ABSTRACT

This thesis presents the design and implementation of software for three virtual reality (VR) motor rehabilitation systems for patients post stroke. The addition of virtual reality software to advanced rehabilitation devices (described in this thesis as a VR system) has opened new doors for in home and clinical rehabilitation. The first project presented is the design of a Multiple User Virtual Environment for Rehabilitation (MUVER). This low cost system is designed for in home rehabilitation of the hand and arm. The MUVER system uses a P5 Glove as an input device and was designed using the Panda 3D game engine. By combining virtual interactions with rehabilitation exercises and computer game design theory the MUVER is able to augment conventional rehabilitation and assist with patient’s motivation. The second project discussed is the Active Hand Rehabilitation Interface (ARHI). The AHRI uses custom hardware designed to provide two degrees of freedom to the user’s hand and wrist. Because the ARHI is actuated it is able to provide force feedback during rehabilitation. A virtual environment system was created in Panda 3D for the ARHI to display practitioner created mazes that mimic rehabilitation exercises. The last project discussed is the VRehab Bicycle Rehabilitation System. Custom hardware was created for this VR system as well as interface software to communication between the hardware sensors and the VR software. This interface software was capable of data logging as well as transmission of data using the user datagram protocol (UDP). Overall it is shown throughout the thesis that the use of VR software with advanced rehabilitation devices can augment and improve patient’s motor rehabilitation post stroke.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Figures</td>
<td>iv</td>
</tr>
<tr>
<td>Thesis Advisors</td>
<td>viii</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>viii</td>
</tr>
<tr>
<td>Dedication</td>
<td>viii</td>
</tr>
<tr>
<td>Chapter 1: Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Virtual Reality Systems</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Definitions and Nomenclature</td>
<td>3</td>
</tr>
<tr>
<td>Chapter 2: Background</td>
<td>6</td>
</tr>
<tr>
<td>2.1 Stroke Information</td>
<td>6</td>
</tr>
<tr>
<td>2.2 Rehabilitation Studies using VR Systems</td>
<td>7</td>
</tr>
<tr>
<td>2.2.1 Rehabilitation Devices</td>
<td>7</td>
</tr>
<tr>
<td>2.2.2 Rehabilitation Software</td>
<td>19</td>
</tr>
<tr>
<td>Chapter 3: Multiple User Virtual Environments for Rehabilitation</td>
<td>28</td>
</tr>
<tr>
<td>3.1 MUVER System Overview</td>
<td>28</td>
</tr>
<tr>
<td>3.2 Initial Prototype Software</td>
<td>33</td>
</tr>
<tr>
<td>3.2.1 Initial Prototype Software Mechanic Design</td>
<td>33</td>
</tr>
<tr>
<td>3.2.2 Initial Prototype Software Virtual Environment Design</td>
<td>35</td>
</tr>
<tr>
<td>3.3 Final Prototype Software</td>
<td>36</td>
</tr>
<tr>
<td>3.3.1 Final Prototype Software Mechanic Design</td>
<td>36</td>
</tr>
<tr>
<td>3.3.2 Final Prototype Software Virtual Environment Design</td>
<td>38</td>
</tr>
<tr>
<td>3.4 Testing of the MUVER</td>
<td>42</td>
</tr>
<tr>
<td>3.4.1 Final Software Testing</td>
<td>43</td>
</tr>
<tr>
<td>3.5 P5 Glove Testing</td>
<td>44</td>
</tr>
<tr>
<td>3.6 MUVER Future Work</td>
<td>49</td>
</tr>
<tr>
<td>3.6.1 Improvements to The MUVER System</td>
<td>49</td>
</tr>
<tr>
<td>3.6.2 Hardware Input Device</td>
<td>51</td>
</tr>
<tr>
<td>3.6.3 Rapid Prototyping Virtual Objects</td>
<td>53</td>
</tr>
<tr>
<td>3.7 MUVER Summary</td>
<td>53</td>
</tr>
<tr>
<td>Chapter 4: Virtual Environments for The Active Hand Rehabilitation Interface</td>
<td>55</td>
</tr>
<tr>
<td>4.1 Active Hand Rehabilitation System Hardware and Controls</td>
<td>55</td>
</tr>
<tr>
<td>4.1.1 Active Hand Rehabilitation System Device Hardware</td>
<td>55</td>
</tr>
<tr>
<td>4.1.2 Device Control</td>
<td>59</td>
</tr>
<tr>
<td>4.2 Virtual Environment Software for the ARHI</td>
<td>62</td>
</tr>
<tr>
<td>4.2.1 UDP Communication in LabVIEW</td>
<td>64</td>
</tr>
<tr>
<td>4.2.2 UDP Communication in Panda 3D</td>
<td>65</td>
</tr>
<tr>
<td>4.2.3 Initial Prototype Software</td>
<td>66</td>
</tr>
<tr>
<td>4.2.4 Final Software Design</td>
<td>67</td>
</tr>
<tr>
<td>Chapter</td>
<td>Title</td>
</tr>
<tr>
<td>---------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>4.3</td>
<td>Testing of ARHI Software</td>
</tr>
<tr>
<td>4.4</td>
<td>ARHI Software Future Work</td>
</tr>
<tr>
<td>4.5</td>
<td>ARHI Summary</td>
</tr>
<tr>
<td>Chapter 5: Interface Software for the VRehab Bicycle System</td>
<td>86</td>
</tr>
<tr>
<td>5.1</td>
<td>VRehab Bicycle System Hardware</td>
</tr>
<tr>
<td>5.1.1</td>
<td>Handlebar System</td>
</tr>
<tr>
<td>5.1.2</td>
<td>Pedal System</td>
</tr>
<tr>
<td>5.1.3</td>
<td>Heart Rate Monitor</td>
</tr>
<tr>
<td>5.1.4</td>
<td>Other Hardware Components and Systems</td>
</tr>
<tr>
<td>5.2</td>
<td>VRehab Software</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Signal Interface</td>
</tr>
<tr>
<td>5.2.2</td>
<td>VRehab Main Interface</td>
</tr>
<tr>
<td>5.2.3</td>
<td>VR Simulation</td>
</tr>
<tr>
<td>5.3</td>
<td>Testing of the VRehab System</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Testing Protocol</td>
</tr>
<tr>
<td>5.3.2</td>
<td>VRehab Testing Data</td>
</tr>
<tr>
<td>5.3.3</td>
<td>Analysis and Discussion</td>
</tr>
<tr>
<td>5.4</td>
<td>VRehab Future Work</td>
</tr>
<tr>
<td>5.5</td>
<td>VRehab System Summary</td>
</tr>
<tr>
<td>Chapter 6: Conclusions</td>
<td>143</td>
</tr>
<tr>
<td>References</td>
<td>146</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

