13] used hand gestures as a 3D sculpting tool. Coupled with augmented reality glasses, he can adjust and create geometry using his hands. His system does not perform any gesture recognition. Rather, it uses the hands for direct manipulation.

Clearly, hand gesture interaction and recognition has been performed in a large variety of domains using a variety of hardware and recognition techniques. Despite all these approaches, we still have not seen direct hand mimicking and hand gesture recognition using finger motion in a video game setting. This is a clear gap between existing methods of gesture interaction and the exact methods of virtual reality. This thesis aims to fill this hole by demonstrating a video game interaction system that

accomplishes these two goals. The placement of the work described in this thesis is illustrated in Figure 1. A scenario of the working system is described in the next section.

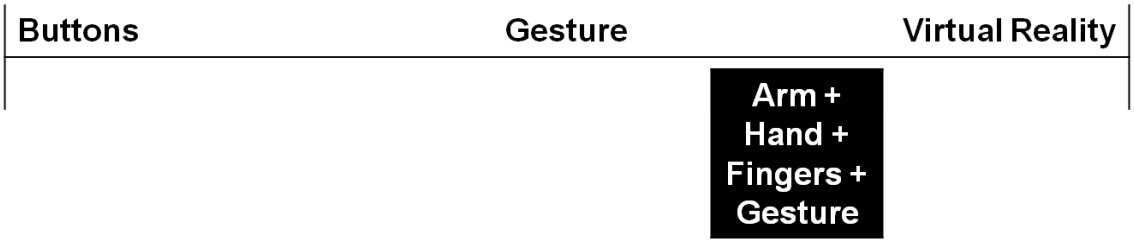


**Figure 1:** The continuum of interaction shows the placement of this work.

# 3. CONCEIVED METHOD OF INTERACTION

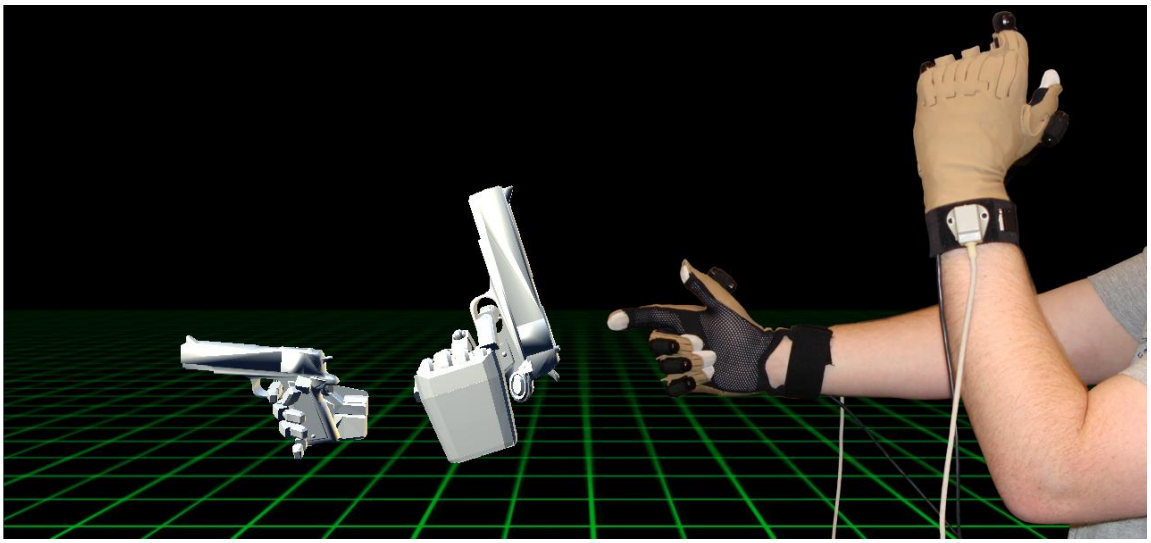

**Figure 2:** A player wields two pistols. The 3D hands move in the same manner as the player's hands. The player is wearing the CyberTouch gloves and has the Flock of Birds sensors attached to the wrists of the gloves.

Imagine you want to play a video game. Instead of picking up an Xbox controller or a Wii Remote, you slip on a pair of gloves. Figures 2-4 provide examples of the types of interaction that can be performed. The following paragraph describes a scenario and the envisioned method of interaction.

When you enter the game, your character stands in the environment, his hands empty. You need to travel to a castle to deliver a message to another character in the game. As you are walking through the woods toward the castle, you hear yelling in the distance. You realize you are about to be attacked by bandits. To prepare for battle, you reach over your shoulder with your left hand and grab your longbow while you reach for your hip and grab an arrow. As the bandits approach you, you load your arrow into the

bow, pull it back, making sure you feel the tension in your back muscles as you lock your elbow. You take careful aim at one of the bandits and release the arrow, hitting him in the chest and disabling him. You then finish off the rest of the bandits in the same way.

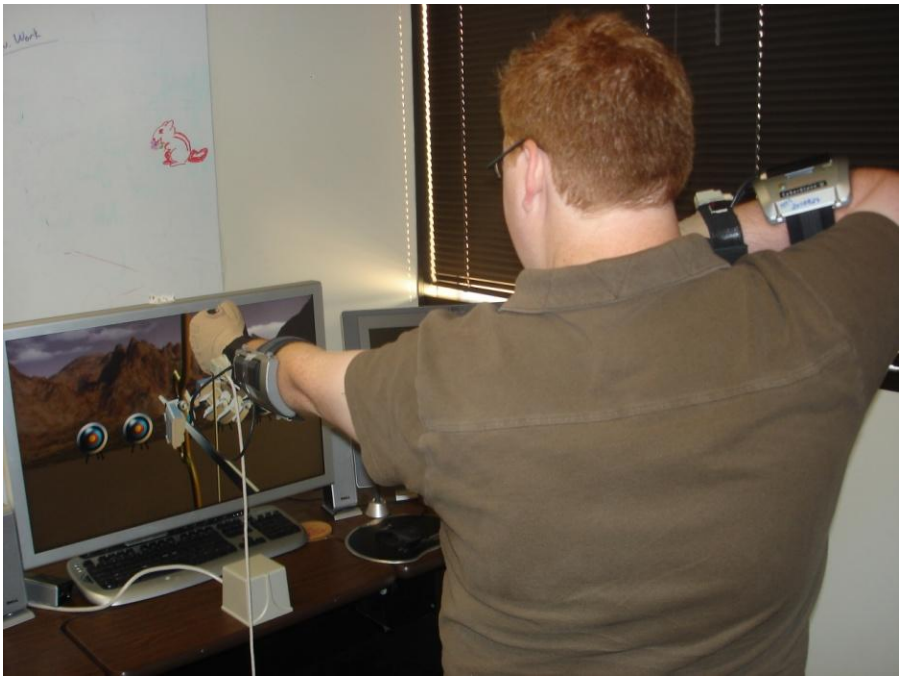

**Figure 3:** A player uses a bow and arrow.

How did you control your character while he was firing at the bandits? You don't have an Xbox controller, so you didn't manipulate a control stick while pressing some buttons. You don't have a Wii Remote, so you didn't aim a controller at the screen. No, you did exactly what the character did; or rather the character did exactly what *you* did. No button presses, no artificial gestures, you actually reached over your shoulder to get

your bow out. You reached to your hip to pull out an arrow. You reached up and placed the arrow in the bow, pulled it back, and released. You aimed the bow by aiming your arms as seen in Figure 3 and Figure 4.

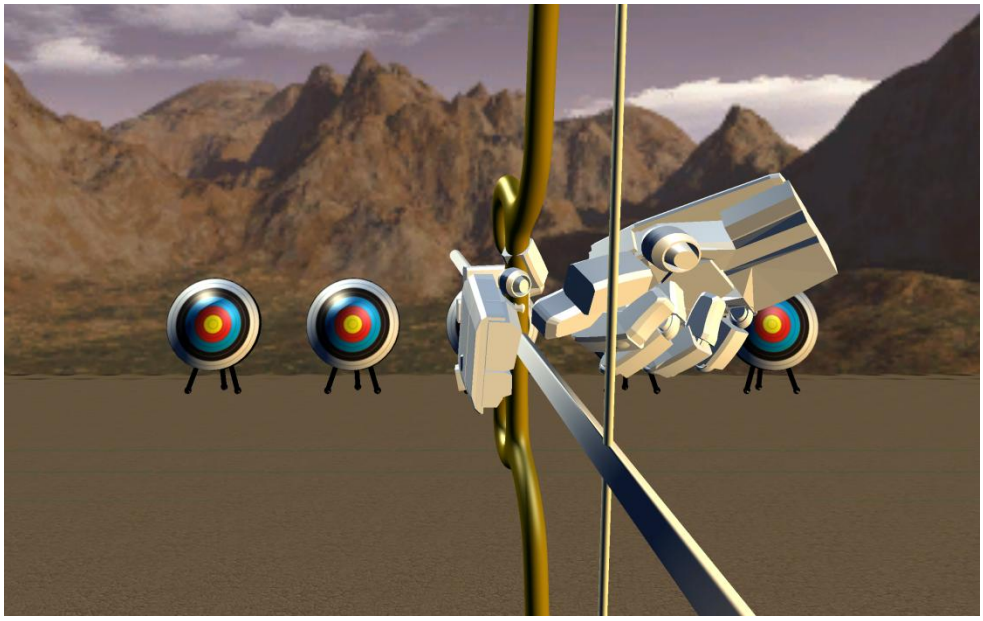

**Figure 4:** Visualization of the player using the bow and arrow.

This is the method of interaction that is the inspiration for this thesis and is the method of interaction that was implemented and will be described in this thesis. Most current gesture interaction systems approximate the actions performed by introducing a controller laden with buttons (Wii Remote, PS3 Move) or treating the hand as one big, fingerless blob (Xbox Kinect). These companies, and many researchers, have excellent reasons to simplify their interaction styles. It is much easier to map a button press to a command than trying to segment a hand gesture from a constant stream of data. It is also

easier to move the entire hand to an area to accomplish a task than trying to use the fingers to manipulate something you are not physically touching. While an improvement over non-gestural systems, the act of simplifying the problem waters down the natural feeling intended by introducing gestures.

The main goal and contribution is to further the use of hand gestures in video games by incorporating the actual use of the fingers as game controllers and directly controlling a virtual character. In many first-person games, the player controls a character whose hands are typically displayed on the screen. When the player performs some action, these static hands will animate in a predefined manner to illustrate the action and provide some basic sense of realism and immersion. However, these hands generally do not mimic the player's actual movements. Even when gestures are used to perform actions, such as with the Wii Remote, the character's hands will still perform some pre-defined animation. This has been disappointing for the first generation of gesture-interaction gamers that expected swords or fists to hit their opponents in the same manner that they swung the controllers.

A first-person perspective video game could exist in which the player directly controls this virtual character's hands by moving his/her own hands. The once static hands displayed on screen now become dynamic and mimic the player’s every action exactly. For example, if the player swings a sword at an enemy, the character's hands will hold the sword at the same angle and swing it on the same trajectory and with the same velocity as the player's hands. Even minor inconsequential gestures, such as waving to another player or scratching your head, open up a new world of realism and

immersion for video games. This introduces many new possibilities for interacting with in-game environments, objects, characters, and other people. This will provide a more engaging and novel experience than current motion controller- or vision-based games can provide.

This is whole-arm interaction. This thesis intends to demonstrate that pure hand gestures can be used to interact with a gaming system, and that this method of interaction exceeds current implementations in terms of naturalness, novelty, and fun.

# 4. DOMAINS

Today's mainstream video games are incorporating more gestures into their controls since the advent of the Wii console's accelerometer-based gesture controls in 2006. The Sony PlayStation Move the Microsoft Kinect have recently been introduced. Nintendo is generally credited with popularizing gesture controls, and the recent introduction of the Microsoft and Sony products will surely further expand gesture interaction. As a result of recent popularity, many companies and researchers are investing in the area of gesture interaction. Because of these reasons, testing in the area of video games seems a good match for this research.

Many video games provide a character for the player to control. These characters perform a wide variety of tasks that can be accomplished with the hands. Such tasks include picking up items, using items that have been picked up, dropping items, using items fixed to the world, driving vehicles, placing items in particular arrangements, and interacting with other characters. There are surely many more conceivable types of interaction as well, and those listed here contain many different variations. In current games, when the player performs an action, the video game character will typically be animated to perform these tasks. The player has typically only pressed a button or performed an unnatural gesture to activate the action.

Current systems only approximate gestures to be performed rather than allowing complete, natural gestures. For example, a Wii gesture typically consists of moving the remote along some path. This path usually does not translate to a literal path in the game,

but rather uses the general direction, speed, and orientation to let the game know that a particular pre-defined action needs to be performed. In advanced cases, the actual angle and speed might be used. However, the actual gesture made by the player does not translate to the game. Many players were excited that they might be able to wield a sword as in real life using Wii Remotes, but games featuring swordplay disappointed by only using a few pre-defined swings that were selected by the gesture. The PlayStation Move has taken steps to fix this issue with its absolute positioning technology, but the limits of controller-based gestures still remain. Whole arm interaction would allow the gestures performed by the user to appear exactly in the game. The character's hands, including fingers, move just as the player's hands move. For example, if the character was just walking down the street, doing nothing, and the player reached up to scratch his head, the in-game character would scratch his head in the same manner. The current system already mimics the hands in 3D.

While the recognition algorithm interprets the gestures in real time, the game creates real-time visualizations that mimics these actions as they exist in real life. With this combination, if the player reaches for an item in the game to grab it, we will see the character reach for the item, and then the character will actually pick the item up. This combination of natural gesture, exact visualization, and recognition would make the game more immersive.

In contrast to video games, virtual reality represents another spectrum of uses for hand gesture recognition. Such systems allow exact human representation in a virtual environment. Typically, exact manipulation happens in a virtual reality setup. For

example, to pick up a cup, the actual physics of grabbing and lifting the cup might be calculated in a virtual reality experience. If the player's hand slightly adjusts the grip, the system will calculate this and adjust the cup properly. If the grip loosens or holds the cup inexactly, the cup could fall out of the player's hands. This could possibly be tedious for gamers, especially those playing fast-paced games such as first-person shooters, because exact grips are not necessary and could ruin the experience. For example, if the player is holding a gun but shifts or slightly relaxes the fingers, the gun could be dropped accidentally.

Hand gestures are used to solve this but keep interaction feeling natural. Instead of exactly grabbing an object (such as a gun), a "grab" gesture (using the fingers) can be performed in the vicinity of the object, and then the object can become attached to the virtual hand. Motion of the hand and arm will adjust the positioning of the held object, and a "release" gesture can cause the object to detach from the player's virtual hand. This type of interaction can feel natural to the player but avoid the pitfalls of virtual reality when applied to gaming.

## 4.1 Alternate Domains

The hand posture and gesture recognition algorithms described in this thesis are also useful for a myriad of other domains. Because people use their hands in most activities, the possibilities are endless. Hand posture recognition is also useful in the field of teleoperation, in which an operator moves his or her hands to control a remote robotic device. Gesture recognition could recognize any gestures that may be outside the

limits of the robot or otherwise deemed unsafe. This could occur with remote surgeries, for example. If an unexpected interruption causes the surgeon's hands to falter in a manner that would lacerate the patient, the software could disable the remote robotic surgeon.

We have seen hand-gesture applications in areas such as art. Some systems have allowed the creation of 3D drawings and sculptures using the hands to directly manipulate the creations [13]. The algorithms described in this document could be applied as such by allowing the use of artistic tools manipulated using hand gestures.

# 5. RESEARCH PROBLEMS

## 5.1 Recognition

The key to whole-arm interaction is effective and accurate hand gesture recognition. Gesture recognition is key to this form of interaction, and the gestures players perform must be natural. The gesture recognition problem can be subdivided into two sub-problems: data segmentation and classification.

The first problem to be solved in a system like this is the segmentation problem. Controller-based systems, including the Wii Remote, PlayStation Move controller, and the computer mouse can easily segment gestures using button presses. For example, to begin a mouse gesture, the user may press down on one of the mouse buttons, move the mouse in some gesture, and then release the mouse button. Such a gesture is segmented by button presses. However, when tracking the hands, we are dealing with an "always on" connection. The player has no buttons to delineate gestures; rather, we must pick out gestures from a single long stream. This is an interesting, and difficult, problem that has many possible novel solutions.

Many researchers have approached the segmentation problem from many angles, including using hand tension [25], dwell time, specific poses, knowledge of human motion [30], HMMs [28], and speed, among many other approaches. Different types of motion require different segmentation as well. The method of segmentation could possibly be dependent on the type of interaction/gestures being performed.

People can perform gestures in many different ways. Though they might seem similar to humans, slight variations in a gesture can look completely different to the computer. This creates a big problem for both segmentation and recognition. For example, if a player performs "fishing pole" gestures, for example if controlling a virtual fisherman, she might cast the line along different trajectories and her hands could be varying distances apart each time she casts her line. When she reels in her line, her hand could travel along an ellipse of varying radius and angle while her non-reeling hand could be grasping the pole at any location. These variations seem trivial to people but could cause significant differences in segmentation and recognition. For effective performance, the segmentation and recognition algorithms must be able to automatically identify the important aspects of each individual gesture.

## 5.2 Defining a Gesture Path

In order to effectively define a gesture path, the system divides the hand postures into time steps each containing information about the current hand posture. Vector $v$ contains the data generated by the hardware at each time step $t$. Each time step consists of a 3D XYZ coordinate, a 3D Yaw/Pitch/Roll coordinate, and 18 glove angles. Therefore, at each time step the system generates a 24-element vector, denoted as $v_t$. As the program runs, the system divides the long vector v into smaller vectors for recognition.

Once the long vector *v* has been divided into smaller, more usable gesture vectors, they must be analyzed to determine if they are meaningful. This is the

classification aspect of the recognition problem. Specifically, the system attempts to determine if the generated real-time collection of hand postures matches a pre-recorded gesture from the training set.

Each time step in the gesture has both a location (similar to in a hand drawn sketch, where each point has its own time stamp) as well as its own static pose as illustrated in Figure 5. If the hand poses are ignored, each gesture becomes a 3D sketch which could possibly be recognized in some manner. The hand poses can either be analyzed alone or in conjunction with the 3D stroke. In order to highlight the power of finger interaction in recognition, the system uses only the finger and hand pose information, and not the location, to recognize both which object the user is holding and how they are interacting with it. We assume that accuracy will only improve with the addition of hand location information; other systems have already shown the power of recognition using hand location.

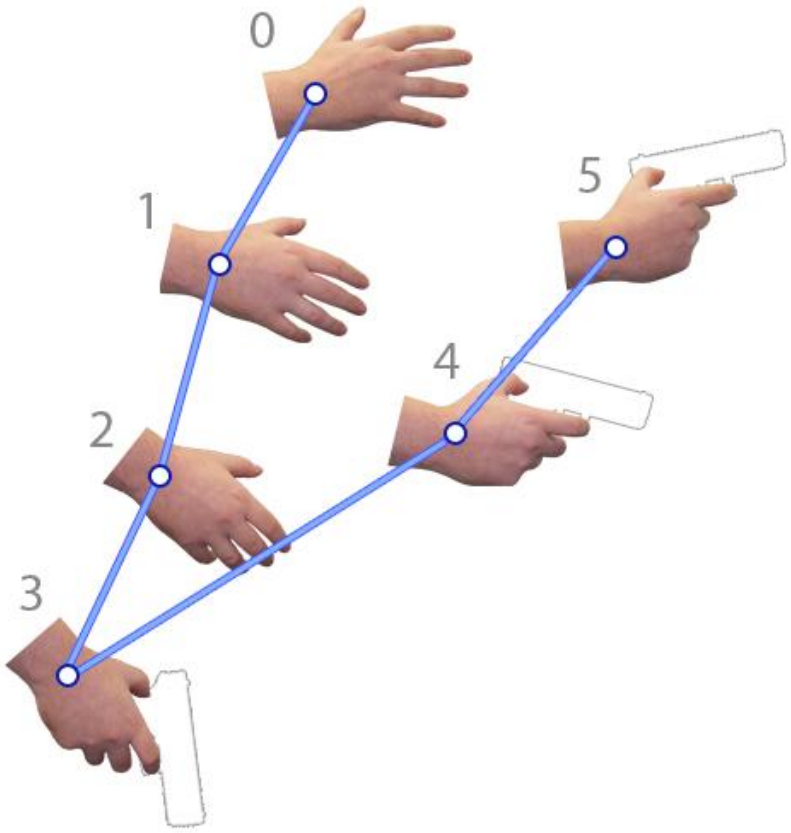


**Figure 5:** Example gesture stream to grab a pistol from the hip and aim it forward. Each white node represents a 3D point. Each point has an associated hand pose and time stamp. In the game, a pistol is obtained using the gesture {0→1→2→3}.

## 5.3 Hand Representation

In order to interact with a virtual world using the hands, the hands need to be represented visually in the 3D world. Because the player will be interacting with an on-screen virtual world using direct manipulation, the virtual world must accurately mimic the actions performed in the physical world. Direct manipulation is necessary to provide real time feedback for the player's actions. In this case, the player's virtual hands must

be rendered in an identical manner to their real-world physical locations and postures. If the system fails to do this, the player will probably not be able to perform any actions in the game because his or her hands cannot be seen in the virtual world. Accurate hand representation is the most important form of feedback for the user.

Hand representation consists of two components. First, the hands' 3D positions must be tracked in real time. This is necessary to place the hands in appropriate locations in the virtual world. For example, if the player forms a 'Y' by holding both arms straight out and to the sides above her head, the virtual hands will move to the corresponding locations in the virtual world. These locations might be close to the top corners of the screen and may in fact extend slightly beyond the screen space if the player has a long reach. By positioning the hands in 3D space, the player gets a sense of where his hands are in relation to virtual objects such as tools or obstacles. It is important that the virtual hands be located reasonably close to real-life equivalent locations because the player will be looking at the screen and not at her hands when performing actions. If the virtual hands are placed badly, the player might have difficulty performing actions and might have to perform unnatural actions to be able to position the virtual hands correctly. Obviously this issue would ruin any naturalness of interaction, so it must be minimized.

The second component to hand representation is accurate portrayal of the fingers. While the hands' locations are tracked in 3D space, the fingers flexion (bending as in when making a fist) and abduction (spreading the fingers apart) must be accurately tracked in real time. The virtual hands must flex and abduct as the player's hands flex and abduct. This is important as it, along with the spatial tracking, helps convey the

gestures the player performs. In fact, the finger positions may be more important than the hand position because the player can perform many intricate actions using the fingers alone. In the real world, if a person loses all her fingers she will become severely limited in the actions she can perform. Similarly, if the player doesn't have use of the fingers in a game, she will not be able to perform many of the possible in-game actions. Therefore, representation of hand pose is extremely important to natural interaction in a virtual world.

Hand position can be tracked in several different ways. Perhaps most naturally, camera-based trackers can be used to locate the player's hands using computer vision algorithms. This form of tracking is advantageous as it does not impose any hardware upon the player. This allows more freedom for the player to freely move around in the gaming space which is highly beneficial for natural interaction. However, vision-based tracking also has the drawback of requiring a large amount of computing power to process the image to extract the player's positions [10]. To be able to accurately track the player's hand movements, the camera needs to view an area at least the entire possible space the hands could occupy. This would conceivably be a six feet by six feet plane around the player. To be able to accurately track the hands, the camera's resolution should be as high as possible. With an increased resolution comes a decrease in performance as more pixels must be processed. Therefore, to be able to most accurately track the hands, a high resolution camera should be used. A typical video game runs around 30-60 frames per second (fps). To be able to interact with the game in real time, 60 fps is required. Such high performance tracking using high resolution camera tracking

is still a hard problem in computer science. Even low resolution tracking systems, such as the Microsoft Kinect, exhibit a short delay, or lag, when parts of the body, especially the arms, are moved. In summary, visual tracking is ideal for natural interaction and immersion but still suffers from performance issues, especially regarding highly accurate tracking.

In addition to hardware-free visual tracking, we can employ a few types of wearable tracking technology to enable real-time, accurate location tracking with minimal delay. First, we can use an array of accelerometers placed at the chest, upper arm, and wrist. This has the advantage of providing only the data necessary for hand location computation using accelerometer values. This means the locations can be tracked very close to real-time with minimal delay. However, the player would have to don at least five sensors which, in addition to taking time to put on, could provide a hindrance, however minimal, due to the unnatural sensor presence. This could possibly affect immersion if the player finds the sensors distracting or uncomfortable. Besides accelerometer-based tracking, we can use other sensor tracking technology such as magnetic tracking. These trackers work by using a base transmitter unit that emits a magnetic field and tracks magnetic sensors that are placed within range of the transmitter. While the transmitter can be large, bulky, and heavy, the sensors are very light and can be attached to the wrists with ease. The tracker can be placed in front of the player, and the player's hands will remain in range of the transmitter. This solution has the advantage of providing extremely fast and accurate tracking while using very small sensors. As with other sensor-based solutions, the sensors may encumber the player and

cause a decrease in immersion. Magnetic tracking systems are also expensive in comparison to accelerometers and especially cameras, though they provide the fastest and most accurate tracking.

For tracking the fingers, there exist similar options to tracking locations. We can either use vision-based or sensor-based tracking. Once again, by using visual tracking the hands will be free to interact most naturally. However, the problems will be exacerbated by the finer motions, smaller size, and the complexity of the hands. As mentioned before, the camera must retain a wide view of the player to also be able to track hand position. Because the fingers are tiny in comparison to the arms, the camera will require a very high resolution to be able to see the fingers in good detail. As before, this will demand a high CPU load which could cause significant delay in the virtual hand response. Because of the hand's complexity, a new problem will also be introduced through finger occlusion in which part of the hand could block, or occlude, other parts of the hand. For example, if the player crosses her fingers, part of one finger will pass behind another finger, which could cause recognition difficulty. More severely, if the player makes a fist, several fingers may become completely blocked by other fingers and the thumb. The occlusion problem is an active research area within visual tracking.

All of these technologies will work, however, we use sensor-based tracking to track the finger's flexion and abduction in real time. To be able to precisely measure the many different angles of the fingers, an array of sensors can be placed all over the hand. Several companies have integrated different types of sensors into gloves for easy application and removal. Two major types of sensors are accelerometers and bend

sensors. Similar to an accelerometer solution for tracking the hands' locations, a glove embedded with accelerometers can use the orientation data to calculate the exact angles of the fingers. Bend sensors, on the other hand, measure angle by calculating the electrical resistance of a circuit passing through a flexible strip. As the sensor is bent more and more, the electrical resistance will increase. This resistance measure can be converted to the exact angle the sensor is bent. Several companies have embedded these types of sensors into gloves as well. Such glove-based tracking solutions have the benefit of very fast and accurate tracking at the expense of wearable hardware that could hinder immersion. Data-collecting gloves cost more than cameras as well. These benefits and drawbacks align with those of the hand position tracking discussed above.

The goal is to maximize immersion for the players using their fingers while playing a game. As such, the gloves are an added hindrance, and a camera tracking system would be preferred so that the players do not need to wear any hardware. However, the weight of the gloves is minimal, and the accuracy is far better than current camera tracking methods (and many of the current camera tracking techniques still require some form of gloves). Given the minimal weight and high accuracy, glove technology is used for hand and finger tracking in this thesis since 1) the application requires high performance and accuracy, and 2) the focus is on the interaction and gesture recognition rather than the tracking technology. When vision-based tracking becomes powerful enough to accurately track hands and fingers, the algorithms described in this thesis will be easily applicable to a camera based solution, but for now the performance and accuracy demands necessitate sensor based tracking technologies.

This thesis only covers visual feedback, which is probably the most important type of feedback for the type of interaction described here. However, haptic and other types of feedback in addition to visual feedback will help to further enhance the gaming experience. Such possibilities will be discussed at the end of this document.

## 5.4 Game Interaction

Once the hands are represented virtually, they need to be used in a meaningful way. As described earlier in the conceived method of interaction, the player's hands will effectively be substituted for a virtual character's hands in the games. While this will be visually pleasing and realistic, the game must respond correctly and realistically to the player's actions. If the player grabs hold of some control, the actual motion of the player should affect the control, and its analog motion should accomplish the desired action. For example, if the player grabs a lever that governs the speed or direction of some other object, the lever should move linearly in accordance with the hand motion. The response of the controlled object should correspond to the actual position of the control instead of some pre-programmed setting.

## 5.5 Hardware Issues

Error (noise) is a problem inherent in sensing technologies. The real world provides perceptually infinite resolution while any sensor is very limited in comparison. For example, the Flock of Birds tracker is accurate to 0.5 mm within 12" of its transmitter and slightly less accurate as the hand moves farther away from the

transmitter. While this is reasonable for static gestures, very fast sensor movement can cause a jitter in Flock of Birds position data. As a result, the captured gesture paths do not accurately reflect the true motion of the hands. For example, if the user reaches forward at what seems to be a smooth motion, the graph of this path might look very jagged.

If hand motion is added to the system in future work, care must be taken to compensate for this high noise level. Possible solutions include extra filtering or hardware replacement with a better tracking system.

The CyberTouch glove data contains much less noise than the Flock of Birds data while the sensor is moving. Though the error is significantly reduced in the CyberGlove, and thus much less of a concern, it is still important and must be paid attention to.