Figure 1: Virtual Reality System Components ................................................................. 2
Figure 2: P5 Glove and Infrared Tower ............................................................................. 8
Figure 3: SensAble Phantom Devices ............................................................................... 10
Figure 4: The Novint Falcon .......................................................................................... 11
Figure 5: Nintendo Wii Remote and Nun’ Chuck ........................................................... 13
Figure 6: CyberGlove II ................................................................................................. 14
Figure 7: 5DT Data Glove 5 Ultra .................................................................................. 15
Figure 8: Rutgers Master II-ND Force Feedback Glove ................................................. 16
Figure 9: Commercial stationary bikes & simulators ..................................................... 17
Figure 10: Panda 3D Advanced Shaders ......................................................................... 20
Figure 11: Blender Animation Interface .......................................................................... 22
Figure 12: Flash CS4 Graphical User Interface .............................................................. 24
Figure 13: MUVER System Schematic ......................................................................... 29
Figure 14: Virtual Interactions for Multiple Users ......................................................... 30
Figure 15: Information Communication Schematic ....................................................... 31
Figure 16: Software Programming Elements .................................................................... 32
Figure 17: Pumpkin Prototype Virtual Scene Movement Schematic ............................. 34
Figure 18: MUVER Initial Prototype Scene ................................................................... 35
Figure 19: Final Prototype Virtual Scene Movement Schematic .................................. 37
Figure 20: MUVER Virtual Scene Rendering During a Session .................................... 38
Figure 21: World Properties Setup Code Segment ....................................................... 39
Figure 22: Collision Traverser Code Segment ............................................................... 40
Figure 23: Model Loading Code Segment ..................................................................... 40
Figure 24: Collision Node Code Segment ....................................................................... 41
Figure 25: Mouse Position Task Code Segment ............................................................ 41
Figure 26: Collision Task Code Segment ...................................................................... 42
Figure 27: Graph of the Time of Virtual Environment Movement Phases ................. 43
Figure 28: Graph of X and Y Direction Movement using the P5 ................................ 45
Figure 29: Graph of Supination and Pronation with Cross Talk .................................. 46
Figure 30: Graph of Finger Flexion and Extension with a Large Hand .................................. 47
Figure 31: Graph of Finger Flexion and Extension with a Small Hand .................................. 48
Figure 32: Complete MUVER System Global Schematic .......................................................... 50
Figure 33: Model of a Prototype Glove Design ........................................................................ 52
Figure 34: Active Hand Rehabilitation Interface Global Schematic ........................................... 56
Figure 35: Active Hand Rehabilitation Interface ....................................................................... 57
Figure 36: Two Degree of Freedom Robotic Hand Rehabilitation Device ................................ 58
Figure 37: The Generic Control Diagram of the System ............................................................... 60
Figure 38: Block diagram of the PI + Feed Forward Controller .................................................. 61
Figure 39: Software Communication Global Schematic ............................................................... 63
Figure 40: Simple UDP Sender Block Diagram ........................................................................ 64
Figure 41: Code Segment for Creating a UDP Socket ................................................................. 65
Figure 42: Code Segment for Reading UDP Data ....................................................................... 66
Figure 43: Initial Prototype Software .......................................................................................... 67
Figure 44: Software Communication Global System Schematic ................................................ 68
Figure 45: Possible Virtual Movement Mechanics ...................................................................... 70
Figure 46: Screen Capture of Final Software Prototype Simple Maze ....................................... 72
Figure 47: Advanced Maze for ARHI Final Prototype Software ................................................. 73
Figure 48: Item Array Code Segment .......................................................................................... 74
Figure 49: World Properties Setup Code Segment ...................................................................... 75
Figure 50: Collision Handler Code Segment .............................................................................. 75
Figure 51: Maze Elevation Map Creation Code Segment ............................................................ 75
Figure 52: Maze Creation and Rendering Code Segment ........................................................... 76
Figure 53: Elevation Evaluation Code Segment ......................................................................... 77
Figure 54: Item Creation Code Segment .................................................................................... 77
Figure 55: Graph of Total Time for ARHI Simple Maze .............................................................. 79
Figure 56: Graph of ARHI Simple Maze Average Force .............................................................. 80
Figure 57: Graph of ARHI Simple Maze Average Torque ........................................................... 81
Figure 58: Maze Progression Schematic ...................................................................................... 82
Figure 59: Venture and Return Mechanic ................................................................................... 83
Figure 60: Other Possible In-Game Objects .............................................................................. 84
Figure 61: Bicycle System Global Schematic .............................................................. 87
Figure 62: Schematic of Handlebar system .............................................................. 89
Figure 63: Rendering and Physical Prototype of the Handlebar System ............... 90
Figure 64: Calibration Hardware .............................................................................. 91
Figure 65: Render of the Left Pedal Assembly ......................................................... 92
Figure 66: Pedal Systems Global Schematic ............................................................. 94
Figure 67: Schematic of SGAU strain gauge signal amplifier .............................. 98
Figure 68: Overall view of power and signal boxes .............................................. 99
Figure 69: Information Communication Diagram .................................................. 100
Figure 70: The Signal Interface Front Panel ........................................................... 102
Figure 71: VR rehab Main Interface Front Panel ..................................................... 104
Figure 72: Loop and Data Logging Controls ............................................................ 104
Figure 73: Handlebar Display and Controls ............................................................. 105
Figure 74: Pedal Force Display and Controls .......................................................... 106
Figure 75: Pedal Angle Display .............................................................................. 106
Figure 76: Velocity and RPM Controls and Display ............................................. 107
Figure 77: Heart Rate and Vibration Controls and Display ................................... 108
Figure 78: Minimum and Maximum Controls and Display .................................. 108
Figure 79: Min/Max Sub VI Block Diagram ........................................................... 109
Figure 80: Heart Rate Sub VI Block Diagram ........................................................ 110
Figure 81: Hall Effect Sensor Sub VI Block Diagram ............................................ 112
Figure 82: Velocity Sub VI Block Diagram ............................................................. 113
Figure 83: Main Handlebar Sub VI Block Diagram .............................................. 114
Figure 84: Handlebar Secondary Calculation Sub VI Block Diagram .................. 115
Figure 85: Main Pedal Load Cell Sub VI Block Diagram .................................... 116
Figure 86: Pedal Load Cell Override Sub VI Block Diagram ............................... 116
Figure 87: Vibrating Element Sub VI Block Diagram .......................................... 117
Figure 88: Data Recording Finite State Diagram .................................................. 118
Figure 89: Default State Block Diagram ............................................................... 119
Figure 90: Recording Data State Block Diagram ................................................ 121
Figure 91: File Save and Reset State Block Diagram .......................................... 122
THESIS ADVISORS

PROFESSOR CONSTANTINOS MAVROIDIS, PhD
Department of Mechanical and Industrial Engineering, Northeastern University

PROFESSOR MAUREEN HOLDEN, PhD, PT
Department of Physical Therapy, Northeastern University

ACKNOWLEDGEMENTS

First I would like to acknowledge and thank my advisors Professor Mavroidis and Professor Holden, without their knowledge, help, and encouragement this document would never have been finished. I would also like to thank and acknowledge the other members (past and present) of the Biomedical Mechatronics Laboratory that I had the privilege of working with: Brian, Azedah, Rich, Maciek, Vish, Ozzy, Yalgin, Ye, and Johannes.

DEDICATION

I would like to dedicate my thesis to my friends and family. My pursuit of a higher education would not have been possible without your support and sacrifices.

THANKS MOM, DAD, SCOTT, SETH, AND ERIN.
CHAPTER 1: INTRODUCTION

The Introduction chapter for this thesis is split into two sections: virtual reality (VR) systems and definitions and nomenclature. The VR systems section contains information about the components of a VR system and how it pertains to the three projects presented in this thesis. The definitions and nomenclature section explains how the thesis is formatted and also defines terms that are used throughout the thesis.

1.1 VIRTUAL REALITY SYSTEMS

Recent technological advancements in the fields of computer processing power, displays, and graphics have blended with input device technologies to create the foundation for virtual reality (VR) systems. VR systems have found uses in many industries including entertainment (computer and video games), defense, aviation, machine operation, medicine, and rehabilitation. This thesis presents projects on the use of VR systems in rehabilitation.

There are four main components in every VR system. The figure below shows a flowchart of the interactions of these components and how they relate to the user and practitioner.
There are three hardware components of a VR system: the input device, display, and computer. The input device (as defined in Section 1.2) is used by the user to interact with the virtual environment. The display (in all projects presented in this thesis a standard computer monitor was used) gives visual feedback to the user by displaying the virtual environment. The computer is connected to both the display and input device. The computer runs the VR software and any other necessary programs that are needed in the VR system.

The focus of this thesis is on the design and use of VR software and virtual environment design. There is also information on custom input devices that were designed for use with the VR software because a VR system requires both software and hardware to function.
1.2 Definitions and Nomenclature

This thesis covers work completed in three projects. To make access to this information easier the thesis is formatted so that all the information, including testing, analysis, and future work, for each project is contained in a separate chapter. Also there is a chapter for introduction, background information, and one for conclusions that are relevant to all three projects.

Throughout the thesis there are terms that require definition. The terms and their definitions as used in this thesis are below.

**Avatar**

An “avatar” is the virtual representation of the user in a virtual environment, scene, or game. If there is no graphical representation of the avatar the term represents the presence of the user in the virtual environment, scene, or game.

**Event**

“Events” is used in this thesis to describe reactions of the virtual environment to the avatar and the mechanics.

**Game**

The term “game” has many different definitions depending on the scope of the subject. In this thesis the term “game” is used to describe virtual environments and virtual scenes where the user has a specific task and is given a score based upon how they perform that task. Note that for this thesis the terms “game”, “video game”, and “computer game” are interchangeable and identical.
**GAME ENGINE**

A game engine is a software program or group of programs that is designed to aid in the development of video or computer games. More specifically this means that the game engine includes classes, libraries, or methods to create and display geometry as well as control physics properties.

**GOAL**

The term “goal” is used to describe the positive endpoint of the mechanic. It is also used as motivation for the user to interact with the virtual environment.

**INPUT DEVICE**

The term “input device” as used in this thesis means a device that provides data to a software program. This includes rehabilitation devices with sensors and video game console controllers.

**INTERFACE**

“Interface” as used in this thesis means the software that the practitioner uses to control the input device and/or the virtual reality, scene, or environment. The term “interface” is also synonymous with “user interface” or “graphical user interface” in this thesis.

**MECHANIC**

The term “mechanic” is used to describe a single component of a virtual scene, environment, or game such as the player avatar’s movement or the scoring. It may also be used as “game mechanic”.

---

4
**PRACTITIONER**

This thesis uses the term “practitioner” to mean the person who is running the rehabilitation session. This can include physical therapists, occupational therapists, doctors, or engineers.

**USER**

The term “user” as used in this thesis means the person that is controlling the input device. In general this means the patient using the rehabilitation device but the term user is used to cover all possible users including healthy people and other practitioners.