# 6. HARDWARE

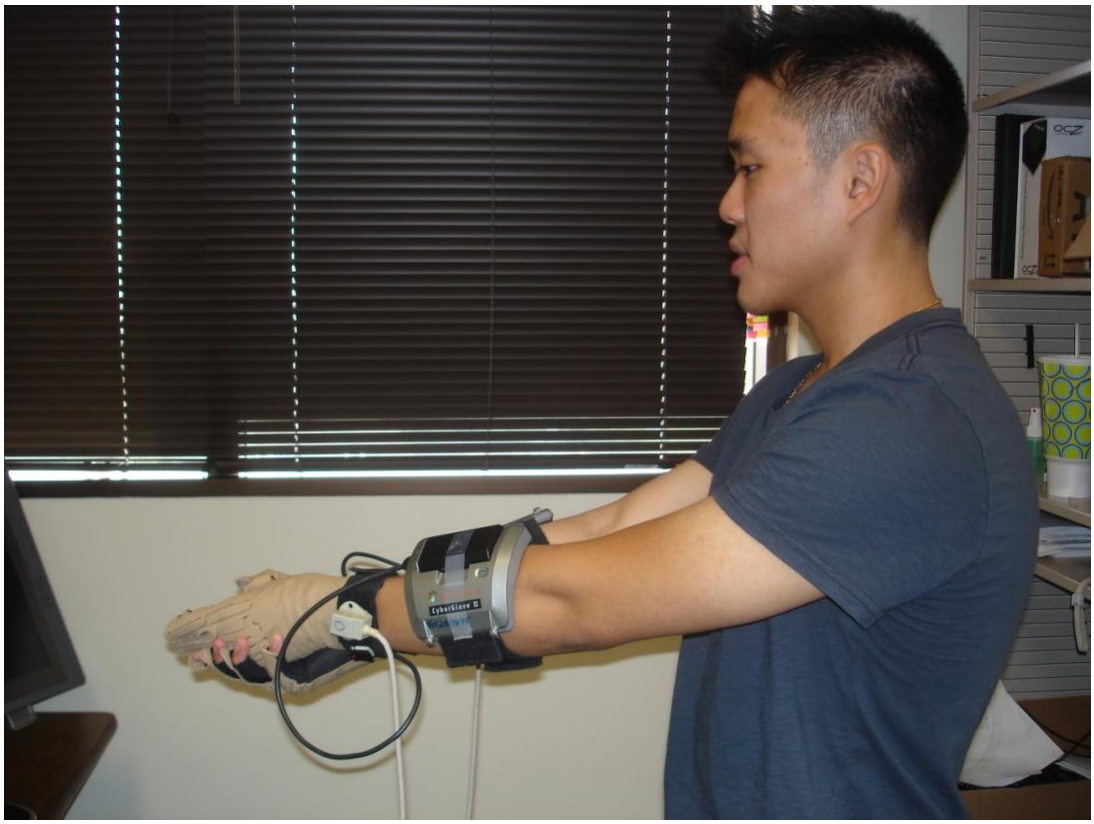

**Figure 6:** A player plays a game using whole-arm interaction. The CyberGloves are visible on the player's hands and the Flock of Birds sensor is attached to the player's wrists.

This implementation uses two wireless CyberGlove [31] gloves by CyberGlove Systems to track the hand postures for effective visualization and gesture recognition and two Flock of Birds trackers [32] to track the wrist motion for effective visualization only. This hardware can be seen in Figure 6. The hand-tracking gloves contain 18 bend sensors to track angle measurements across the hand. The accompanying VirtualHand SDK [33] uses this raw data to give a calibrated 22-element vector as well as calculates

the 3D position of each hand component. The sensors capture finger flexion (bending the finger so it isn't pointing straight) and abduction (moving the fingers apart from each other), wrist flexion and abduction, and thumb roll. An example of flexion and abduction can be seen in Figure 7. The VirtualHand SDK performs the kinematic calculations to render the hand in 3D in real time. Figure 8 provides an illustration of the sensor locations on a real hand.

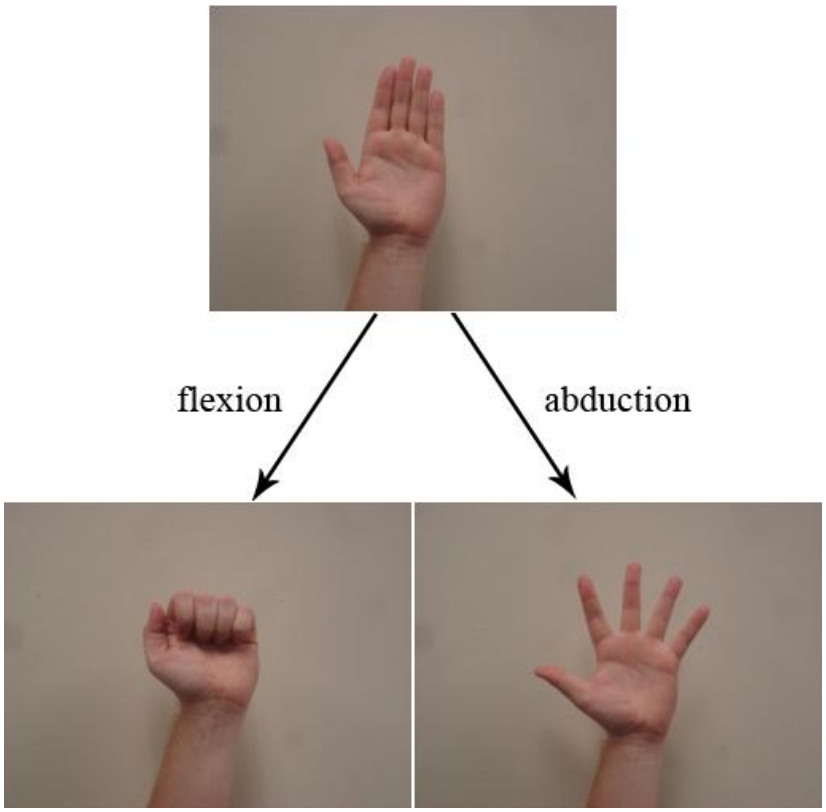


**Figure 7:** The fingers of the hand flex (lower left) and abduct (lower right). The CyberGlove can sense these two types of motion of the fingers and the wrist.

The CyberGlove products are among the best commercially available hand-tracking gloves in the world at the time of this writing and have been for many years [1]. They provide <1 degree of resolution at 90 samples per second and have a repeatability of 3 degrees (the average standard deviation between wearings) according to the CyberGlove Systems web site.

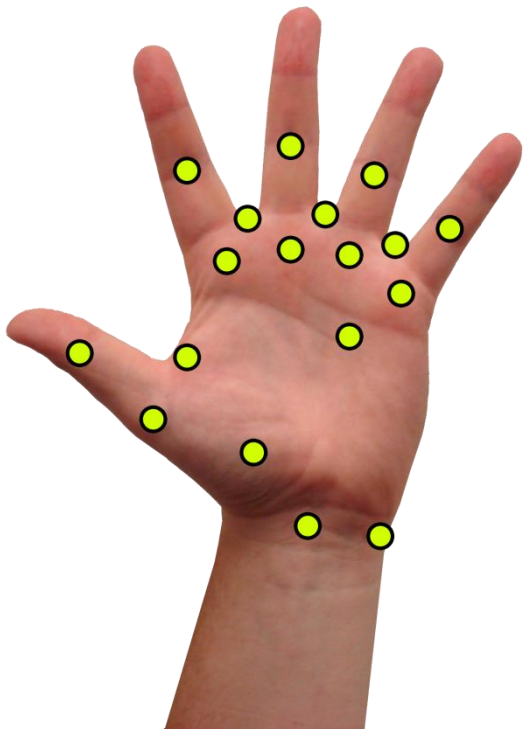

**Figure 8:** Joints whose angles are measured by the CyberGlove.

The Ascension Flock of Birds DC magnetic tracker was used to track the position of the hand in space. The Flock of Birds is able to track multiple sensors' position and

orientation in 3D space. One sensor was attached to each wrist to track each hand's location and orientation.

Unlike the CyberTouch, the Flock of Birds is an older piece of hardware and not the best in its class. In fact, similar technologies were used for tracking as far back as 1980 [2] with the first gesture interaction systems. It can accurately track the position of its sensors to 0.5 mm (at 12 inches from the transmitter) at 144 recordings per second, though the tracking can "jump around" when moving, especially at high speed. This noise factor is discussed in more detail in the following sections. Although hardware availability and accuracy needs force the use of the Flock of Birds, better, more current tracking hardware could enhance the experience, and possibly also segmentation and recognition, by reducing movement noise. Indeed, newer revisions of the Flock of Birds improve upon this issue.

The Flock of Birds gives standard XYZ positioning and yaw/pitch/roll orientation. The CyberGlove data is more specialized with its 22-element vector of angles. One goal of this research is to develop device independent methods so that the approach is usable in other configurations with alternative hardware and input methods, even vision-based algorithms.

The CyberGlove does not have the issues with latency that current vision-based approaches have. As the focus of this thesis is not to create better tracking software, but instead focus on interaction style, the methods have been developed to allow for alternative input devices. Once vision-based methods become powerful enough for this style of interaction, such "naked hand" interaction should prove superior to a glove-

based approach with regards to game immersion. As such, hardware related restrictions are not a major concern of this thesis. In fact, the only hardware restrictions come from the wired 3D tracking sensors. Wireless trackers are currently available, including the current successor to the Flock of Birds and the new Kinect, but they were not available at the time of initial implementation. Some critics assert that the gloves will also hinder immersion. Once optical methods become viable, the gloves can be removed, and the same methods used. The gloves and tracking hardware can be seen in Figure 6.

The data collected by the hardware allows the computer to know, in real time, where the hands are and what the hands are doing. This is a combination that many hand gesture implementations do not provide due to many factors, including required processing power, algorithm complexity, hardware cost, implementation cost, and dearth of research in the area. This thesis demonstrates that current computing hardware and sensors have reached a point where the implementation of complex input solutions can increase immersion in video games.

# 7. IMPLEMENTATION

## 7.1 Recognition

Determining the length of the sequence of gestures to send to the recognizers (known as segmentation) is a difficult problem that is still being researched today. As the main contribution centers on interaction, this thesis does not aim to solve the segmentation problem in a unique way. However, some solution is required for gesture recognition. This thesis uses a simple "sliding window" approach (see Figure 9) which takes the last $n$ samples from the hardware. The size $n$ of the window is the average length of each recorded gesture. For example, if the average length of the grab gesture in the training data is 1 second, the system uses the last 1 second of data for gesture recognition. Therefore, every HMM will be sent a stream of data of varying lengths. This simple approach enables simple, reliable segmentation that will be suitable for testing. Future work may include more sophisticated segmentation methods.

After segmentation, the extracted gestures need to be recognized. The initial implementation included a simple Nearest-Neighbor recognizer identical to that used by Paulson [14] for static pose recognition. This worked well for gestures consisting of static hand poses, such as pulling the trigger of a gun, but did not work for gestures where the fingers must move, such as grabbing an object. Thus, to increase recognition capabilities, future implementation used a Hidden Markov Model (HMM) based recognizer.

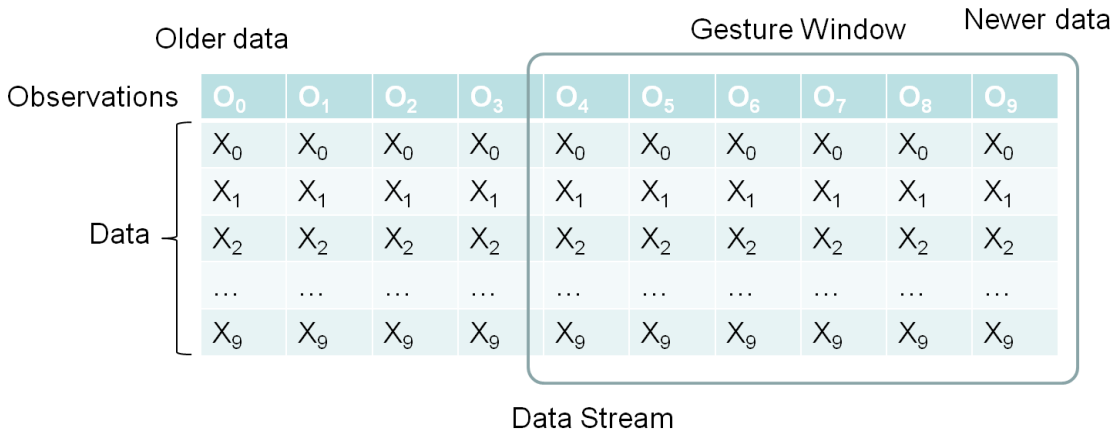


**Figure 9:** Example of a sliding window segmentation approach. In this example, the window size is 6, so the 6 most recent observations are used for recognition.

Rabiner [34] provides a guide on HMM structure, algorithms, and use. In a nutshell, an HMM is a statistical model consisting of a collection of "states" and some transition matrices. HMMs are very useful for classifying patterns in sequences of data. Hand gestures are just that, and so HMMs are therefore useful for recognizing hand gestures.

The Accord.NET toolkit [35] contains many machine learning and statistical tools relevant to hand gesture recognition and the Accord continuous HMM classifier served as the basis of the gesture recognition system. Each HMM contained five states and forward topology. Empirical studies of increasing numbers of states and both forward and ergodic topology showed this choice to be the best balance of performance and accuracy. As the number of hidden states increased towards 20, performance (time to recognize and evaluate the gesture sequences) increased dramatically, such that real time recognition was no longer possible. Additionally, accuracy did increase discernibly

after five states, making five states the ideal balance of accuracy and performance. A simplified diagram of the HMM structure can be seen in Figure 10.

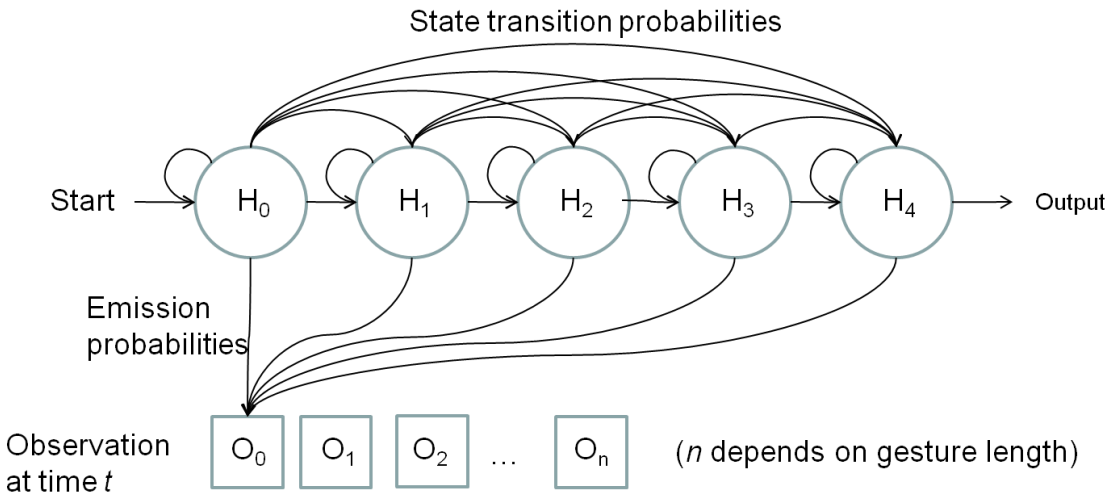


**Figure 10:** Simplified illustration of the HMMs used. Each arrow between states contains a state transition probability. Two states with no arrow means the state transition probability is zero. Note the forward state structure. The emission probabilities are simplified in this illustration.

The HMM is used to perform hand gesture recognition. A gesture consists of a sequence of hand poses over time. Each hand pose is a state consisting of the 18 finger angles captured by the CyberGlove. Rather than classify each pose, the HMM used the 18 raw values as the input, since the instantaneous pose classifiers would have to be trained for each user. Two of the bend sensors input values corresponded to wrist flexion and wrist abduction. As wrist motion was not intended to drive classification, they were omitted from the list of sensors, leaving only 16 input sensor values.

To ensure that one sensor didn't overpower another just due to the allowable range of motion, the data was normalized by dividing the data by the standard deviation

for each of the remaining sensor values. Keeping the original 16 data points per gesture frame caused very low online recognition accuracy, especially for gestures where part of the hand doesn't move (e.g., pulling the trigger of a gun). Small motions in the non-moving part of the hand could have remarkable effects on recognition. Principle Component Analysis (PCA) reduced the dimensionality of the data, both improving online recognition accuracy as well as performance. To summarize, PCA computes the covariance matrix of a sequence of data (here, the recorded data for each classifier) and uses the eigenvalues and eigenvectors to project the data into a lower dimension.

Projecting to slightly lower dimensions, such as 14-15, did not help online recognition accuracy and in fact occasionally caused errors in the decomposition of the covariance matrix. Using very low dimensions, such as 5, degraded the online recognition accuracy significantly. When examining the eigenvalues of the covariance matrix, the most significant drop occurred between values seven and eight. Using a value of seven worked fairly well for online recognition, but did degrade the accuracy of partial hand gestures (such as the trigger gesture as mentioned above). A value of ten seemed to work best in practice.

Training the classifier for a particular gesture required 60 examples per gesture per person. Three people trained the system, resulting in 180 examples of each gesture for training. Each gesture had its own trained HMM for use in recognition. The idea is that when a gesture needs to be recognized, all available HMMs will be fed the gesture data, and the gesture corresponding to the HMM with the highest confidence (or

probability) in generating that sequence will be selected, and any applicable actions corresponding to that gesture will be executed.

The XNA framework provides an update function that is automatically called at each time step of the game. The default time step size is 16 ms, or 60 frames per second. This frame rate is typically the target of most games as it is fast enough that even the fastest user interaction will not cause any perceivable latency. In order for interaction to be usable, recognition must occur significantly below this threshold. Therefore, any complicated recognition is out of the question or must be highly optimized. Fortunately, the Accord HMM implementation runs well within this threshold for small to medium sized models. If the number of states becomes high, HMM evaluation takes a significant amount of time, though it may be more accurate. Keeping the number of states at a lower yet reasonable number will keep performance high while still maintaining fairly high recognition accuracy. A forward topology model with five hidden states meets these criteria, achieving a high accuracy while evaluating the sampled data nearly instantaneously.

At each 16 ms time step of the game, finger data and hand orientation are sampled from the hardware. The most recent several seconds of gesture data are always available to the system for use in recognition to ensure that the system retains a long enough stream of data to recognize any gesture. At any time during the game, a certain set of gestures will be possible. The available gestures depend on the state of the player or the game. For example, if the player is holding a pistol, some pistol gestures, such as shoot, will be available. If no objects are active, then only basic gestures are enabled,

such as grab and release gestures. To perform gesture recognition, a sequence of hand poses sampled from the user is fed to each active model. The game then performs the action of the gesture corresponding to the model with the highest confidence.

The recognition accuracy was fairly high using models with five hidden states linked in a forward manner. As described later, four gestures were used in testing. Training data was collected for each of the four gestures and performed preliminary offline recognition testing to ensure the HMMs work properly. Using 4-fold cross validation on this training data, the system achieved 99.7% accuracy for each gesture (see Figure 11). In practice, the finger-gesture recognition works fairly well. It nearly always registers and recognizes gestures correctly, with an occasional misrecognition and occasional lack of recognition.

To show the power of recognition from finger bend sensor data only, the HMM uses only the CyberGlove sensor data for gesture recognition. This means that only the motion of the fingers will be used for each gesture. As the focus of this thesis is on interaction using the fingers, using only the finger sensor data highlights the how its use can improve the gaming experience. Of course, hand motion does play a part in interaction, as the player could not perform any actions without it. However, the motion of the hand is not used in recognition. Future work could include this feature, and indeed there is research in the area of gesture recognition that focuses on hand motion gestures.

Confusion Matrix: 4-fold CV totals

| Actual \ Recognized | Grab | Release | Shoot | Arrow Release |
|---|---|---|---|---|
| Grab | 540 | 0 | 0 | 0 |
| Release | 0 | 540 | 0 | 0 |
| Shoot | 2 | 0 | 536 | 2 |
| Arrow Release | 0 | 0 | 0 | 540 |

**Figure 11:** Confusion matrix for four-fold cross validation on data for four gestures. The total number of tests for each gesture was 540 (since 4 gestures = 720 training examples per gesture). The only few misrecognitions occurred with the shoot gesture.

## 7.2 Data Collection

A modified version of the data collection program developed and used by Paulson in [14] was initially used. The data collection interface allows the developer to easily define and record new gestures. The interface consists of two tabs: one for collecting gesture data (Figure 12) and one for setting up a collection of gestures (Figure 13).

To perform data collection, the developer will first choose the setup tab to create a gesture collection study. He enters the title, directory, poll time, and gestures themselves on this tab. A check box can be selected to randomize the gestures when collecting data. The developer can enter as many gestures as he wants as well as the

number of times to record each gesture. When the save button is clicked, an XML configuration file containing all this information will be created. A sample gesture collection setup can be seen in Figure 13.

The collection tab is used for actual data collection. Three buttons are present on the left side of the interface to select which devices to record. These include the left CyberGlove, the right CyberGlove, and the Flock of Birds (with 2 sensors). Clicking a button corresponding to a device toggles that device's connection, a status icon beside the corresponding button and the button's text. For example, the interface in Figure 12 is ready to record data from the right glove. Once the desired devices are connected, the developer clicks the "Load Study" button and browses for a configuration file created in the setup tab. When the study is loaded, a gesture name will appear on the right side of the interface. A series of buttons below the gesture name controls the recording and saving of the gesture data. The developer clicks the record button to begin recording the gesture. The study subject then immediately performs the gesture, and the developer clicks the stop button when the gesture has been performed. If the gesture was performed successfully, the developer clicks the right arrow button to save the recorded data and advance to the next gesture. Each time the right arrow button is pressed, a single XML file is saved for each connected device in the directory specified in the setup tab of the collection study. If the gesture was not recorded correctly, the record/stop buttons can be used to re-record the gesture.

The initial data collection system was developed by Paulson to record data from a single CyberGlove. The original intent of the program was to record static hand poses.

Whole arm interaction requires the ability to record dynamic gestures that vary over time. It also requires data from up to two gloves and data from the Flock of Birds. Since all the data must be recorded at once, the program was modified to record any configuration of the three devices. Because the interface was designed to record static hand poses, the mouse was a sufficient tool for selecting the record and stop buttons as static poses don't have a defined start and end point. Dynamic gestures do, however, and there were significant gaps between the actual end of the gesture and the end of the recording when attempting to record dynamic gestures using the mouse. Keyboard controls were added for the record, stop, and advance buttons so that they can be more quickly activated. When making this change, there was a slight increase in offline recognition accuracy and a large increase in online recognition accuracy.

This actually touches on the issue of gesture segmentation as described above. It is very important that the recorded training gestures be properly segmented to help optimize recognition. One area of future work could be to develop a gesture editing application that opens up a saved gesture from the collection program, plays back the gesture in 3D and allows the developer to cut off any beginning and ending lulls, or tails, that are not actually part of the gesture. This could be very similar to common video editing applications.


**Figure 12:** Data collection program collecting gesture data for the right hand.

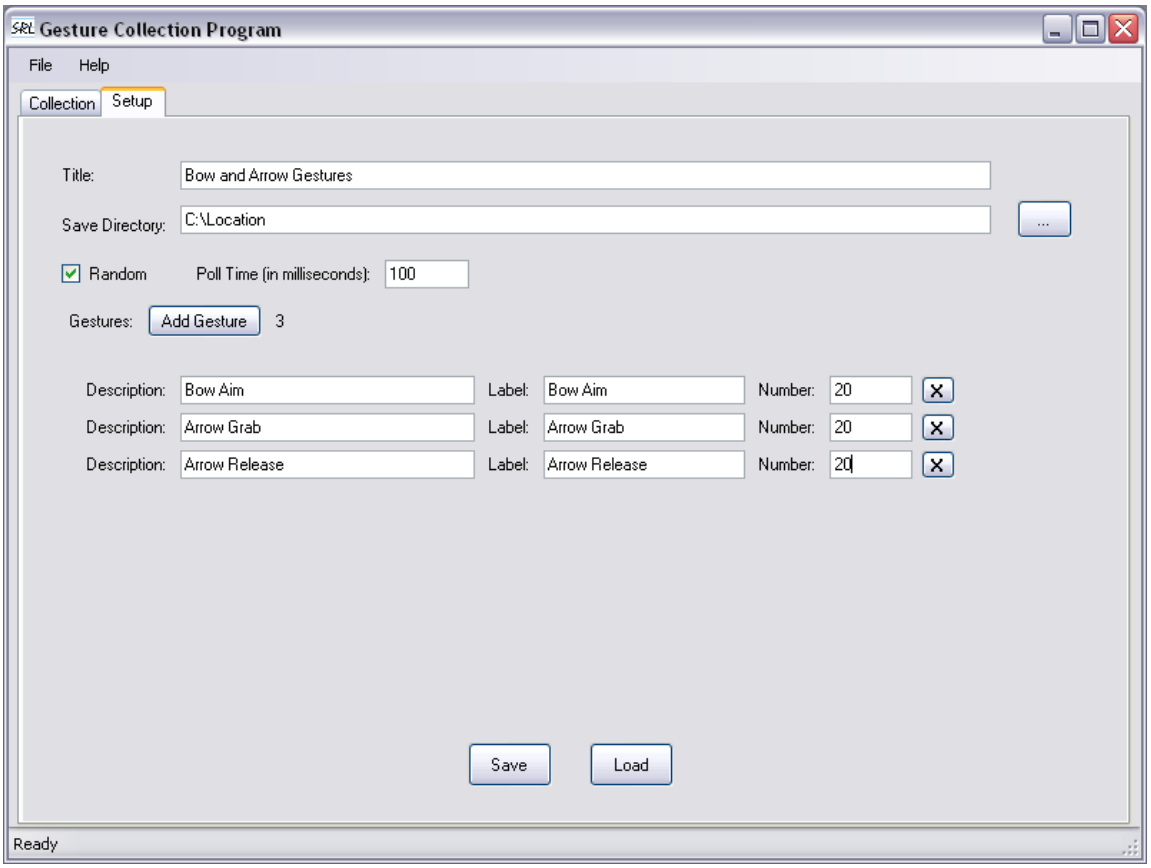


**Figure 13:** Defining a gesture set for "bow and arrow" gestures.

## 7.3 In-Game Recognition

To facilitate gesture recognition in-game, a recognition manager object was implemented to handle all recognition tasks and coordinate which gesture models should be evaluated for recognition. The recognition manager is created from within the hands object, so whenever the hands are used, recognition can be easily performed.

Though each HMM can evaluate well within the game's 16 ms time steps, recognition is performed every 100 ms to guarantee that the it doesn't overly tax the system. To match this recognition rate, all training gestures were collected using the

same 100 ms rate. Many mainstream, big budget games are highly complex and contain many components all required to execute within the time constraints of the game. By keeping the recognition system lightweight, it can be integrated with games of the future without sacrificing game performance.

The recognition manager keeps track of the most recent ten seconds of data in order to analyze the gesture stream. When recognition occurs, the game sends the current hand state consisting of both the finger and tracking data. The recognition manager adds this new state to the end of the saved gesture stream and deletes the oldest data.

The recognition manager contains a list of global gestures that can be performed at any time. When the player performs a gesture, this list of gestures is always evaluated and the results of each global gesture model (HMM) are added to a list of recognition results.

In addition to global gestures, the recognition manager contains a list of active objects and the gestures that are associated with those objects.

## 7.4 Virtual Environment

A 3D environment was required to be able to visualize the hands in 3D. The Microsoft XNA game development platform was desirable as it offered all the features necessary for 3D visualization including 3D model rendering and transformations. Since it is a game development platform, it could also be used to implement games and interactive elements as well (which are explained later).

The XNA platform allows the developer to create a 3D view and load 3D geometry for visualization. It also allows total freedom to manipulate world, projection, and model matrices to visualize a scene in any desired manner. A simple scene was used consisting of a ground plane, simple skybox for context, and constant lighting.

A simple 3D model of a hand was segmented into its various components: the palm, finger joints and thumb joints. The same hand model was suitable for both left and right hands due to its simplicity. To place the hand in the 3D space, the position and orientation of each component must be set. The method used for obtaining and setting the proper hand component values is described in the next section.

## 7.5 Hand Representation

As mentioned earlier, it is important that the player's hands be accurately portrayed in the game environment. Focused was spent on determining the position of the hands, orientation of the hands, and state of the fingers using the hardware mentioned previously. The hands were reasonably portrayed in the 3D game world.

To calculate the position and orientation of the hand in 3D space, the Flock of Birds 6DOF tracking system by Ascension Technologies was used. To summarize, it consists of a magnetic transmitter and one or more (in this case two) wired sensors. When a sensor is within range of the transmitter, the hardware is able to calculate the 3D position and orientation of that sensor with high precision.

The CyberGlove is physically designed to work in conjunction with the Flock of Birds and provides a Velcro mounting point for the sensors on the wrist of each glove.

This is a good method for tracking the hands as it does not involve any extra hardware to attach the sensors to the hands.

A custom GloveKinematics library (dll) was created to handle all Flock and glove related tasks. It calls functions from both the Flock of Birds and VirtualHand SDK for ease of use in other programs. One needs only to include this library and create a GloveKinematics object. It is then easy to set up and connect to all hardware. It contains functions to get the current Flock and glove sensor data. The VirtualHand SDK calculates the 3D positions of all hand joints which can then be used to position the 3D hand components.

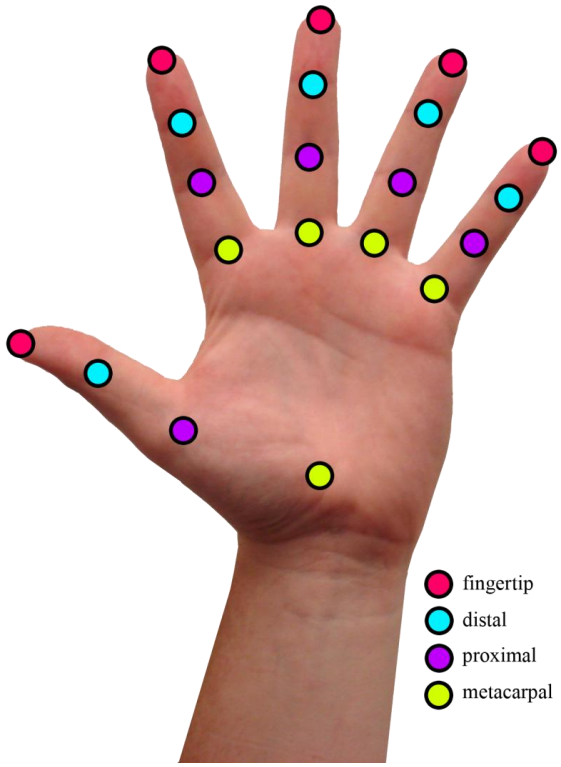


**Figure 14:** Each digit has four joint positions calculated by the VirtualHand SDK.

To place the hand components correctly in 3D space, the coordinates obtained from the GloveKinematics object were used. The palm was trivial to place, as it could be directly placed and oriented. The fingers were more complex to position since they are offset from the palm and then translated to the world position. Each finger and thumb has four coordinates corresponding to the four finger joints: metacarpal, proximal, distal, and fingertip (see Figure 14). Since each finger has three components and each component is spanned by two joints and the center of each component lies at the center of each 3D model, the joint data was manipulated to position each finger component. The positions of each pair of adjacent joints were interpolated to find the center point of each component. Each component's orientation was calculated by rotating along the flexion and abduction axes by the angles of the line formed by two adjacent joints.

Once each finger and thumb component's position and orientation relative to the palm was set, all hand components were transformed by rotating about the palm origin by the Flock orientation data. All components were then translated by the Flock position data. At this point, the hands moved around the 3D space similarly to the player's hands.