**VIRTUAL REALITY, ENVIRONMENT, AND SCENE**

The term “virtual reality” is used in this thesis to denote that a software program that is controlled by a user via an input device and that has a graphical component. “Virtual environment” refers to the graphical component of the virtual reality software, it is also where the user’s avatar interacts with other virtual objects or performs task. A “virtual scene” is a section of the virtual environment and is used to describe a specific mechanic or task that takes place in the larger virtual environment.
CHAPTER 2: BACKGROUND

This chapter is separated into two major sections: information on stroke and rehabilitation studies using VR. The stroke information section contains information on the causes of stroke, stroke statistic in the United States, and information on rehabilitation. The section on rehabilitation studies using VR has a special focus on rehabilitation for stroke using either a glove or hand device or a bicycle system.

2.1 STROKE INFORMATION

Strokes occur when the blood flow in the brain is hampered and brain tissue is damaged. There are two major kinds of strokes: ischemic and hemorrhagic. Ischemic strokes account for about 87% of all strokes and they occur when a blood vessel in the brain is clogged by a clot [1]. Hemorrhagic strokes occur when a blood vessel in the brain ruptures and blood leaks into the brain.

The severity of damage to the brain tissue is dependent on many factors including age, area of the brain affected, and other medical conditions. However, most people who suffer a stroke have their motor skills and abilities affected.

Stroke is the leading cause of disability among adults in the United States with more than 795,000 people annually suffering a new or recurrent stroke [2]. Stroke accounted for 6% of all deaths in the United States in 2005 [2]. As the number of stroke fatalities continues to decrease the numbers for surviving stroke victims continues to rise. This grants the need for more efficient and advanced ways to rehabilitate those who have suffered a stroke.
Six months post stroke, 55-75% of survivors still have limitation of upper extremity (UE) function [3]. Once standard rehabilitation has been completed many patients remain highly motivated to make further gains, and studies have shown that patients with chronic stroke do have the potential to improve [4]. This motivation effect is one of many aspects that make the use of VR systems appropriate for rehabilitation.

2.2 Rehabilitation Studies using VR Systems

This section presents information on other VR systems and their use in rehabilitation. There are many systems in use in academia today so this section focuses on the use of systems for stroke rehabilitation using either the hand or a bicycle system. Also most studies involve rehabilitation of patients with stroke.

2.2.1 Rehabilitation Devices

This section contains information on the use of rehabilitation devices with VR systems. Many of the devices presented in this section are computer input devices that have been taken off the shelf for use with rehabilitation. The most common reasons for this adaptation are because the devices are low cost, easy to setup, and many include a software development kit to aid with virtual environment design.

2.2.1.1 P5 Glove

The P5 Glove was released from Essential Reality in 2002 aimed at the home personal computer (PC) video game market [5]. The hand device itself is attached to an Infrared tower via a PS/2 cable, that tower is attached to the PC via a USB cable. The P5 comes with a CD that has device drivers that allow the device to work on any computer running
a Microsoft operating system and also on any Mac using OS version 9 or below [5]. The P5 can emulate a mouse for any application but can also provide 3D interaction with applications compatible or designed for the P5. A picture of the P5 is below.

![P5 glove](image.png)

**Figure 2: P5 Glove and Infrared Tower**

The P5 glove was not designed to be a rehabilitation device. However, its low cost and easy to use software has made it popular for rehabilitation uses. There is no commercially available rehabilitation software for the P5, however many researchers have created their own software or used compatible game software.

One such use of the glove was conducted in 2006 at Rutgers University [6]. The researchers used a P5 Glove and an Xbox console to display virtual environments that were created using Java 3D. The hardware and software for the VR system was chosen
based upon cost and what a patient could afford in their home. The researchers found that the low cost setup including the P5 did have much lower functionality than a traditional hand rehabilitation VR system [6].

The P5 Glove was chosen to use for hardware in the MUVER project (Chapter 3). The reasons it was chosen were presented above. Its low cost and ease of setup were the two factors. However, as testing was completed (Section 3.5) on the device it became evident that a superior device must be obtained for future work with that project.

2.2.1.2 PHANTOM DEVICES

Sensible Technologies has a product line of haptic input devices designed to gather input and give feedback to the user’s fingers, hand, and arm. The devices most suited to be used in rehabilitation VR systems are the six degree of freedom PHANTOMs. Users grasp the phantom stylus (device the shape of a pencil or pen) in either hand and have control of x-direction, y-direction, z-direction, roll, pitch, and yaw. A figure of two SensAble PHANTOM devices can be seen below [7].
The PHANTOM six degree of freedom device is interfaced with the user’s PC via a parallel, Firewire, or USB port. The device also comes bundled with several software demos and a software development kit specific to the inputs and limitations of the device. SensAble devices have been used in rehabilitation studies but their main use is as input devices for computers [7].

The PHANTOM was used as a haptic handwriting aid for training and rehabilitation at Deakin University in Australia [8]. In this study the researchers used the Omni PHANTOM device as a handwriting aid to assist in rehabilitation post stroke. The researchers used the SensAble software development kit along with their own code to create the virtual environments. In house testing was performed on the VR system and the use of the Omni showed great promise as a rehabilitation tool [8].
The main advantage the PHANTOM devices have over other devices presented is force feedback. The main disadvantage of is that the stylus must be grasped at all times so the user can not exercise their fingers using the device.

2.2.1.3 NOVINT FALCON

The Falcon device from Novint is a computer input device that is designed for playing video games. The Falcon has a 4” x 4” x 4” workspace and a two pounds force feedback capability. It is a 3 degree of freedom device in terms of movement, however by using different grips more degrees of freedom can be added by buttons or dials [9]. A figure of the device can be seen below.

![Figure 4: The Novint Falcon](image)

The Falcon interfaces with a PC using a USB 2.0 port. The Falcon has several games already available for it as well as driver software to play other games. The Falcon software development kit is available for download [9].
The device has been chosen for use in rehabilitation research in the past because of its simple programming interface and relatively low cost. One such study was conducted in 2008 in the UK [10]. The researchers picked the Novint Falcon device because of low cost, three degrees of freedom, force feedback, and other factors. The study was setup to examine the rehabilitation of the upper extremity of stroke patients. The researchers used some of the built in or off the shelf virtual environments in conjunction with the Falcon. Subjects who had experienced a stroke at least six months prior were split into two random groups four. One of these groups performed tasks in virtual environments using the Falcon and the other group did virtual reality “relaxation” exercises that did not include using the software with the Falcon. The researchers found that the group that used the Falcon in conjunction with the virtual environment showed improvement after only six of the twelve sessions [10].

The Falcon device has many advantages but because of its design only having three degree of freedom and not having good finger sensors or interaction it is not a suitable device for a project like the MUVER. While it does have force feedback it lacks the fast response and high level of force that the ARHI is capable of producing.

2.2.1.4 Nintendo Wii

The Nintendo Wii was released in the fall of 2006 as a personal video game console [11]. The Wii controller has two components: a remote and “nun’ chuck”. The Wii remote is normally held in the player’s dominant hand. If the player is using both the nun’ chuck and the remote, the remote is usually held in the right hand. The nun’ chuck plugs into
the remote and is usually held in the left hand and not a requirement for all Wii games. A figure of the Wii remote and nun’ chuck can be seen below.

![Wii Remote and Nun' Chuck](image)

**Figure 5: Nintendo Wii Remote and Nun' Chuck**

The Wii console connects directly to a television (either via component video and audio or some kind of converter) and the Wii controllers communicate with the console using Bluetooth wireless technology. There is a large library of Wii games, however at this time there is no licensed software development kit available to the public.

The Wii was used in a case report for rehabilitation of an adolescent with Cerebral Palsy [12]. Deutsch et al chose to use the Wii because of the age of the subject as well as the low cost of the system. They also found that using the off the shelf software was sufficient in aiding rehabilitation.
The largest drawback of the Wii system is lack of readily available software development kit. However, since the remote uses Bluetooth for communication it is possible to use it with a Bluetooth capable computer but this is far from ideal. The other drawback of the Wii is that the user is required to grip the remote therefore making exercise of the fingers difficult.

2.2.1.5 CyberGlove II

The CyberGlove II is a wireless device and has a capacity of twenty-two high-accuracy joint-angle measurements [13]. The glove uses a proprietary resistive bend-sensing technology to capture real-time digital joint-angle data. The CyberGlove is the device closest to an actual glove that is discussed in this paper. A figure of the CyberGlove II can be seen below.

![Figure 6: CyberGlove II](image)

The CyberGlove II connects to a computer by a wireless USB receiver. The device has batteries in a forearm-mounted compartment. Software does come bundled with the glove which is for evaluation and not for virtual reality. Also there is no publicly available software development kit.
The CyberGlove boasts very good accuracy and data resolution but the cost of the system and lack of sensor data for absolute hand position are the main drawbacks of the device. Also lack of software and expensive software support are an issue in its use in a VR system.

2.2.1.6 5DT Data Glove 5 Ultra

The 5DT Data Glove 5 Ultra is designed for use in motion capture and animation [14]. The device measures finger flexure and interfaces with a PC via a USB cable. The device features automatic calibration and has an on-board processor. A figure of the 5DT Data Glove 5 Ultra can be seen below.