The scale of the hands was set so they mapped more closely to the player's hands. The game camera was positioned where the player's head would generally be in a normal FPS game. The virtual hand positions were offset to position them relative to the player's view. For example, if the player places his hands at his sides in a restful position, the virtual hands will move below the bottom of the screen out of view. If the player raises his hands above his head, the virtual hands will move to the top edge of the

screen. As the player moves his hands in his real world field of view, the virtual hands will be visible on the screen.

From personal experimental experience, this mapping of the real world hands to the virtual hands feels natural. It feels easy to use the virtual hands without looking at your own hands. The initial test results seem to confirm this ease of use through other players' responses to the natural feel and ease of use in playing the FPS game.

**Figure 15:** A player observes the 3D hands being replicated in the virtual world.

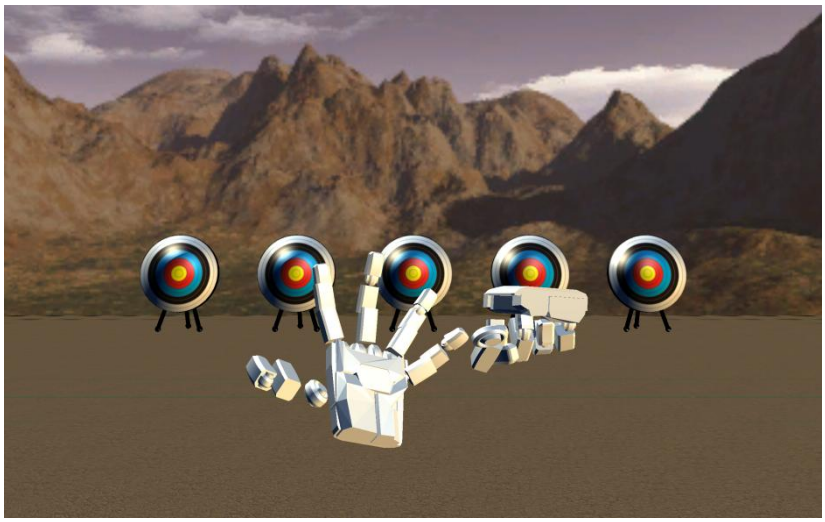

**Figure 16:** Screenshot of the player in Figure 15.

## 7.6 Game Interaction

Once the hands are properly visualized and respond to player input accordingly, some action must be taken to be able to interact with the virtual world. The fundamental tasks of grabbing and releasing combined with the hands' natural motion were used to facilitate interaction. Depending on the type of game being played, additional object-specific gestures may also be available.

To add some more realism to the games, a simple physics system called JigLibX [36] was used. This allows objects in the world to react naturally to player input.

Before beginning the game, the player puts on the two gloves and attaches a tracking sensor to each wrist. Currently the player must stand in a fixed position in front of the screen to ensure proper tracking. When the devices are connected, the two 3D hands appear on the screen. As the player moves her hands and fingers, the on-screen hands mimic the player's exact movements. The 3D tracker positions the hands on the

screen in the same general locations relative to the character as the player's hands move in the real world. The gloves capture the player's finger movements, and the 3D hands' fingers move identically to the player's fingers. The player's hands are effectively replicated in the 3D world, allowing realistic interaction. Figure 15 and Figure 16 show examples of 3D hand replication in a gaming environment.

## 7.7 Video Game Implementation

Once the player's hands were visualized, the players required a way to interact with a virtual world. A choice was made between modifying an existing game (such as the popular Half-Life 2 as modified in [37]) or developing a custom game from the ground up. When considering this, it was noted that the fundamental game design depended on hand interaction. Since no mainstream games, even those that support modification, are designed with hand input in mind, it was determined that modifying a game would require just as much, or more, work as implementing new games. Therefore, it would be best to implement new games from the ground up. The Microsoft XNA Game Studio was used because it includes extensive libraries for 3D visualization and game mechanics as described previously. XNA is designed to make the game development process simple in order to allow independent game developers to easily create games. It was therefore an excellent choice for developing experimental games to incorporate hand visualization and interaction.

## 7.8 FPS Game

Two simple games were created that are playable with the hands. The first game is a first-person shooter game that lets the player wield a variety of weapons and use them against some targets (as seen in Figure 17). The purpose of this game is to test the player's ability to access items, perform context-sensitive gestures, and uniquely wield and use weapons using the hands as the video game input.

To play the game, the player can access any of several weapons by grabbing the weapon from its imaginary "holster" located somewhere on the player's body. There are 10 such holsters, 5 on each side of the body. These holsters consist of a pistol holster located at the hip of the same side as the hand, a sword scabbard located at the hip on the opposite side of the hand, a pistol holster located on the side of the torso opposite the hand, a belt holster on the same side as the hand, and a shoulder holster on the same side as the hand. All these holsters can be seen in Figures 18-20.

**Figure 17:** Screenshot of a player aiming two pistols at the targets in the FPS game.

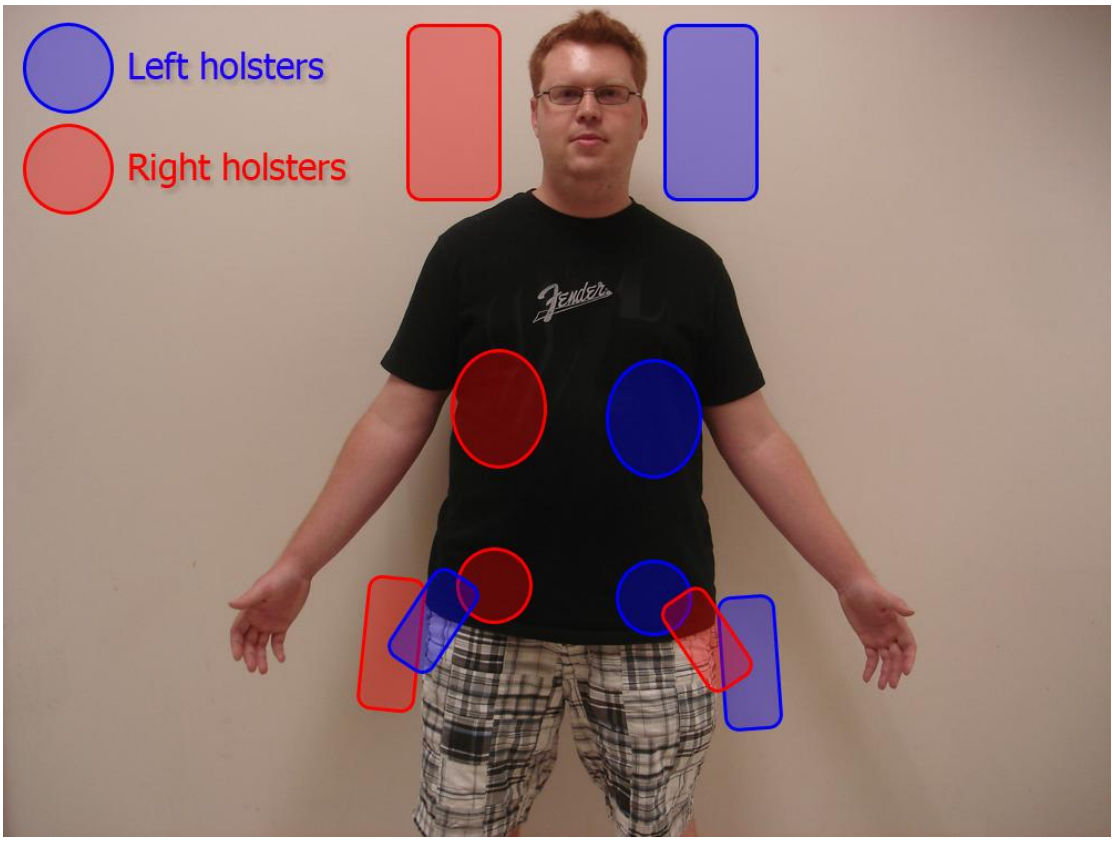


**Figure 18:** Illustration of the ten holsters across the body. The right hand accesses weapons in the red areas and the left hand accesses weapons in the blue areas.

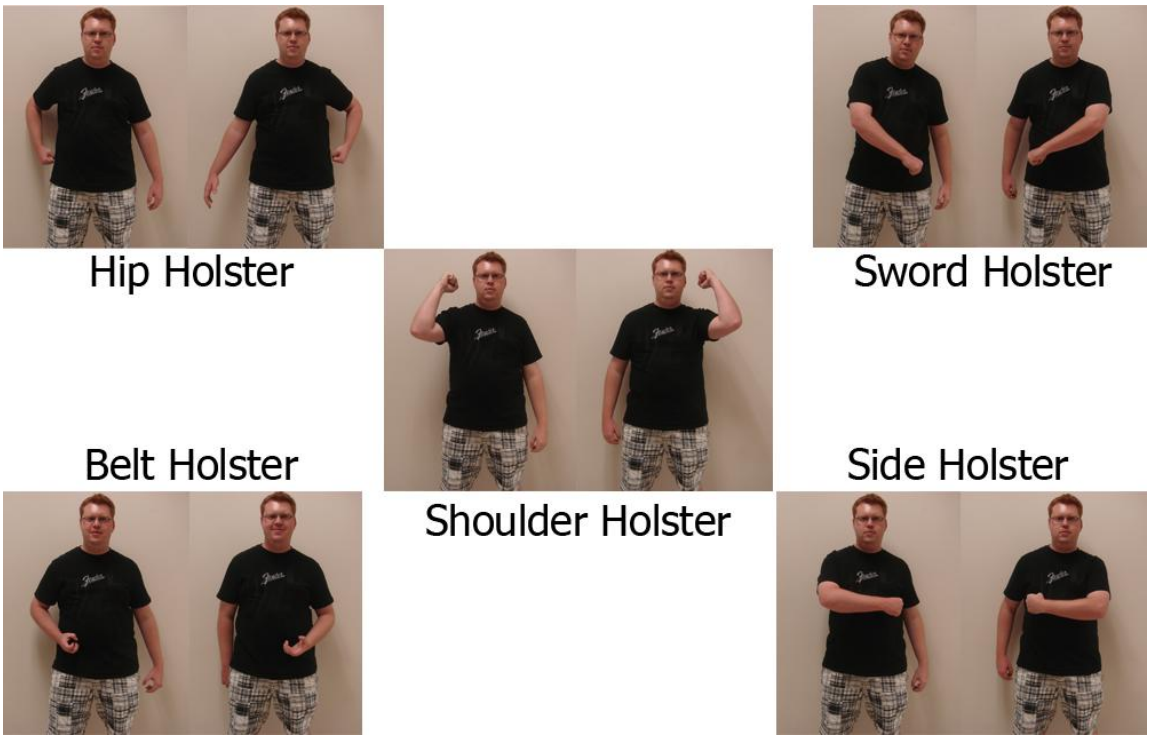


**Figure 19:** Examples of accessing each holster using both the left and right hands.

**Figure 20:** Sequence of motions and gestures used to fire a bow and arrow.

**Figure 21:** A player grabs a pistol from its holster in the FPS game.

The player accesses a weapon by grabbing in a holster area as seen in Figure 21. In this case, the hand's orientation in addition to the finger gesture is used for recognition. To determine the proper orientation, 15 examples of the grab gesture were recorded for each holster area and the average orientation as well as the standard deviation of the orientation components were calculated. The average value and standard deviation values are calculated independently for the X, Y, and Z values of the orientation. When a grab gesture is performed to access a weapon, the hand’s orientation is sampled in addition to its finger angles. The distance of the sampled orientation to

each saved average orientation are calculated by subtracting the X, Y, and Z values and using the absolute value. The hand is in the correct orientation if this distance from the average orientation is within two standard deviations of the average orientation. This is the only scenario in which orientation is used in conjunction with the gesture. The orientation data is only used as a trigger to perform selection and is not actually input into the HMM classifier. All other gestures use finger motion only.

Each of these "accessor" gestures can be performed by either the right or left hand. The accessor gestures are identical for each hand except that they are mirrored, i.e. performed oppositely for each hand. This is a novel type of interaction that makes use of the whole-arm input. It is designed to be more natural than selecting a weapon by scrolling, pressing a number key, or bringing up a menu of weapons. The hands will grab each weapon in a unique manner for how they are stored on the player's body.

The weapons contained in each holster are configured by the developer. Either single-handed or two-handed weapons can occupy each holster. The player can hold two different single-handed weapons simultaneously, and if one hand contains a single-handed weapon, the other hand can still be used to access another single-handed weapon. If a holster contains a two-handed weapon, the player can only access that weapon if he is not currently holding any weapons. If a two-handed weapon is held, then no other weapons can be accessed until the weapon is put away. Any weapon can be put away by performing a "release" gesture that consists of opening the hand. In the case of two-handed weapons, either both hands must perform the release gesture simultaneously or one dominant hand can release the weapon.

Once a weapon is held by the player it will be displayed in the virtual hands. She can then use that weapon as it is used in the real world, and the virtual weapon will be placed in the virtual hands as a real weapon would be placed in a person's hands.

Because two single-handed weapons can be used simultaneously, this means that the player has independent control of the weapon in each hand. Few gaming systems allow this versatility. For example, many games allow the player to "dual-wield" pistols, but they are still used as one weapon and both aim at one single point. In this whole-arm interaction implementation, dual wielding pistols means that each gun can be independently aimed and used. Obviously, two different types of weapons could be used at the same time. For example, the player could be using a pistol with one hand and pull out and throw a grenade with the other without releasing the pistol. Similarly, the player could use a sword for fighting close-range enemies and a gun for long range fighting simultaneously. This can make for intriguing and immersive game play.

When used in this manner, the weapon can be used in a more natural way than with other types of gesture controls. Different types of guns can be used in unique manners instead of one manner for each gun, for example. Guns and other types of firing weapons can be precisely aimed using realistic arm motion. Grenades and other throwable objects can be thrown, and the parameters of the throw can be determined from the player's actual throw gesture. None of this is possible using current gaming inputs because these actions require the use of the fingers. Without the fingers, a gun cannot be realistically gripped unless the controller is shaped like the same weapon being used. It is uneconomical and cumbersome to produce a unique controller for each

unique type of weapon, and a throwable controller is largely useless. In addition, vision-based controls don't even allow proper wielding of weapons which renders them useless for such applications.

While a weapon is held, the player can perform gestures associated with that weapon. For example, the player can pull the trigger on a pistol as seen in Figure 22, release an arrow with a bow and arrow, or squeeze the handle on a flamethrower. When a gesture is performed, the virtual objects will react accordingly. The pistol will shoot a bullet, the arrow will fly, and the flamethrower will spew fire.

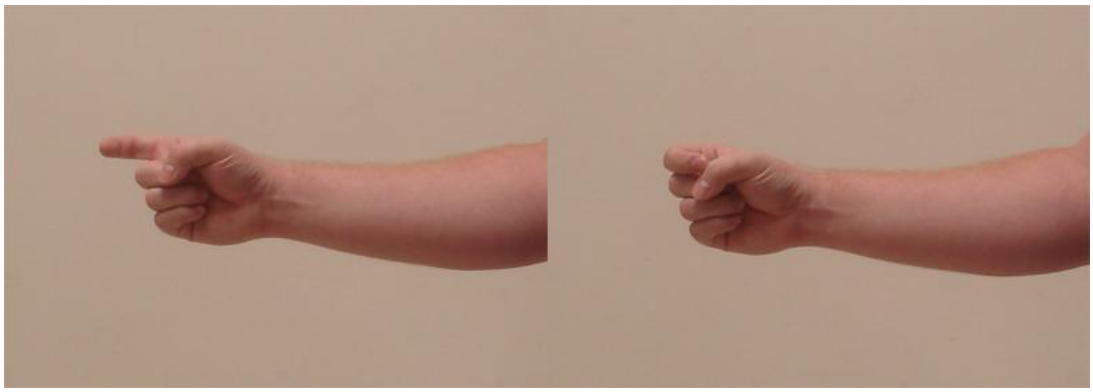

**Figure 22:** Gesture used to fire a gun.

To give the player some tasks to perform, five stationary and three moving archery-style targets are provided for the player to fire at. Some of these targets can be seen in Figure 23. The player can use any gun for target practice. When the player fires a gun, the game traces the bullet path using a ray originating from the unique gun muzzle. If the bullet path intersects a target's face, it counts as a hit and the score is calculated.

The score is highest at the target's bull's eye and lowest at the edge. When a target is hit, it disappears. When all targets disappear, the target practice game is over and the scores are displayed.

**Figure 23:** A player plays the FPS game.

During development, many types of weapons were experimented with, including pistols, submachine guns, bows and arrows, grenades, Frisbees, rocks, flamethrowers, and rifles. Each of these could be used in its own unique way.

## 7.9 Driving Game

The FPS game explores the obtaining of unseen items and the usage of held items in a virtual world. However, a virtual world might also contain objects the player can approach and interact with, such as a control panel. To explore manipulation of objects existing in the world, a simple driving game was implemented in which the player can manipulate a virtual steering wheel and throttle control. In this game, the set of hand gestures is limited to grab and release gestures. The hand motion will determine the analog values of the controls, i.e. the steering angle and throttle percentage. The steering wheel is used in an identical manner to a real car. The throttle control, however, is based on a modified shift lever since the focus is whole arm interaction.

Upon starting the game, the player is sitting in a car on a racing circuit. The car's dashboard is displayed in front of the player and the steering wheel and throttle are within reach. The player can grab hold of the steering wheel and throttle control and manipulate them.

To steer the car, the player simply moves her hand in a steering motion while the steering wheel is grabbed. Upon grabbing the steering wheel, the initial angle of rotation about the steering wheel's rotation axis is calculated and saved. As the player's steering hand rotates around this axis, the steering angle is calculated and the steering wheel steers the car accordingly. The physics system causes the simulated car to steer itself realistically.

To use the car's throttle, the player grabs the throttle stick in the same manner as she would grab the shift knob in a real car. To accelerate the car forward, the player pushes the throttle stick forward. This action causes a forward force to be applied to the vehicle. To brake the car while it is moving forward, the player pulls back on the throttle control. This action actually applies a reverse force on the car, so if the throttle is pulled back when the car is at rest it will begin to reverse. By holding back on the control, the player can accelerate the car in reverse. Pushing forward on the throttle while the car is in reverse will cause the car to slow as it accelerates forward. Figures 24 and 25 illustrate the driving interaction style.

The car will react appropriately to the steering and throttle, and the player can then drive the car around the track. To let go of the steering wheel or throttle, the player simply performs the release gesture or moves the hands far enough away from the objects to no longer be able to interact with them. The distance for deactivating an object is a "bubble" twice the size of the object.

**Figure 24:** A player plays the driving game using hand gestures and motion.

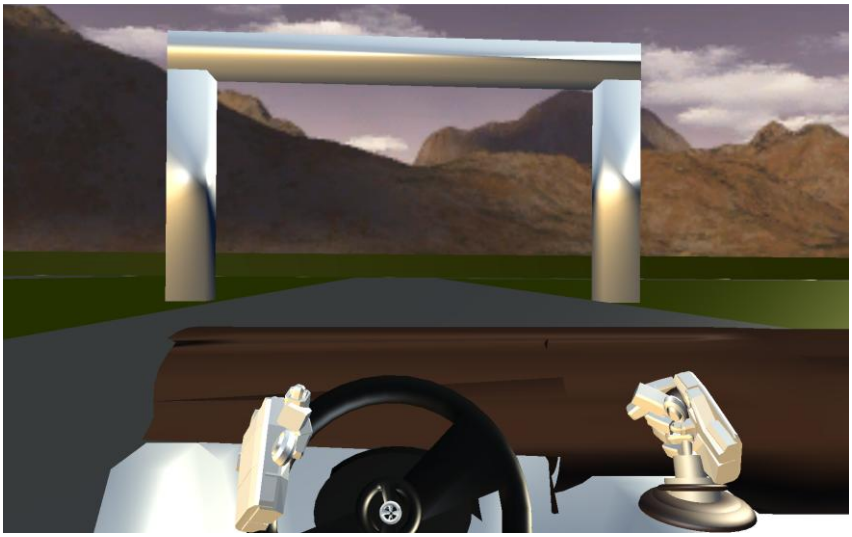

**Figure 25:** Screenshot of a player playing the driving game.

## 7.10 Menu Navigation

Though menu navigation is typically regarded as unnatural interaction, most games inevitably have a menu system to select options and set up the game. Game designers will undoubtedly incorporate menus into their gesture-based games. All current gesture-based video game consoles have games that contain a menu system. The consoles themselves even have integrated menu systems. From the author's personal experience, it can be difficult to navigate menus using point and click controls in the air or hand waving and dwelling. In these cases, existing directional controls or a physical controller is to navigate the menus.

Since the ultimate goal is to design a game where the only controller is the hands, a simple, natural way for the player to navigate a menu system has been created. It eliminates the intermediate control of a handheld controller and allows the hand to directly select menu items. It also eliminates annoying dwell time that can cause fatigue and irritability.

Upon starting the game, a main menu is displayed. The player's hand movements will be projected to the 2D screen plane, and the 3D hands will be positioned according to this projection. Figure 26 shows a player interacting with the menu system. Essentially, the player will move his hand in the air corresponding to the X and Y axes of the screen. When a virtual hand hovers over a menu item, the menu item will become highlighted to indicate its selection, as seen in Figure 27. To activate the menu item, the player performs the grab gesture, as if he is actually taking hold of the menu item. This allows menu items to be quickly selected in a fluid manner and should ensure a quicker,

more responsive, and less frustrating experience when compared with other gesture input systems that require some dwell time to activate a menu item. For simplicity, only the left hand was enabled for menu item activation, though either or both hands could be enabled for menu interaction.

To project the player's 3D hand position to the 2D screen position, the 3D Flock of Birds coordinates were manipulated. The Flock gives position coordinates on its X, Y, and Z axes. The Flock's X and Z axes are parallel to the screen's X and Y axes, so the hand is placed using the 2D coordinates formed by the X and Z axes and use a constant value for the Y axis.

When hovering over a menu item, the center point of the palm is used for checking if the hand is over an item. If the palm position is within the bounding box of a menu item, it becomes selected.

To select a menu item, the player performs a grab gesture. Because the menu system uses the same hands object as the game, the same gesture recognition system is constantly running. If the player performs a grab gesture while a menu item is highlighted, the action associated with that menu item will be taken by the game. The grab gesture is the only gesture used in the menu system.


**Figure 26:** A player navigates the menu by moving his hand to select a menu item.

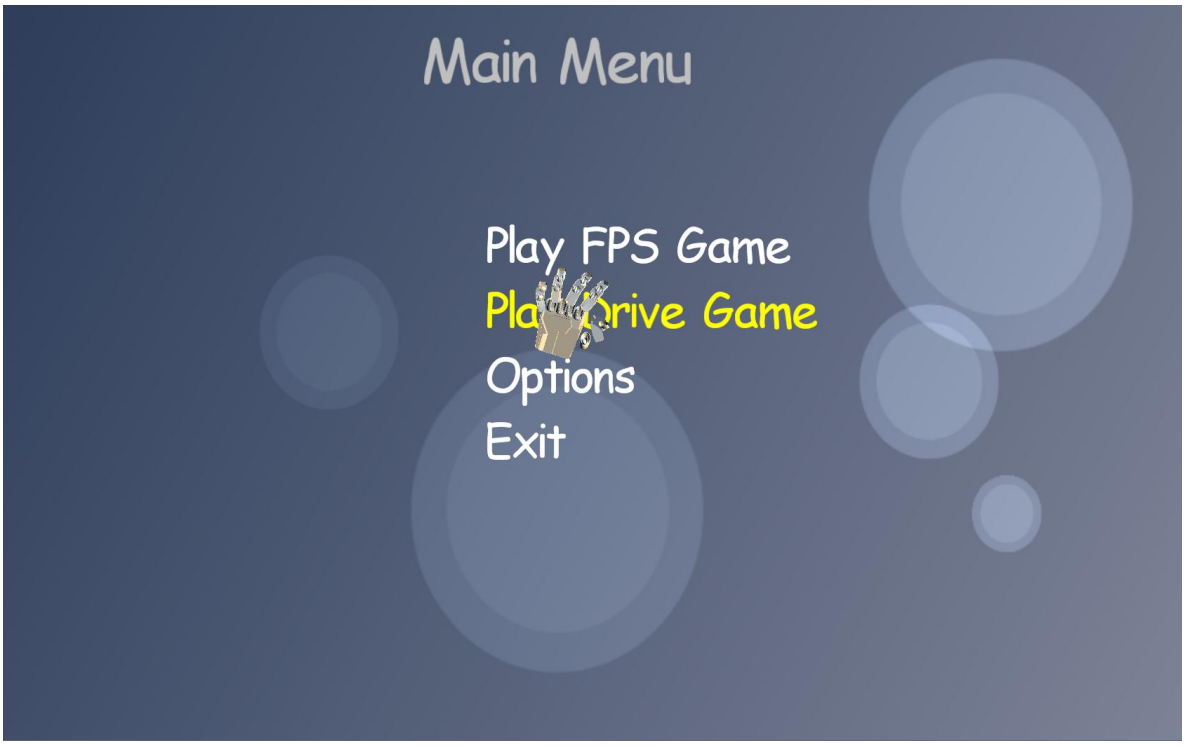


**Figure 27:** A player selects the "Play Drive Game" menu item. Note the hand hovering over the highlighted menu item.

# 8. USER STUDIES

## 8.1 Data Collection

A “dynamic gesture collection” tool was used that takes simultaneous input from two gloves (left and right hand) and the Flock of Birds at once over time. This allows easy capture of any gesture and creates simple XML files from which the data can be extracted to train the HMMs. Data was collected as mentioned previously.

## 8.2 Test 1: Simple Interaction

Four people played the games to test this method of interaction. This section describes the tasks the players were to perform in each game and in the menus. As each game was begun, the players were briefed on the types of interaction and gestures that can be performed. The goals of each game were then presented. The players were given as much time as desired to experiment with the game and get familiar with the interaction style and the gestures. When the player felt ready, he then performed the tasks for the game.

Upon starting the game, the player is presented with a menu containing these menu items: "Start FPS Game", "Start Drive Game", "Options", and "Exit" as shown in Figure 27. The player must first select the "Start FPS Game" menu item to begin the FPS game.

To play the FPS game, the player must take out any gun of choice and attempt to shoot some targets. The first round consists of five stationary archery targets that the

player must shoot. When a target is hit, a score is recorded for the target and it disappears. The score is calculated based on the distance from the target's bull's-eye.

Once all targets in the first round are hit, the second, more difficult round begins. Three new targets appear and they move alternately left and right. The motion of the targets tests the player's ability to quickly move the hands to aim. These tasks are designed to test both the gesture interaction as well as the possible accuracy achievable with this type of interaction. Once each task is complete, the scores were logged for later analysis. After the target practice tests are complete, the player may optionally perform the tests again or simply experiment in playing the game. Several other weapons, such as the bow and arrow, grenade, and flamethrower may be tested. Figure 23 shows a player playing the FPS game. Once finished, the main menu is opened. The player will then select the Driving Game menu option to begin playing the driving game.

For the driving game, the player must complete as many laps around the track as desired. The track contains a start/finish line, and a stopwatch starts when the player crosses it and passes it each lap. When the three laps are done, the lap times are recorded for later analysis. The player then opens the main menu as before and selects the Exit menu item to close the game.

The players could optionally play either game for fun after all tasks were completed before the game is exited. Once the player exits the game, he or she was interviewed to gauge thoughts on the type of interaction and the games. The players also provided opinions and suggestions on how the games and interaction style might be improved. The results from the target practice and lap times allowed quantitative

analysis while observations of game play and player comments gave qualitative feedback usable for future work.

## 8.3 Test 1 Results

All players greatly enjoyed the menu navigation using hand gestures. They found it intuitive for the menu item under the hand to be selected, and all players liked the use of a grab gesture for menu item selection. Two players expressed that the grab was much more satisfying than the dwell time found on other gesture menu controls such as the Kinect. The only problem any player had was an occasional accidental selection of a menu item immediately above or below the intended menu item. This was due to slight hand movement caused by the grab gesture. In a few cases this slight movement was enough to allow the hand to hover over a different menu item. This small issue can be remedied by increasing the spacing of the menu items. In general, all players found the menu interaction very intuitive and fluid. One player suggested that a "fast" or "swipe" gesture could also be used to select menu items since the actual hovering over the items was generally done in a slower manner.

In addition to the positive feedback on menu interaction, all players equally enjoyed the FPS game interaction. All players immediately understood how to control the game and began interacting before the directions on how to play were finished being explained. In particular, players greatly enjoyed the concept of accessing different weapons by grabbing them from different points on the body. One player commented that it felt much more natural than pressing the 1-9 keys on a keyboard, as in many

computer games, to select a weapon. The players thought that the mapping of weapons to a physical location was much more natural than artificially selecting a weapon using other forms of interaction. Players also felt that the act of shooting the guns was very natural and noted that the "trigger" gesture was very similar to real-life shooting. Several players reported feeling very satisfied at the game's response to the physical input of the gesture. Some players suggested some improvements to the act of aiming the guns, such as snapping the camera to the top of the gun and using hand motion to aim the camera or projecting a crosshair onto the targets to get a feel of where the gun is aiming. No players reported any fatigue during the FPS game, and one player even commented that the FPS game allowed rest.

From observation, the players seemed to have difficulty aiming at the targets. A few players also complained of difficulty aiming the guns. However, no player reported overall frustration with the FPS interaction, and in fact all players highly praised the FPS game for its fun and realistic interaction.

All players had mixed reviews of the driving game. Each player demonstrated a longer time to learn how to drive the car than how to play the FPS game. Both games directly map objects to the hands, and the FPS game uses Weapons in a direct and absolute manner. In the driving game, however, players did not enjoy the absolute controls as much as they had in both the menu navigation and FPS game. Typically, while driving the car, the players' hands tended to drift away from the locations of both the steering wheel and the throttle control, resulting in unusual steering and a decrease in throttle. One player steered the car by simply moving the hand left and right instead of

pivoting around the center of the steering wheel. Another player desired to steer the car using an open palm approach instead of properly grabbing the wheel, which resulted in only occasional control. Most users reported some fatigue in the throttle hand due to the fact that the throttle hand must remain in a static position to keep the car moving. Some users expressed a desire for physical devices to grab onto in order to both rest the hands and provide physical feedback on the state of the controls. Despite difficulties with the controls, all players reported having fun with the driving game.

Though all players were told to drive as many laps of the track as desired, only one player did two laps while the rest did one lap. The fastest lap clocked in at 2:42.33 while the slowest lap was 5:50.0. The average lap time was 4:00.31. In the case of the player who performed two laps the second lap was much slower at 3:33.7. This is indicative that such constant manipulation tasks such as driving can quickly cause fatigue. This has been demonstrated with other hand interaction systems as well.