![Figure 7: 5DT Data Glove 5 Ultra](image-url)
The 5DT Data Glove 5 Ultra has Bluetooth technology to make it wireless. It also has a cross-platform software development kit. The bundled software that comes with the device has no rehabilitation applications.

The 5DT Data Glove 5 Ultra lacks sensors for absolute hand position. The cost of the device is moderate but still too high for in-home use. The wireless technology is an advantage for use in a large workspace and for compatibility.

2.2.1.7 Rutgers Master Glove

The Rutgers Master II-ND Force Feedback Glove was developed in 2002 at Rutgers University [15]. The Rutgers device uses custom pneumatic actuators and sensors placed on the palm to avoid wires at the fingertips. The direct-drive configuration of the actuators provides force to the tips of the fingers that connected to the device.

![Rutgers Master II-ND Force Feedback Glove](image)

**Figure 8: Rutgers Master II-ND Force Feedback Glove**

The Rutgers Master II-ND device is a research only device and because of this there is no publically available software development kit. The Rutgers Master II-ND device has
several advantages including force feedback and possibility of finger exercises. However, the pneumatic actuators can be cumbersome and decrease the range of movement a user can use. Also the lack of sensors for absolute hand position is a drawback for the device in the uses described in this thesis.

### 2.2.1.8 Exercise and Rehabilitation Stationary Bicycles

There are currently a wide variety of stationary exercise bikes commercially or nearly available. The following section describes examples of comparable technology implementation from least to greatest functionality and complexity.

**Figure 9: Commercial Stationary Bikes & Simulators**

- A. Precor 836i Upright Exercise Cycle
- B. HONDA Riding Trainer System
- C. Dogfight V2 Simulator
- D. Expresso Stationary Bike
- E. Trixter XDream System
Most gyms now require the most straightforward stationary bikes to have a display for exercise conditions like crank RPMs, power generated, calories, heart rate, and difficulty setting. To add interest for the user’s routine some models come including a built-in monitor for music or television program entertainment.

Similar systems have been designed as training tools rather than exercise evaluators. To train user’s reactions and attention for riding a motorcycle or scooter in everyday city traffic as well as emergency situations. Some of these devices are USB-equipped and readily mountable to a desk for interaction with the simulation. They are both input device and do not track the user’s physiological state, extremity forces exerted, and lack haptic feedback.

A comparable integrated exercise bike can monitor parameters for tracking a user’s exercise progress with greater range of motion and interaction than a traditional stationary bike. The user can experience a more immersive workout by utilizing the upper extremities to navigate laterally, initiate turns, and shift gears to increase resistance around the crank.

Some systems have the capacity to connect with each other and allow several riders to exercise simultaneously in the same environment. This has also been taken one step further by making the competition the focus of the interaction using a pedal-powered flight simulator for two-player aerial combat. The Handlebar System stabilizes the user whilst giving them control over the games functions like flight trajectory and combat systems.
The most complex commercial devices with greatest range of motion and instrumentation simulate a mountain biking experience close enough for professionals to train on year round like the Trixter XDream. This device aims at combining upper body and core strength training with increased range of motion for the handlebars and ‘front fork’ which allow the rider to lean and twist similarly to navigating a mountain bike through technically challenging terrain. Data logging is available for velocity, power generated and several others, but the system does not include force measurement for the handles and pedals.

Although the majority of these devices have adequate functionality and data capture capabilities to allow a healthy user to exercise, only a narrow range have been implemented in physical therapy retraining. Physical therapists benefit from greater knowledge and tracking of the user’s capabilities whilst allowing the user to have a greater level of device instrumentation and virtual immersion. Many of the more complex systems are expensive for a clinic, some in the range of $7K-$8K and space can be very limited to procure several systems.

These reasons necessitate a low cost state of the art system with diverse measurement functionality, immersion, and adaptability to any current stationary bike in a practitioner’s office.

**2.2.2 Rehabilitation Software**

The following section presents rehabilitation software platforms that could be used to create virtual environments for VR systems. Besides the software packages listed many
researchers chose to create their own custom software or adapt off the shelf technology to their needs.

2.2.2.1 Panda 3D

The Panda 3D game engine (Panda) was created by Disney and is used by Carnegie Mellon University [16]. It is open source and freely available, also any software created with it can be sold without giving royalties or credit to the creators of Panda 3D. The engine uses Python as the scripting language. Panda3D also has input methods that all for the direct input of Head Mounted Displays (HMD) and VR trackers.

Figure 10: Panda 3D Advanced Shaders
The Panda 3D game engine was used in both the MUVER and ARHI projects (Chapter 3 and 4 respectively). The greatest advantage of Panda 3D is prototyping speed for a virtual environment. The greatest drawback is the performance of the engine. Because the VR systems in general run multiple software programs during a rehabilitation session it is necessary to have an engine that has optimum performance.

The main systems within Panda that were used are the event system, collision detection, loader, timer, and user interface text. The event system within Panda is used to control mechanics and events within the virtual scene. The collision detection system triggers events when models collide with each other or with other collision geometry (like the maze walls in the ARHI software). The Panda loader uses .egg files to load in 3D models and other assets. The timer in Panda is based off the Windows system timer and it is used to time events and mechanics in the virtual scenes. The user interface text system is used to display the score and instructions for the user without issues of window size or resolution.

The Panda 3D engine is a fully featured free game engine that is capable of creating virtual environments and scenes quickly and integrating them with VR systems.

2.2.2.2 Blender

Blender is an open source software package that focuses on digital modeling and animation [17]. It has an integrated game engine (BlenderGE) that uses Python as a scripting language with some visual control. Because Blender is used as a digital animation package it has many features to use armatures (such as a hand).
The BlenderGE was used in the design of a prototype virtual scene for the MUVER project. After the modification of the original virtual scene the BlenderGE software was found to be counter intuitive. Also the scripting and linking to armatures in the animation was a difficult task that was prone to failure, because of this BlenderGE was not used in later versions of the software.

2.2.2.3 XNA

XNA is a product of Microsoft and it was originally release in March of 2006 [18]. The latest stable version of the software (XNA 3.0) was released in October of 2008. The
software is free to download although depending on service and support it may require a yearly subscription.

XNA uses C# code and a shared library to make the creation of games and virtual environments less complicated and more intuitive. Code that is written in XNA can easily be used on both a Windows computer and the Microsoft Xbox 360.

The design of a VR system for vision based games for upper-limb stroke rehabilitation used XNA to prototype the virtual environments [19]. The software was written to run on a Windows computer instead of the Xbox 360 because of the input device (a digital camera) that was being used to control the user’s avatar in the virtual environment.

The biggest advantages of XNA are that the community is very knowledgeable and it is designed specifically for making games. However, because XNA is not a game engine it requires more code and hours of work to create a virtual environment.

2.2.2.4 Flash and ActionScript 3.0

Adobe Flash is mostly used for websites and internet applications. The biggest advantage of using Flash is that many Flash applications can be accessed from any web browser and do not need to be installed. Like XNA, Flash does not have a bundled game engine and would require more programming than other software solutions for virtual environments.

The release of ActionScript 3.0 has opened new doors for the flash platform. ActionScript 3.0 introduced object oriented programming attributes to the language and also makes the creation of more advanced programs (like virtual environments) possible.
Most Flash programs are displayed in 2D although some limited 3D is possible. A screen capture of the Adobe Flash CS4 Graphical User Interface can be seen below.

Flash has been used in rehabilitation before. In 2004 Flash was used (albeit not ActionScript 3.0) to create a software used in the rehabilitation of patients with Aphasia [21]. Aphasia affects a person’s ability for speech and communication. While it is not motor rehabilitation Aphasia is common in patients with stroke. Flash was an obvious pick for this software because it is a simple and fast way to display 2D graphics such as text and sounds such as words.
The virtual environment that was created to be used included a picture of a room that has no colors. The user is then asked to find and click on items based on the text displayed or sound heard. If the user can find the item and click on it the item gets colored in to show feedback to the user that they have completed the task. The researchers found that the Flash software was easy to use for development and also created a virtual environment that was useful in rehabilitation [21].

Motor rehabilitation usually requires 3D virtual environments because they most often mimic real world movements. However, as is the case in Chapter 4 of this thesis 2D virtual environments can be used with motor rehabilitation.

2.2.2.5 GlovePIE

GlovePIE stands for Glove Programmable Input Emulator. It doesn't have to be used with VR Gloves, but it was originally started as a system for emulating Joystick and Keyboard Input using the Essential Reality P5 Glove [22]. Now it supports emulating a variety of input devices including Polhemus, Intersense, Ascension, WorldViz, 5DT, and eMagin products. It can also control MIDI or OSC output.