In general, all players enjoyed finger-based interaction versus current methods of video game interaction, including both gesture and non-gesture based control methods. Each player had video game experience using a variety of controller types, including keyboard and mouse, game pad (such as Xbox or PlayStation controllers), Wii Remotes, and Kinect. They found the finger interaction very intuitive and easy to learn. Each player thought this form of interaction would be very good for certain game genres, especially the FPS genre. They commented that this method of interaction could be a very natural method of control in such genres.

Most users also commented that some form of tactile feedback might be helpful and help increase immersion in both games. For the FPS game, players suggested some form of feedback when a gun was fired to simulate the kick of the gun. In the driving game, a few players commented some form of tactile feedback could help cue the player on where exactly the controls are and help keep the hands in the proper places while driving. Players never requested that physical props be used with the exception of the driving game in which the players wanted some object to rest the hands upon while driving.

Several players expressed interested in future development of the games and wished to play enhanced future versions.

## 8.4 FPS Game Modification

While the results from the first test were encouraging, one aspect of the FPS game that could be improved: gun aiming. All players reported that they greatly enjoyed playing the FPS game and that it felt natural and intuitive, but they all did suffer from a lack of accurate aiming. It would be interesting to solve this aiming problem, since an FPS game would be useless if the players could not aim a weapon.

In most current FPS games, the character's hands are shown holding a weapon along the bottom edge of the screen. Typically, a crosshair is also drawn in the middle of the screen so the player knows where she is aiming. In some other games the player can press a button to allow the weapon to be brought up to the center so the player can look down the sights of the weapon to aim precisely.

To be realistic, there should be no crosshairs in the FPS game. The other solution was to somehow allow the player to look down the sights of the gun to aim precisely. The only way the player could do this before is if he held the gun in the center of the screen. Indeed, most players were able to get a high score on centered targets. A problem arises if the player wants to aim at an object not in the center of the screen. Usually, the player can rotate the character to place the target in the middle of the screen. Since only the hands are tracked and not the player's body, this cannot be done. To overcome this limitation, a method was used for the player to lock the camera to the back of the gun so that the camera is always looking down the gun's sights. The player would still be able to move her hands to aim the gun, but the camera would be fixed to the gun, so no matter where the gun is aimed, the player can accurately fire at any target.

The next problem was how to engage this aim mode, which will be referred to as "weapon sight aiming." The solution to this turned out to be simple. When a person wants to aim down the sight of a gun, he raises it up to the level of his eyes. In the FPS game, the level of the player's eyes is approximated by the center of the screen. Therefore, the player should be able to raise the weapon to the center of the screen and line up the sights, then the camera can snap to the gun. Now, when the player moves his hands around, the camera will remain attached to the gun so she can aim in any direction by moving her arms and hands.

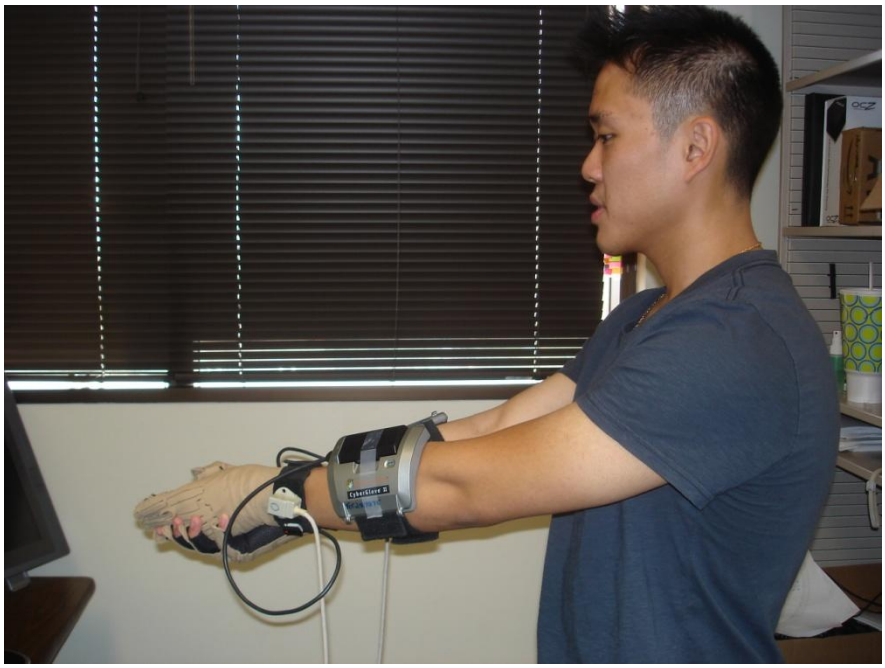

**Figure 28:** A player uses weapon sight aiming to accurately shoot targets. Note both hands holding the gun.

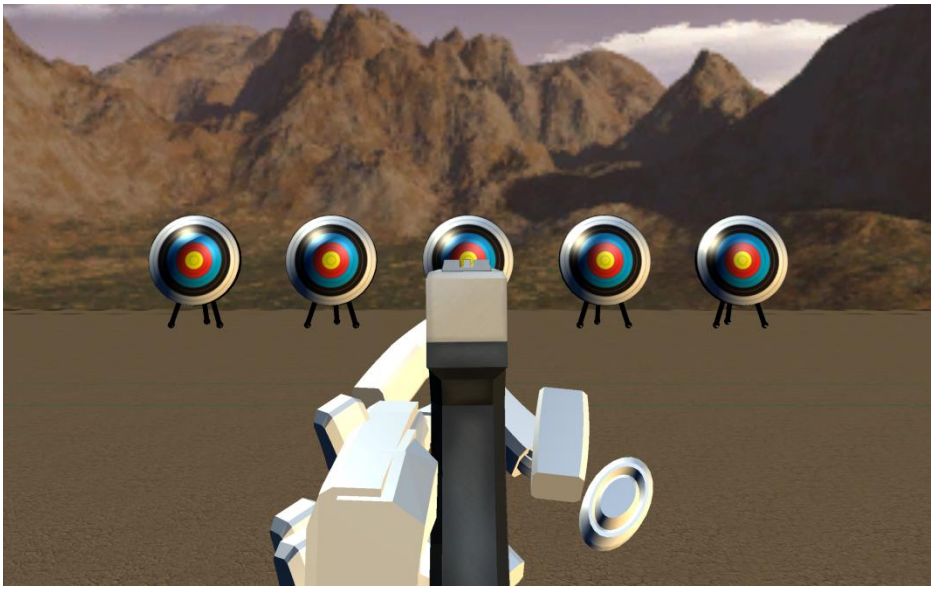

**Figure 29:** Screenshot of weapon sight aiming. The camera is attached to the gun.

A new problem is introduced if the player activates weapon sight aiming in this manner. What does the player do to disengage weapon sight aiming once she is done using it? Just placing the arm back at her side would mean the camera would point at the ground. To solve this problem, both hands to be in close proximity to one another. This is accomplished by the player only using one gun (in either hand) and holding the empty hand as if she is holding the gun with both hands. This is natural because when holding a real gun to look down the sights, two hands are typically used. Indeed, many existing games that allow this form of aiming display both character's hands holding the gun.

Now that both hands are used to "hold" the gun in weapon sight aiming, the more precise aiming mode will be disengaged if the hands are moved apart. This will occur naturally if the player puts one or both of his hands to his side when he is done with weapon sight aiming. When the hands move apart, the camera snaps back to its original position of the character's view. Figure 28 shows a player using weapon sight aiming, and Figure 29 shows a screenshot of this aiming mode.

## 8.5 Test 2

To test weapon sight aiming, three people played the FPS game. Two of these had previously participated in the first test, and one had not. All players had video game experience, and some had experience with other types of gesture interaction.

In this test, the players became familiar with weapon sight aiming as well as learned or remembered how to interact with the FPS game. The players navigated to the FPS game in the main menu and then pulled out a pistol to perform target practice. Each

player hit all stationary and moving targets using normal aiming, and the scores were logged. They all then shot each stationary and moving targets using weapon sight aiming, and those scores were logged as well. Each player was allowed to perform target practice as many times as desired, and each player performed it two or three times.

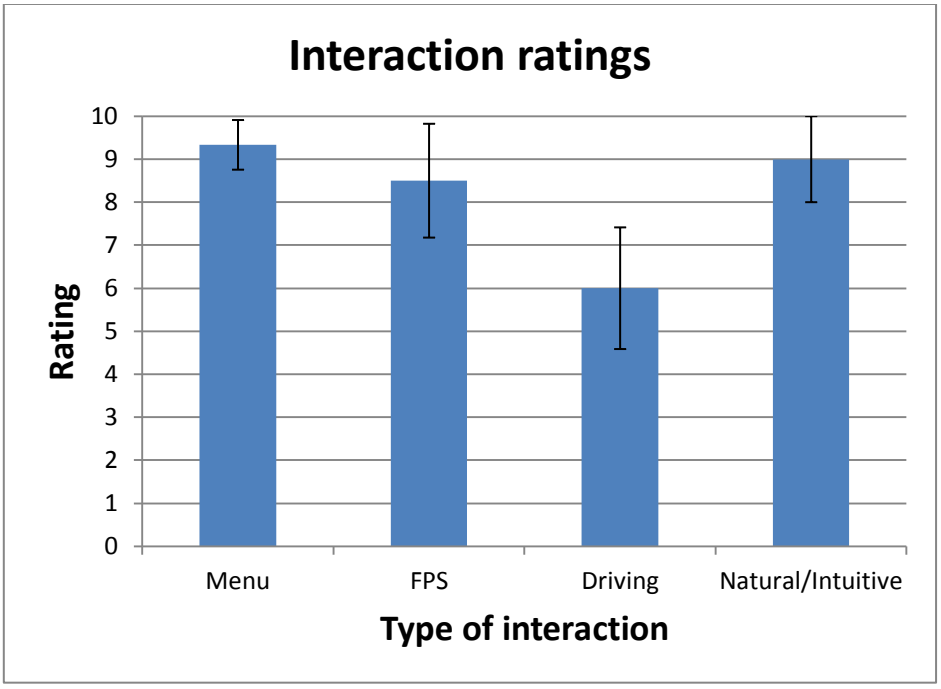


**Figure 30:** Ratings for various types of interaction in this implementation. Bars are the mean ratings and error bars are the standard deviations of the ratings.

When each player was finished playing, he or she rated the menu interaction, FPS game interaction, and the overall natural/intuitive feeling of the games on a scale from 1 to 10. For those who had previously played the driving game, they rated it as well. The menu interaction, FPS game, and overall natural/intuitive feel gained very high marks. All players commended this form of interaction for being natural, intuitive, and

fun. The driving game received lower marks, but players still said it felt intuitive but was tiring. A summary of the ratings can be seen in Figure 30.

## 8.6 Test 2 Results

As previously mentioned, the players' scores were recorded for both stationary and moving targets when performing target practice. The maximum number of points per target was 50. For stationary targets with normal aiming, the mean score was 25 with a standard deviation of 14.8. For stationary targets with weapon sight aiming, the mean score was 44.3 with a standard deviation of 6.8. For moving targets with normal aiming, the mean score was 32.8 with a standard deviation of 11.3. For moving targets with weapon sight aiming, the mean score was 27.8 with a standard deviation of 12.6. These mean scores and standard deviations are graphed in Figure 31.

While the differences in mean scores are technically not statistically significant, we can observe a clear improvement with stationary target scores. There is not much change within the moving target scores, which may have to do with the fact that players tended to attempt to hit moving targets at an angle instead of waiting for a moving target to pass over the exact center of the screen to get a high score. In a real game, the player generally cannot sit still and wait for an enemy to move into place to be shot. Also, only scores for each target hit during the test were noted. While the player's were shooting, a clear improvement was observed in shooting speed and a reduced number of missed shots. Finally, all players said the weapon sight aiming felt more natural and made it

easier to aim. It is therefore clear the weapon sight aiming improved the gaming experience.

This is an example of an innovative solution to a problem introduced by the limitations of the hardware. Because the player's head and body are not tracked, he will inherently have trouble aiming if he must remain focused on the screen while his hands move to aim. The weapon sight aiming solution is interesting in that it is a less natural motion than true aiming but all players instantly learned how to use this type of aiming. Therefore, it remained an intuitive and somewhat natural solution to an interesting problem.

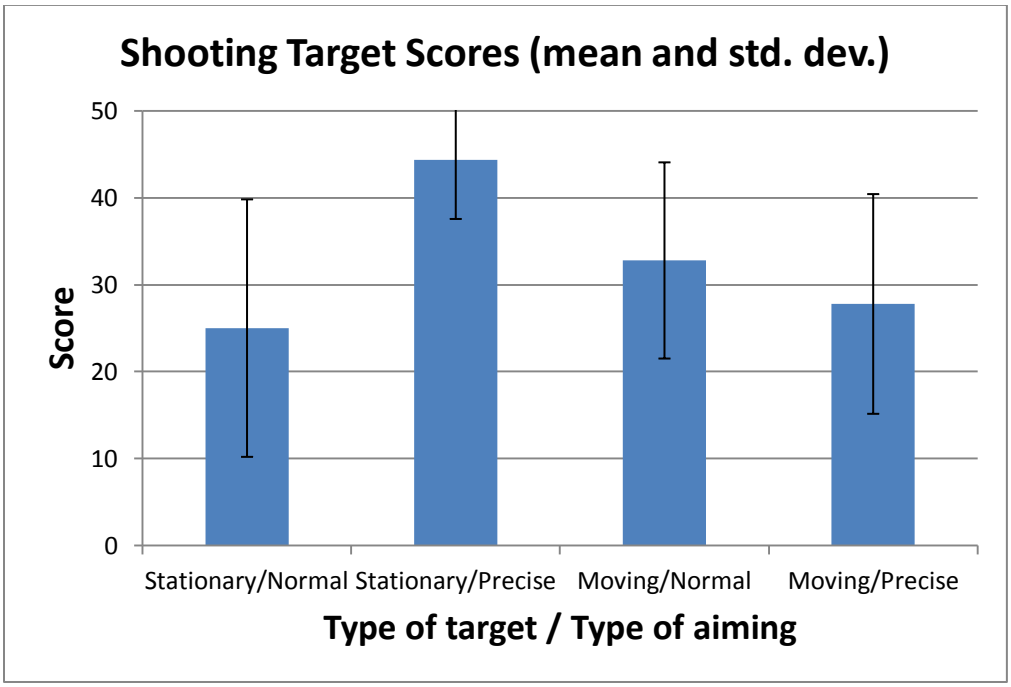


**Figure 31:** Mean scores and standard deviations of target practice.

# 9. DISCUSSION

An arm, hand, and finger-input gaming system that uses hand gestures to perform in-game actions has been successfully implemented. This implementation bridges the gap between existing gesture controls that lack finger input and complex virtual reality approaches that exactly mimic the players. The approach mimics the player's hand and arm motions and allows the player to interact with a virtual environment.

This system is perhaps one of the first of its kind that has been successfully implemented. To stress, this system is functional and can be the basis of a full gaming system. The gesture recognition and segmentation works in real time and achieves high in-game accuracy. The games respond accordingly to player input and allow unique interaction using both hands simultaneously for different tasks.

Whole-arm interaction is highly suitable for some types of games, and other types might need future development for their types of interactions. In addition, this type of system can be implemented with today's hardware. Future hardware improvements may further enhance whole-arm interaction.

The players' feedback is highly encouraging, especially with regards to the FPS genre. It appears that finger-based interaction is highly suitable for genres associated with absolute positioning and control, such as the FPS genre in which the player's hands correspond to the video game character's hands to use items held by that character. This direct mapping coupled with performing exact gestures as one would in real life proved popular with the players. The players gave unanimous praise for the FPS interaction, and

they felt that the FPS game was easy to learn and natural to play. Clearly, future developments in this genre may prove fruitful.

Players also enjoyed the menu interaction method. While extremely simple, players expressed the superiority of the grab gesture for menu item selection versus menu selection with the Kinect. Activating menu items was snappy and precise. Such quick activation is only naturally possible by using the fingers to grab the menu item. Without finger interaction, the players would be forced to use dwell time to activate menu items. However, dwell time was criticized from experience by several players as being frustrating, tiring, and inexact. Therefore, despite its simplicity in this case, finger incorporation can make a big difference in small details as well.

In general and despite a few issues, the players could quickly and dynamically play both games by using hand gestures consisting of finger-gestures and hand motion. Both the driving game and the FPS game require real-time response and fine motions to accomplish their goals. The simple tests show that the hands can be used as the controller in high performance games when the fingers are captured and used for interaction. The positive feedback validates future work in this area. The FPS game is mostly a toy game with only one simple goal, and the feedback can be used to expand the interaction in the FPS genre by incorporating more gestures and goals, among other things.

Some genres may be less suitable for finger-based hand interaction as observed from the driving game test. In cases where constant manipulation and absolute positioning are required, performance and ease of use suffered. However, the results of

the driving test seem to indicate that the majority of the player's difficulties lay with hand motion rather than finger motion. Perhaps finger-based gestures can be used to augment the interaction in such genres rather than be the central mode of interaction. For such genres in which fatigue may occur, finger gestures could be used in conjunction with specialty physical controllers, thus reducing fatigue while retaining detailed hand recognition for interaction. Clearly more work can go into this area as well to adapt whole-arm interaction to this type of game.

This implementation is a base system which can be immediately expanded upon. Such immediate additions include more in-game interactions and gestures, more sophisticated gesture recognition, more in-game control manipulation investigation, and other types of aiming interaction. There are also many areas of future work that can be built off this system as well.

# 10. FUTURE WORK

There are many possible ways to extend this research. Full-body interaction is being explored in many domains, and here only the arms, hands, and fingers are used. There are seemingly infinitely many ways to extend this work, some of which are listed here.

The current gesture recognition method is simple in order to highlight the importance of the fingers for interaction. In future games, more sophisticated gestures could be used. Specifically, hand motion combined with finger motion could be used to define a very large gesture space. These types of gestures will introduce new problems or exacerbate existing problems as well, the biggest of which is probably gesture segmentation. While current segmentation is simple, gestures involving hand motion will require more complicated, dynamic segmentation.

Current sketch recognition techniques could be used to both segment and recognize gestures. The combination of hand path processing with the existing finger-gesture recognition should allow me to enhance the games with many more possible gestures and methods of interaction. This would introduce many problems as mentioned previously, namely noise. This may necessitate more advanced and precise hardware which is currently available. In addition, this would introduce 3D sketch recognition, which itself is an active research area.

To further immersion in the video game, this work could be combined with virtual or augmented reality vision hardware. Stereo 3D headsets are commercially

available and allow the user to view 3D content in 3D, thus giving the player the impression that he is inside the game. Simply projecting the 3D world into a headset would allow the player to interact within the game, but augmented reality opens up even more possibilities for immersion and interaction. For example, the player could see her own arms as she plays. When a weapon is accessed, it could be shown in the player's actual hand. Other players could be real humans as well and the 3D world could be synthetic or take place within the real world. Of course, such interaction would require developments in vision-based tracking as well.

In addition to 3D vision hardware, other virtual reality techniques could be used to increase realism, immersive interaction, and recognition accuracy. Such additions could include various hand models to more accurately understand a player's actions, such as the current grasp state. Prachyabrued's hand model [38] is used to determine when an object is released based on the finger states. Incorporating this and other similar models could increase recognition accuracy and account for the fact that virtual items are being held in an empty hand. Since the current game only incorporates a motion model, a hybrid motion/static model might be beneficial. With regards to interaction, other virtual reality techniques could be used such as contact points on each held object as first proposed by Bergamasco [39]. This could allow more realistic grasping and manipulating of objects which could lead to increased immersion. It could also be used for aiding recognition that requires objects to be held in certain manners, such as a trigger pulling gesture on a gun.

This method of interaction could be combined with augmented reality to enhance the experience. While this approach mimics the player's real hands to interact within a virtual world, an augmented reality interface using a head mounted display would integrate the virtual world with the real world. Instead of mimicking the player's hands, the player's hands would actually be the game character's hands. For example, when the player reaches down and grabs a pistol, a virtual pistol will be drawn over the player's actual hand as if he is actually holding a pistol. In addition to increasing immersion and realism, it could make the game easier in some aspects, such as aiming. Such an approach would obviously be very complex and would require many new solutions including synchronization and possibly advanced computer vision algorithms. Such a system will be built eventually, and this method of interaction and gesture recognition could be a foundation for interaction in that system.

This work might have applications in further domains. Glove-based interaction has been used for domains such as 3D sculpting, in which the user's hands manipulate some 3D object [13]. There may also be other applications within art, education, training, rehabilitation, and medical domains. This method may be combined with other technologies as well to form a more complete method of interaction.

There is one problem with the current approach. While playing the FPS game, players noted that there is no current way to move about within the virtual environment. This was also an early observation of mine, and it does pose an interesting problem. Because the hands are occupied in controlling the character's hands, they cannot manipulate some controls for moving the character within the environment. There are

many possible ways to implement character motion which could be added to the existing system. One way may be to incorporate foot controls for walking gestures.

Finally, the current implementation only provides visual feedback. Immersion could be increased by incorporating vibrotactile (haptic) feedback on the hands when interaction occurs. For example, when a pistol is shot, the player could feel a jolt corresponding to the kick of the gun. Alternatively, if the player punches an object or hits something with the sword, directional haptic feedback may enhance that interaction. The CyberTouch gloves from CyberGlove systems could be used. These gloves are the same data-collecting gloves as the CyberGlove but with vibrotactile actuators on the tip of each finger and the palm. Each actuator can be individually controlled with 5 levels of vibration. Indeed, early tests using these gloves can be seen in Figure 2. Haptic feedback using these gloves or some other method of vibrotactile feedback could be used.

In summary, this work can be viewed as a platform on which to add many new technologies and methods of interaction. The current implementation can be a foundation for finger-interaction, and these many possibilities for expansion are very exciting and filled with intriguing possibilities for exploration.

# 11. CONCLUSION

While underexplored, the area of arm, hand, and finger-gesture interaction is full of opportunities and has great potential to influence the way we interact with video games. A gaming system has been implemented that allows whole-arm interaction to play an FPS game, a driving game, and to navigate a menu system. Gesture recognition and segmentation methods have also been implemented to recognize hand gestures using finger motion. All the games and menu system were implemented from the ground up with whole-arm interaction in mind. This game system integrates all these elements and functions properly, allowing games to be played using whole-arm interaction.

Arm, hand, and finger motion as well as finger-gestures can be used in several game genres with varying degrees of success. Of these genres, finger-gestures are highly suitable for first-person style games in which the player uses objects held by the controlled character. In addition, by adding finger interaction to hand motion, the hands can now be used as controllers in high performance, precision games. The test setup has garnered praise from all participants, who were interested and enthusiastic about this interaction style.

Enthusiasm should grow further as more complex yet natural interaction styles and recognition methods are implemented. Whole-arm interaction could be used for more than just gaming. This work should draw attention to this method of interaction, as there is much to be done and potentially much to be gained by researching this area of interaction.

# REFERENCES


1. D.J. Sturman and D. Zeltzer, "A Survey of Glove-based Input," *IEEE Computer Graphics and Applications,* vol. 14, no. 1, pp. 30-39, 1994.
2. R.A. Bolt, "“Put-That-There”: Voice and Gesture at the Graphics Interface," *SIGGRAPH ’80: Proceedings of the 7th Annual Conference on Computer Graphics and Interactive Technique,* vol. 14, no. 3, pp. 262-270, 1980.
3. J. Miller and T. Hammond, "Wiiolin: a Virtual Instrument Using the Wii Remote," *Proceedings of the 2010 Conference on New Interfaces for Musical Expression (NIME 2010),* pp. 497-500, 2010.
4. Nintendo Wii Controllers, http://www.nintendo.com/wii/console/controllers, Accessed on 11/1/2011.
5. PlayStation Move Motion Controller, http://us.playstation.com/ps3/playstation-move/index.htm, Accessed on 11/1/2011.
6. L. Kratz, M. Smith, and F.J. Lee, "Wiizards: 3D Gesture Recognition for Game Play Input," *Future Play '07: Proceedings of the 2007 Conference on Future Play*, pp. 209-212, 2007.
7. J. Payne, P. Keir, J. Elgoyhen, M. McLundie, M. Naef, et al., "Gameplay Issues in the Design of Spatial 3d Gestures for Video Games," *CHI ’06: CHI ’06 Extended Abstracts on Human Factors in Computing Systems*, pp. 1217-1222, 2006.
8. M. de La Gorce, D.J. Fleet, and N. Paragios, "Model-Based 3D Hand Pose Estimation From Monocular Video," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 33, no. 9, p. 33, 2011.
9. Z. Ren, J. Yuan, and Z, Zhang, "Robust Hand Gesture Recognition Based on Finger-Earth Mover’s Distance with a Commodity Depth Camera," *Proc. Of ACM Intl. Conf. on Multimedia (ACM MM 11),* 2011, *in-press.*
10. J.P. Wachs, M. Kölsch, H. Stern, and Y. Edan, "Vision-Based Hand-Gesture Applications," *Communications of the ACM,* vol. 54, no. 2, pp. 60-71, 2011.
11. R.Y. Wang and J. Popovi, "Real-Time Hand-Tracking with a Color Glove," *ACM Transactions on Graphics,* vol. 28, no. 3, pp. 1-8, 2009.
12. Xbox Kinect, http://www.xbox.com/en-us/kinect, Accessed on 11/1/2011.
13. S. Schkolne, "Drawing with the Hand in Free Space," *Leonardo,* vol. 35, no. 4, pp. 371-375, 2002.
14. B. Paulson and T. Hammond, "Office Activity Recognition using Hand Posture Cues," *BCS-HCI '08: Proceedings of the 22nd British CHI Group Annual Conference on HCI 2008,* pp. 75-78, 2008.

15. T. Schlömer, B. Poppinga, N. Henze, and S. Boll, "Gesture Recognition with a Wii Controller," *Proceedings of the 2nd International Conference on Tangible and Embedded Interaction (TEI -08),* pp. 11-14, 2008.

16. A. Shirai, E. Geslin, and S. Richir, "WiiMedia: Motion Analysis Methods and Applications Using a Consumer Video Game Controller," *Proceedings of the 2007 ACM SIGGRAPH Symposium on Video Games (Sandbox '07),* pp. 133-140, 2007.

17. W.T. Freeman, D. Anderson, P. Beardsley, C. Dodge, H. Kage, et al., "Computer Vision for Interactive Computer Graphics," *IEEE Computer Graphics and Applications*, vol. 18, no. 3, pp. 42-53, 1998.

18. EyeToy USB Camera. http://us.playstation.com/ps2/accessories/eyetoy-usb-camera-ps2.html, Accessed on 11/1/2011.

19. M. Störring, T.B. Moeslund, Y. Liu, E. Granum, "Computer Vision-Based Gesture Recognition for an Augmented Reality Interface," *The 4th IASTED International Conference on Visualization, Imaging, and Image Processing*, pp. 766-771, 2004.

20. H. Hamer, K. Schindler, E. Koller-Meier, and L. Van Gool, "Tracking a Hand Manipulating an Object," *2009 IEEE 12th International Conference on Computer Vision*, pp. 1475-1482, 2009.

21. Y. Sugiura, et al., "An Operating Method for a Bipedal Walking Robot for Entertainment," *ACM SIGGRAPH Asia Emerging Technologies*, p. 103, 2009.

22. T. Komura and W.C. Lam, "Real-Time Locomotion Control by Sensing Gloves" *Computer Animation and Virtual Worlds*, vol. 17, no. 5, pp. 513-525, 2006.

23. B. Paulson, D. Cummings, and T. Hammond, "Object Interaction Detection Using Hand Posture Cues in an Office Setting," *International Journal of Human-Computer Studies*, vol. 69, no. 1-2, pp. 19-29, 2010.

24. The Peregrine, http://theperegrine.com, Accessed on 11/1/2011.

25. P.A. Harling and A.D.N. Edwards, "Hand tension as a Gesture Segmentation Cue" *Proc. Gesture Workshop*, pp. 210-216, 1996.

26. J. Aleotti and S. Caselli, "Grasp Recognition in Virtual Reality for Robot Pregrasp Planning by Demonstration," *IEEE International Conference on Robotics and Automation.* pp. 2801-2806, 2006.

27. J. LaViola, "Whole-Hand and Speech Input in Virtual Environments, " *Master's Thesis, Brown University, Department of Computer Science*, 1999.

28. S. Iba, J. Weghe, C. Paredis, and P. Khosla, "An Architecture for Gesture Based Control of Mobile Robots," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'99)*, pp. 851-857, 1999.

29. K. Bernardin, K. Ogawara, K. Ikeuchi, and R. Dillmann, "A Sensor Fusion Approach for Recognizing Continuous Human Grasping Sequences Using Hidden Markov Models," *IEEE Transactions on Robitics*, vol. 21, no. 1, pp. 47-57, 2005.

30. K. Kahol, P. Tripathi, and S. Panchanathan, "Automated Gesture Segmentation from Dance Sequences" *Sixth IEEE International Conference on Automatic Face and Gesture Recognition*. pp. 883-888, 2004.

31. CyberGlove II, http://cyberglovesystems.com/products/cyberglove-ii/overview, Accessed on 11/1/2011.

32. Ascension Technology Corporation - Flock of Birds, http://www.ascension-tech.com/realtime/RTflockofBIRDS.php, Accessed on 11/1/2011.

33. VirtualHand SDK, http://cyberglovesystems.com/products/virtual-hand-sdk/overview, Accessed on 11/1/2011.

34. L.R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of the IEEE*, pp. 257-286, 1989.

35. Accord.NET, http://accord-net.origo.ethz.ch, Accessed on 11/1/2011.

36. JigLibX Physics Library, http://jiglibx.codeplex.com, Accessed on 11/1/2011.

37. T. Sko and H. Gardner, "The Wiimote with Multiple Sensor Bars: Creating an Affordable, Virtual Reality Controller" *Proceedings of the 10th International Conference NZ Chapter of the ACM's Special Interest Group on Human-Computer Interaction (CHINZ '09)*, pp. 41-44, 2009.

38. M. Prachyabrued and C.W. Borst, "Dropping the Ball: Releasing a Virtual Grasp." *IEEE Symposium on 3D User Interfaces (3DUI)*, pp. 59-66, 2011.

39. M. Bergamasco, P. Degl'Innocenti, and D. Bucciarelli, "A Realistic Approach for Grasping and Moving Virtual Objects," *Intelligent Robots and Systems '94. 'Advanced Robotic Systems and the Real World', IROS '94. Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems*, pp. 717-724, 1994.

# VITA

Drew Anthony Logsdon received his Bachelor of Science degree in computer science in May 2009 and his Master of Science degree in computer science in December 2011 from Texas A&M University in College Station, Texas.

Mr. Logsdon may be reached at dalogsdon@gmail.com or at Texas A&M University, TAMU 3112, College Station, TX 77843.