As stated above the GlovePIE software was created for use with the P5. The way it works is to use software macros to emulate movements made with the P5 glove and bind them to the keyboard or a joystick. By using this method of emulation the program needs only to use either keyboard or joystick control and GlovePIE then extends the program to use the P5. The main issue with this method and hardware emulation in general is that only certain movements that are emulated can be used. This puts a limit on what movements are available to the practitioner and user.
2.2.2.6 National Instruments LabVIEW

LabVIEW (short for Laboratory Virtual Instrumentation Engineering Workbench) is a sensor data acquisition and device control prototyping software.[23] The original version was released in 1986 for the Apple Macintosh. The current version, LabVIEW 8.6.1, was released in February of 2009.

LabVIEW uses a visual programming language to create virtual instruments (VIs) that can be used to control devices and acquire data from sensors. This programming style is built upon creating block diagrams for the information flow of the VI. All the LabVIEW VIs discussed in this thesis use a while loop to iterate a block diagrams functionality at a frequency that is determined by the hardware and sensor speed.

The ARHI and VRehab projects both used LabVIEW as a middleware between the sensors and hardware and the VR software. All data acquisition was also done in LabVIEW for these projects and not done in the VR software.

2.2.2.7 Virtual Reality Peripheral Network

The Virtual-Reality Peripheral Network (VRPN) is a set of classes within a library as well as a set of servers that are designed to implement an interface between application programs and input devices (tracker, etc.) used in a VR system.[24] The system is setup to have a computer at each VR station that controls the input devices and VRPN provides the connections between the VR software and the devices. This is done by accessing the class of the various input devices in VRPN using it to create the connection. The application remains unaware of the network topology.
VRPN also provides an abstraction layer that makes all devices of the same base class look the same to the application software. This means that all input devices produce the same types of reports. At the same time, it is possible for an application that requires access to specialized features of a certain input device to derive a class that communicates with this type of device. The current system types are Analog, Button, Dial, ForceDevice, Sound, Text, and Tracker. Each of these types abstracts a set of semantics for a certain type of device. There are one or more servers for each type of device, and a client-side class to read values from the device and control its operation.[24]
The Multiple User Virtual Environments for Rehabilitation (MUVER) system project was completed from January 2007 to January 2008 [25]. The hardware used for this project was the P5 glove which is explained in Section 2.2.1.1. The following sections discuss the various designs used for the project as well as all the testing that was performed.

Professor Maureen K. Holden PhD, PT was a co-adviser on this project. Her work in virtual environments was used in the creation of the MUVER software [26-28].

3.1 MUVER System Overview

The purpose of this project was to create virtual environments designed for specific rehabilitation exercises and for multiple users to interact together. The MUVER system designed for this project was composed of many different elements that are outlined in the figure below.
The system was designed to be modular, meaning that the amount of users and size of the virtual environment can change without significant redesign. To facilitate this each user installs the MUVER software on their personal computer and the communications all take place over the internet using a peer-to-peer model, this is different than many virtual environments which have their own server dedicated to running the environment. The user then uses the chosen input device to interact with the other users and the virtual environment. It is then possible to log the communications and data from the internet to the practitioner’s computer for later evaluation.

The design of a unique virtual environment such as the MUVER created in this project had several stages. The first stage was to determine what rehabilitation exercises or
movements would be appropriate and feasible to emulate in a virtual environment. Once the rehabilitation exercises had been chosen the next step in the design was to choose the type of multiple user interaction. The four common types of multiple user virtual interaction are illustrated in the figure below.

**Figure 14: Virtual Interactions for Multiple Users**

There are four common kinds of multiplayer or multiple user virtual interactions that are possible: cooperative, counter-operative (also called versus), competitive (also called asynchronous) and mixed. Cooperative is the type of interaction where two or more players work together to complete a common goal or task. Counter-operative or versus interactions are defined by one user working against another user for a common goal only one can obtain. Competitive interaction is where users complete the same tasks and the same goals for competitive score, without any direct interaction between the players.
Mixed interactions can be defined as interactions where users work together to complete a common goal but are scored competitively or any other combination of the first three interactions.

Programming the MUVER had two parts, hardware-software communication and virtual reality software. The P5 Glove has bundled with it driver software that made the hardware-software interaction less complicated. A schematic of the hardware-software communication can be seen below.

**Figure 15: Information Communication Schematic**

The user’s movements with the P5 Glove are transmitted to the personal computer (PC) using a USB connection. The device driver for the P5 Glove, which is installed in the operating system, interprets the data and feeds it to the virtual reality software. The virtual reality software then takes the P5 data and outputs the appropriate image on the monitor. The programming of the actual MUVER software includes three different pieces that are illustrated in the figure below.
The MUVER software is composed of the game engine, the 3D models and graphics, and the scripting code. The game engine that was used in this project was Panda 3D. The code in the game engine controls many of the low level features like physics and display. The graphics and 3D models populate the engine code and follow the rules of the engine. The scripting code for the MUVER will be written in Python. The scripting code can overwrite some of the low level engine code as well as give unique features to the models. The scripting code was the major area of work in this project.
3.2 Initial Prototype Software

The initial MUVER virtual scene that was designed was created with the Blender Game Engine (BlenderGE) and the GlovePIE middleware software. Keeping the design of the virtual scene simple allowed it to be usable by a broad range of users. Several ideas for an original setting were discussed (involving simple rehabilitation movements of the hand and correspondingly simple settings and mechanics such as carnival games or simulated musical instruments). The final setting was chosen based on an earlier prototype scene (created by Thomas Dyar) of a castle with a jack-o-lantern (pumpkin that is hollowed out and cut to be used as a decoration for the American holiday Halloween). The user’s avatar in this scene was represented by a skeleton of the right hand.

3.2.1 Initial Prototype Software Mechanic Design

The rehabilitation movement was chosen to be a combination of grasp and release and supination and pronation of the hand. To match this movement a mechanic needed to be designed inside the virtual scene. A schematic of the user’s avatar’s movements through the scene can be seen below in Figure 17.
The movement in this virtual scene had 3 events. In the first event the user moved in the x-y plane to the position roughly in the middle and above the pumpkin. The second event involved the user lowering their hand in the z direction to the top of the pumpkin lid. In the third event the user gripped the top of the pumpkin and lifted off the lid by
gripping and moving in \( z \) direction. At the end of the third movement event a short reward event was seen by the user signaling that they have completed the task.

3.2.2 Initial Prototype Software Virtual Environment Design

This virtual environment was created using the BlenderGE. The 3D model assets were placed inside the virtual world using the built in coordinate system. The models were taken from a free to use model library. A screen capture of the virtual scene can be seen below.

![Image of the virtual scene](image.png)

**Figure 18: MUVER Initial Prototype Scene**
The scene was constructed using a pumpkin model in the center with a separate lid. The lid was programmed to an animation. This animation was played based on the data received from the P5 Glove and interpreted by the GlovePIE middleware. The ghost model and the far gated door were also linked to this animation because it provided visual feedback to the user.

3.3 Final Prototype Software

One proof of concept virtual environment was developed. The virtual scene used a competitive virtual interaction between two users. This scene is designed to allow practice of active grasp (pincer, 2-5 fingers with thumb) and release and maintained grasp with active supination (hand turning toward palm up). These movements are essential to improved hand function in users with stroke.

3.3.1 Final Prototype Software Mechanic Design

The mechanic was divided into phases which were timed and counted separately, in order to provide more feedback to subjects about their performance if the entire trial was not successful. These phases were 1) start position to grasp of lid; 2) grasp plus supination to threshold value and transport toward right side of screen; 3) pronation, transport back to left to position lid on pot, then release of fingers (finger extension to a threshold value). The ability to count successful phases and trials and record time for each was incorporated in the design for the scoring function. Each element can be displayed separately to subjects to provide feedback about performance either during the session or after. These elements are 1) count for number of successes for each phase; 2) count for
number of successful trials (all 3 phases completed); 3) time for each phase, and 4) time for each trial. Time can also be displayed as a mean for block of trials, with the number in block adjustable. A schematic of this movement mechanic can be seen below.

**Figure 19: Final Prototype Virtual Scene Movement Schematic**

In this scene, the user must first grasp the lid (at the correct grasping position the lid turns blue, once grasped the lid turns green), next s/he turns the lid (hand supination) while maintaining grasp and transports the lid to the right. Upon successful grasp plus supination and transport, the lid turns red. If the user looses grasp while turning, the lid drops back to the start position on the pot and returns to the neutral grey color. Supination threshold for success is currently set to 45 degrees, but is adjustable. Finally, the user pronates and transports the lid back to the pot and releases the grasp by extending the fingers, at which point the lid turns grey, and the trial is counted as a ‘success.
3.3.2 Final Prototype Software Virtual Environment Design

The virtual environment design section is split into two sub sections: visual rendering of the virtual scene and Panda 3D scripts.

3.3.2.1 Visual Rendering of the Final Prototype Software

The final prototype software was created using the Panda 3D game engine. The 3D models used came from free models available on Turbo Squid.[29] A screen capture of the final prototype virtual scene can be seen below.

![Instructions](image)

**Figure 20: MUVER Virtual Scene Rendering During a Session**

The virtual scene has a built in light that creates the shading on the models in the scene. The other virtual scene systems include a collision detection system for the lid of the
front left pot and the plane that can be seen in the screen capture. The text and score mechanic was displayed on the screen in 2D while the rest of the scene is presented in 3D. All the code for the scene was left in a single file for simplicity but future versions will have better object oriented design.

3.3.2.2 Panda 3D Scripting Systems

The scripting of the Final Prototype Software has several components which are explained below. The code presented is written in Python and also uses some functions and syntax that is native to Panda 3D.

The first pieces of code that must be placed in the file for the virtual scene are imports for the packages of Panda 3D, these will not be discussed as they are an innate part of the Panda 3D engine. Some system properties need to be set after the imports and the code can be seen below.

```python
1. props = WindowProperties()
2. props.setCursorHidden(True)
3. base.win.requestProperties(props)
4. base.disableMouse()
5. base.camera.setPos(0, -120, 120)
6. base.setBackgroundColor(0, 0, 0, 1)
```

**Figure 21: World Properties Setup Code Segment**

Lines 1-4 in the code above removes the mouse cursor from the Panda 3D window so that the user does not get confused about what their avatar is in the virtual environment. Line 5 of the code segment is used to set the camera in the virtual environment. The position of the camera is the view of the virtual world the user sees in the monitor. Line 5 of the code segment sets the background color of the virtual world to black using a RBGA (red, green, blue, alpha) color format. After these properties are set for the world the class for
the scene is constructed and then the collision traverser is created, as seen in the code segment below.

```
1. #Initialize Traverser
2. base.cTrav = CollisionTraverser()
3. #Initialize Handler
4. self.cHandEvent = CollisionHandlerEvent()
5. self.cHandEvent.addInPattern('into-%in')
6. self.cHandEvent.addOutPattern('outof-%in')
7. self.cHandEvent.addAgainPattern('again-%in')
```

**Figure 22: Collision Traverser Code Segment**

The comment character for Python is the "#" so line 1 and 3 of the above code segment are comments. The collision traverser is setup on line 2 and it is a component of the Panda engine that checks all the collision geometry every frame for a collision. Lines 5, 6, and 7 of the code segment setup collision events for movement into and out of the 3D model of the lid. Below is the code segment for loading a model.

```
1. p1 = loader.loadModel("pot1")
2. p1.reparentTo(render)
3. p1.setPos(-17, 17.5, 94)
4. p1.setScale(0.5, 0.5, 0.5)
5. p1.setTexture(tex1, 1)
6. p1.setMaterial(inactivePot)
```

**Figure 23: Model Loading Code Segment**

The Panda 3D loader is used to load all assets into the engine. In the code segment above a 3D model of a pot is being loaded in line 1. Line 2 adds the model to the scene graph which means that it will be rendered in the virtual world. The next four lines (3, 4, 5, and 6) all modify properties of the model. Line 3 and 4 change the position of the model in the virtual world and change its scale. Line 5 sets the texture for the model (the texture was also loaded using the loader). Line 6 sets the material for the pot which changes the models lighting and alpha blend (gives the appearance of a metallic shine).
1. cNode = CollisionNode('p3lid')
2. cNode.addSolid(CollisionPolygon(Point3 (-50, -50, 27), Point3 (-20, -50, 27), Point3 (-20, -20, 27), Point3 (-50, -20, 27)))
3. p3lidC = self.p3lid.attachNewNode(cNode)
4. base.cTrav .addCollider(p3lidC , self.cHandEvent)

**FIGURE 24: COLLISION NODE CODE SEGMENT**

A collision node in the Panda engine is a simple geometric shape that is used by the collision traverser to check for collisions with another collision node. In line 1 above the collision node is created and in line 2 the geometry of the collision node is created (in this example it is a plane defined by 4 points). In line 3 the collision node is attached to the target model and in the last line it is added to the collision traverser so that it can be checked each frame. In order for collisions to occur the avatar must have a task to move in the virtual environment. The code segment for the mouse position task is below in Figure 25.

1. def mouseTask(task):
2. if base.mouseWatcherNode.hasMouse():
3. mpos = base.mouseWatcherNode.getMouse()
4. mousePosX = base.mouseWatcherNode.getMouseX()
5. mousePosY = base.mouseWatcherNode.getMouseY()
6. self.hand.setPos(mousePosX*50, 20, 100+ mousePosY *50)
7. return Task.cont
8. taskMgr.add(mouseTask, 'mouseTask')

**FIGURE 25: MOUSE POSITION TASK CODE SEGMENT**

In the first line of the above code segment a task is defined that is called “mouseTask”. The next lines use the built in features of Panda to get the mouse x and y position in the window and bind them to the model of the avatar. Line 7 returns “Task.cont” which means that this task will be run every frame that the virtual world is active. The last line adds the task to the task manager so that it will be run. The code segment below contains the scripting code that controls the collisions.
The first line of the above code segment defines a collision task called “collEntry”. The next lines trigger when there is a collision with the pot lid. They change the color and material of the pot lid as well as set it so that the avatar can grasp and move and the lid. The color changing and movement of the pot lid is used as visual feedback for the user.

The code segments presented in this section outline the major Panda systems that were used in the final prototype software and how each was implemented.

3.4 Testing of the MuVER

Testing was performed on the final prototype virtual scene and also on the P5 Glove. The following sections contain information about the testing. The final prototype was tested with six subjects and the P5 Glove was tested with four. The P5 was tested after the final prototype because the analysis of the final scene data brought into question the accuracy of the data coming from the P5.
Six healthy subjects (4 males, 2 females; 5 R handed, 1 ambidextrous) performed a competitive interaction using the MUVER system. Each user donned and calibrated the P5 glove, and was instructed in the hand rehabilitation movement, and given up to five minutes to practice and become accustomed to working in the virtual environment. Next, subjects were asked to complete ten movements as rapidly as possible. A short rest was given between each trial. Total time and time for each phase was recorded for each trial and each subject.

Mean duration ± standard deviations across blocks and subjects are shown for each sub-phase of the movement task, and for the total task in Figure 3 below. Phase 1, Start to completion of grasping lid, averaged 1.6 ± 0.7 sec; Phase 2, Grasp with turn and transport to right, averaged 1.8 ± 0.6 sec; Phase 3, pronation, lid return, finger extension to release grasp, averaged 2.1 ± 0.9 sec. Mean time for the total task was 5.5 ± 1.6 sec.

**Figure 27: Graph of the Time of Virtual Environment Movement Phases**
Both experimenters and subjects noted that the mapping of real world to virtual movements was not completely accurate, which probably accounts for times that are somewhat slower than expected in healthy subjects. This inaccuracy could be due to a hardware problem in the P5 Glove position detection (explained further in the next section) or to the software that maps position and orientation data to the virtual scene elements. This issue will need to be resolved prior to testing the device with a user population. On the other hand, all subjects tested were relatively close together in performance for each phase and for total time, so the errors that are present appear to be relatively stable ones that should be fixable.

Despite these difficulties with system accuracy, the basic scene worked as intended, and the phasing feature looks promising as a useful way to provide additional feedback to subjects about performance.

### 3.5 P5 Glove Testing

After testing of the MUVER system was completed it became obvious that the P5 Glove needed to be tested separately so find out the accuracy of the data. Four subjects did a variety of tests using the P5 Glove without the MUVER system software. Instead a software package that simply gathered the raw data from the P5 was used. The testing protocol is listed below:

- The experimental group was instructed on how to safely and accurately use the virtual reality training system at Northeastern University.
- Subjects assumed a seated position in a computer chair without arms with the glove properly adjusted to their affected hand. They will be supervised at all times.
• The activities the subject performed included extension/flexion of the wrist, extension/flexion of the finger at all joints (PIP, DIP, and MCP) in variety of degrees, ulnar/radial deviation of the wrist, and gross movement of the upper extremity in the X, Y, and Z planes.

The data collected from the four subjects showed certain trends. The P5 Glove uses infrared (IR) technology to collect data on the absolute position of the user’s hand. Because of this it is important that the IR tower be setup directly in line with the user and glove. The graph below shows data for the absolute position of the glove as it is moved in the y-direction.

![Graph of X and Y Direction Movement using the P5](image-url)

**Figure 28: Graph of X and Y Direction Movement using the P5**
The movement data shifts prominently during the exercise but the movement in the x-direction is also substantial. This crosstalk could be from the position of the IR tower, user error, or just inaccuracy in the sensor. The graph below is another example of this phenomenon.

![Supination/Pronation (roll) Example of cross talk](image)

**Figure 29: Graph of Supination and Pronation with Cross Talk**

The data from the graph above was gathered during a test of roll. The other two rotation axis show movement that corresponds to the roll movement this could be due to IR tower position, user error, or sensor inaccuracy. While the position data showed signs of crosstalk and other errors the finger bending data is reliable. The graph below contains data from a test of finger bending with a large hand.
The P5 Glove uses piezoelectric sensors to gather the data for the bending of the fingers. The data is accurate and repeatable with the exception of the thumb data. Because of the design of the glove the thumb sensor has a joint that sometimes shifts preventing the bend sensor from accurately collecting data. It was found during testing that a larger hand performed better during the testing due to how the bend sensors are attached to the hand. Near the end of each bend sensor is a plastic ring that the user’s finger slides through, while the position of this ring is changeable the size is not. Therefore a larger hand is a better fit for the rings than a small hand. Data for bend sensor testing can be seen in the graph below.

**Figure 30: Graph of Finger Flexion and Extension with a Large Hand**

- Movement Units
- Time (sec)
- Thumb 1st Digit
- 2nd Digit
- 3rd Digit
- 4th Digit
- 5th Digit
The smaller hand bend data was also repeatable and accurate. The thumb data did follow the other fingers in terms of movement but simply by observing the user during the test it became obvious that the thumb sensor was not bending as much as the others.

The P5 testing was successful in illustrating the issues in accuracy that was experienced during the testing of the MUVER system. While the cause of the errors has not been confirmed the fragility of the system poses an issue for the MUVER system because it is designed for home use. These errors emphasize the importance in design of the MUVER for a specific rehabilitation exercise and the need for custom hardware which is explained in the future work section of this chapter.
3.6 MUVER Future Work

The future work for this project comes from three possible areas: the full MUVER system, designing a custom input device, and using rapid prototyping to assist immersion in the virtual environment.

3.6.1 Improvements to the MUVER System

The MUVER described in Section 3.1 is not the full possible scope for the system. A modified figure for the complete MUVER system is below. Beyond the basic system this complete system includes more options for the practitioner as well as virtual object support and a new hardware input device.
The additions for the practitioner expand on the data collection of the basic system in two major ways: providing feedback and modifying the virtual environment. The practitioner can be a spectator in the MUVER and watch the actions and interactions of the users. They also have the ability to privately or openly give feedback to a user or multiple users at once. As another form of feedback the practitioner can change or control the MUVER.
based on the actions and interactions of the users. Modifying the MUVER has several advantages including changing the level of difficulty to better suit rehabilitation and also directing the MUVER to facilitate certain interactions between users.

3.6.2 HARDWARE INPUT DEVICE

Newer technology could provide the same kind of interface for users while maintaining the high usability and low cost for the practitioner. The P5 Glove that was used in this project has 6 degrees of freedom (yaw\pitch\roll\x\y\z) that are collected by use of infrared position sensing. Using newer technology such as accelerometers or gyroscopes could decrease the errors of the data and increase the work area, it could also be wireless. The P5 uses a USB 1.1 connection, while current standards are at USB 2.0, a better USB connection allows for more data to be transferred at a faster rate. A patent search on other available input technology is needed to determine the viability of a new hardware input device.
The MUVER system requires data on the bending of the user’s fingers, the tilt of the hand (represented in roll, pitch, and yaw), and the absolute position of the hand in the workspace (represented in units of x, y, and z) that will be provided by the input device (i.e. glove). Bend sensors will be used to collect data on finger movements. The tilt of the hand will be collected using either an accelerometer or tilt sensor. The absolute position of the hand is the most difficult data to gather. Other devices have used infrared (IR) to gather absolute position (P5 Glove and the Wii Remote) or magnetic fields (Polhemus). Each of these solutions has problems like the requirement for line of sight for IR or the safety concerns of strong magnetic fields in the home setting. In order to overcome these issues, a different system will be used with the low cost device. Built into the software will be a starting position for the user to begin the exercises in the game. This starting position can then be used as a starting point to calculate absolute position.
An accelerometer that calculates relative position can be coupled with a known starting position to calculate absolute position. The issues involved in a system like this include drift from the accelerometer and the necessity of a separate structure to use as a starting/resting point.

The glove will contain haptic feedback (vibrations) on the finger tips to augment the visual feedback of the software and give a better experience to the user. The device will be easy to put on and adjustable for both the left and right hand and for a range of hand sizes.

3.6.3 RAPID PROTOTYPING VIRTUAL OBJECTS

Rapid Prototyping (RP) is a manufacturing method that allows for custom objects to be created from a .STL file quickly and for low cost. The MUVER is populated by models that can be exported as .STL files. Therefore the practitioner can take an object from the real world and recreate it in the MUVER for virtual rehabilitation and then create it using RP for use in actual rehabilitation exercises. By using items created in RP that are from the virtual environment it can enhance the immersion for the user.

3.7 MUVER SUMMARY

The Multiple User Virtual Environment for Rehabilitation (MUVER) system has a successful proof of concept virtual scene. This scene is based upon a first prototype scene created in the BlenderGE and using the GlovePIE middleware to pair the P5 Glove to the virtual scene. Testing was performed on the virtual scene using six subjects. The subjects were required to do all phases of the movement mechanic and their times were
recorded. During this testing errors in the data were found so testing of the P5 Glove was performed. The P5 Glove testing showed that the movement axis were prone to error and inaccuracy but the finger bend sensors gathered reliable data on all fingers but the thumb. Future work for the MUVER system includes additions to the systems networking as well as creating a custom hand device that will replace the P5 Glove. The MUVER system shows promise for patients that are post stroke but still want to continue their rehabilitation in their homes.
CHAPTER 4: VIRTUAL ENVIRONMENTS FOR THE ACTIVE HAND REHABILITATION INTERFACE

This project began in July of 2008 and at the time of writing is still ongoing [30]. The device hardware was designed by Brian Weinberg and the control software was created by Ozer Unluhisarcikli. The project was assisted by staff at Spaulding Rehabilitation Hospital in Boston, MA and Dr. Paolo Bonato. This chapter is segmented into three major sections: Hardware and controls, virtual environment design, and testing. The virtual environment design section contains information about all versions of the software.

4.1 ACTIVE HAND REHABILITATION SYSTEM HARDWARE AND CONTROLS

This section has two subsections, one for the rehabilitation device hardware and one for the device’s closed-loop control software.

4.1.1 ACTIVE HAND REHABILITATION SYSTEM DEVICE HARDWARE

The Active Hand Rehabilitation System (ARHI) utilizes Electro Rheological Fluid (ERF) to create a hydraulic two degree of freedom (2DOF) device. The 2DOF that are possible with the device are grasp and release of the hand and supination and pronation of the wrist. A figure showing the global system is shown below.
Figure 34: Active Hand Rehabilitation Interface Global Schematic
The hardware of the ARHI consists of several custom pieces of hardware as well as off the shelf components. The system has two computers, a real time target system which runs a real time operating system and a host computer which runs the actual interface and the VR software. More information about the software systems can be found in the following sections of this chapter. Connected to the real time target computer is the data acquisition hardware which is connected to the actual hand device’s sensors. The device has a strain gauge, load cell, and other sensors that collect data about the force, torque, and position of the user’s hand on the device. The device itself is composed of a rotary and linear hydraulic actuator and damper system. Other hardware includes an amplifier, high voltage power supplies, and a custom made handle system. A figure showing the system setup on as a desktop system can be seen below.

![Figure 35: Active Hand Rehabilitation Interface](image-url)
In the setup in the figure above the system is placed on the right side of the host computer. The user sits a comfortable distance from the device and uses their right hand to control the device. The VR software outputs to the monitor to give the user and practitioner feedback on their performance. A close photograph of the device can be seen in Figure 36 below.

**Figure 36: Two Degree of Freedom Robotic Hand Rehabilitation Device**

The device consists of a linear and a rotary ERF actuator. The actuators were designed custom for the device by Brian Weinberg. Although it is possible to provide the assistive force with conventional actuators such as voice coils and hydraulic cylinders, the use of an ERF actuator has several superior features. First, it has a high force density (up to 500 N in a compact design) and second it has a high response rate to varying electric fields which enables smooth control of forces during rehabilitation sessions.
An ERF actuator was designed as the principle component in the device. The hydraulic actuator uses electro rheological fluid as its control fluid and performs exceptionally compared to standard hydraulic systems due to the proximity of its control valve. The control valve of the ERF actuator is situated inside the actuator (opposed to externally in a manifold) and it is controlled by varying the electric field across the valve. This change in electric field changes the yield stress of the ERF fluid, which modulates the pressure drop across the valve. Since the valve is inside the actuator, close to the piston and the ERF reacts in milliseconds, precise force control is possible. The control systems of the device are discussed in the next section.

4.1.2 Device Control

The software of the AHRI runs on a real-time (RT) platform because regular data acquisition (DAQ) hardware running on a general-purpose OS, such as Windows, could not guarantee performance. In contrast, RT hardware running on a RT operating system (RTOS) allows the programmer to prioritize tasks so that the most critical task (such as differentiation or controls) can always take control of the processor when needed. This property enables reliable applications with predictable timing characteristics.

The code is developed on the host, and then deployed to the RT target. Communication between the host and RT target is via high speed Ethernet. The RT target streams critical parameters back to the host for monitoring. The host displays and saves the streamed data, and also displays the user and practitioner interfaces. To facilitate cost reductions and portability, future versions of the system can utilize board level RT targets that
feature FPGA chips and single board computers. These solutions offer a significant packaging advantage at the expense of flexibility. A block diagram of the device’s controller can be seen in the figure below.

**Figure 37: The Generic Control Diagram of the System**

A closed-loop controller uses some sort of feedback to compare what is actually happening to the expected response. Almost all of the automatic control processes in the industry are operated by Proportional-Integral-Derivative (PID) controllers. These controllers act upon the error in the process (which is the difference between the desired response and the actual response) in one of the three ways (or a combination of them).

The control action that is directly proportional to the error is called the “Proportional Action”. The magnitude of this proportion is called the “Proportional Gain”, or $K_p$. This is the main driving force in a PID controller. A higher gain in the system leads to a faster response. But there is a limit to the admissible gain, due to the concept of instability. In the case of instability the control action becomes self-growing, resulting in large oscillations in the system. Delays and/or noise in the feedback signal may be the source
of instability. Another control action is the “Integral Action” where the controller outputs a signal that is proportional to the accumulated error. Integral controller eliminates steady-state error, but adds a lag to the system. Integrator wind-up should also be accounted for while implementing these controllers. The final control action is the “Derivative Action”. It acts on the rate of change of the error, which is predictive in nature and thus adds lead to the system. However, derivative controllers are highly susceptible to noise. A block diagram of the PI + Feed Forward controller can be seen in Figure 38 below.

![Block Diagram of the PI + Feed Forward Controller](image)

**Figure 38: Block diagram of the PI + Feed Forward Controller.**

The output is saturated to prevent excessive commands that might otherwise damage the system. The inverse model calculates the required voltage for the desired force/torque, which is used as a feed forward term.

After conducting many experiments, the controller gains were selected $K_p = 0.5$, and $K_i = 0$ for the linear actuator. The proportional gain was optimized for accuracy and the
smoothness of the feeling (there is a trade-off between them). Although higher gain gives better accuracy, it also creates larger oscillations in force. The integral gain was set to zero due to issues related to actuator saturation, wind-up, and increased oscillations.

4.2 Virtual Environment Software for the ARHI
For advanced rehabilitation devices it is important to have state of the art software interfaces for both the practitioner and the user. This requires the use of multiple software packages to handle the necessary functionality. Communication between these programs is vital to the overall experience for both the practitioner and user. There are several possible communication types available for this system. However, since all of the software for the system was being controlled from a single Host Computer the best solution is to use User Datagram Protocol (UDP). UDP communication does not require a connection transmission so only the data needs to be sent in a string format. Also the speed at which UDP data can be sent far exceeds the speed of most 3D engines. The figure below shows the global data flow for the system.
The sensors on the input device controlled by the user send data to the RT system. The RT target is connected to the host computer and the control software. The control software outputs the practitioner interface and also sends data via UDP to a 3D Engine. The 3D engine creates the virtual reality scene which is the user’s interface.

The data acquisition software used is LabVIEW and the 3D game engine is Panda 3D. Panda 3D is scripted in Python so the most elegant solution is to use the built in socket and UDP application programming interfaces (APIs) available in Python. Currently Panda3D is only receiving data and LabVIEW is only transmitting.

**Figure 39: Software Communication Global Schematic**
The LabVIEW Virtual Instrument (VI) for the 2DOF device is written in LabVIEW Real-Time and uses shared variables for communication between the Real-Time and host systems. The code responsible for the UDP communication with Panda 3D is on the host machine. LabVIEW has built in capabilities for creating UDP sockets and sending information in the form of a string of characters. A screen capture showing the block diagram of a simple LabVIEW UDP sender VI can be seen below in Figure 40.

![Block Diagram](image)

**Figure 40: Simple UDP Sender Block Diagram**

The gray rectangle is a while loop in LabVIEW. Anything on the left side of the loop is run before the loop beings and anything to the right is run after the loop is terminated. The block on the left side of the while loop creates a UDP socket at port 2056 (2056 is arbitrary but it must be different than the target port). Once inside the loop the block used is a UDP write block. This block takes the data in the form of a string and sends it to the internet protocol (IP) address and port specified. Once the loop is terminated the block on the right side of the loop closes the UDP socket.
The build in UDP functionality of LabVIEW makes communication to the game engine simple to implement, control, and expand.

4.2.2 UDP Communication in Panda 3D

The UDP communication for the Panda 3D game engine is carried out by two segments of code. The UDP socket code used is from the Python socket interface and not the Panda 3D networking classes. The code segment that creates the UDP socket can be seen below in Figure 41.

```
1. self.UDP_IP="localhost"
2. self.UDP_PORT= 2055
3. self.sock = socket.socket( socket.AF_INET, socket.SOCK_DGRAM )
4. self.sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
5. self.sock.setsockopt(socket.SOL_SOCKET, socket.SO_RCVBUF, 512)
6. self.sock.bind((self.UDP_IP, self.UDP_PORT))
7. self.sock.setblocking(0)
8. self.sock.settimeout(.000000000000000000000001)
```

**Figure 41: Code Segment for Creating a UDP Socket**

The first line of the code segment sets the target IP address. Because the communication is taking place on the same computer the IP address is simply “localhost”. In the next line a port is chosen (2055 has no significance except that it must match the port that the LabVIEW VI is sending data) for the socket. The next three lines change or set properties of the socket so that it performs as needed. Line 6 binds the IP address and the port to the created socket. The last two lines deal with how the socket is shut off. By setting a very low time out the socket the rest of the program will not be impacted if data is slowed or stopped to the socket. The code segment below is placed in the program after the setup code and contains the function that receives the data.
The first line of this segment is the method declaration and it creates a method called “lookForData” and takes in as input a Panda 3D task (making this method a Panda 3D task). Inside the task the “try” command is used to try to collect data from the UDP socket created in the previous code segment. The data received is a string type so it is recast as a float in line four. Then the data is used to move the player’s avatar on the screen. The “except” command on line six is implemented so that if no data is received from the socket the loop does not pause for data and completes. By returning “Task.cont” on the last line Panda 3D knows to run this method continuously.

The UDP code that is used inside Panda 3D is a small part of the overall program but it is of upmost importance because without a simple form of communication between the input device and the game engine creating a virtual environment is not possible.

4.2.3 Initial Prototype Software

The major design considerations for this device include having a customizable world design so that the practitioner can choose to use either both or a single degree of freedom. A diagram of the prototype virtual environment for the ARHI is below.
The user’s avatar was represented as the green circle in this virtual environment. They use the device to move around a track designed by the practitioner. Feedback is given to the user in the form of accuracy in movement and also in the speed the user completes the track. The red lines represent the start and end of the track and this can be modified depending on what the practitioner wants the user to do for an exercise. The yellow ellipse is a force gradient that the user tries to stay inside as they move around the track.

4.2.4 Final Software Design

In order to create a software package that gives the practitioner the most flexibility it is necessary to have a way for the practitioner to create unique virtual reality scenes for users. To facilitate this, a separate software program is needed.
The virtual scene design software needs to be simple for the practitioner to use and also have the data from it be read by both the virtual reality software and the LabVIEW control software. In order to do this and also have the data be savable the design software must create a file that is read by the other programs. The simplest solution is to have the design software write a text file that contains the data necessary for the other programs to recreate the scene. The data that each scene will need includes the position of the start and end points for each line segment as well as their “feel”, the equation for each line segment, and the “feel” of each line segment.

**Figure 44: Software Communication Global System Schematic**
The practitioner will be able to create the scene in the design software much like a paint or art program. Once the practitioner is satisfied with the design they will save it in the design program. The design program will then create a text file that contains the necessary data for that scene. This will be done by calculating the start and end points of each segment as well as its equation. The “feel” was chosen by the practitioner during design.

Once the data is calculated the design software will write it to text in a way that is both readable in a normal text format and also readable by the other programs. The virtual reality software will read in the text file and then create the scene geometry based on the data in the file. The control software will also read in data from the text file of the scene and create the necessary feel and forces that the scene requires. Because the design software creates a text file for each scene it is savable and reusable by the practitioner. A diagram of the possible virtual mechanics for the device is below.
With two degrees of freedom there are several possibilities for virtual reality scenes. The first possible design is to use both degrees of freedom to control X and Y position in a Cartesian coordinate system. The second possible design uses one degree of freedom to control direction and another to control velocity. An example of this design would be to use the supination-pronation degree of freedom for direction and the grasp and release for velocity. The third design would be to use neither degree of freedom for position, direction, or velocity and instead use them to control another aspect of the virtual reality scene.

Based on the unique characteristics of the device there are also several possible “feels”. The first is to have the force field position pull the device to the desired position. The second is to use the force field to push the device away from the desired position.
The user controlled position can be directly or indirectly specified. For the direct method, each position of the handles corresponds to a specific position on the screen. When the user moves the handle, the represented position follows the handles exactly. For the indirect method, there is an equation of thrust or something similar that drives the represented position. The representative dot is given mass; the corresponding friction and inertia is calculated, and the position changes accordingly. The user controls the level and direction of thrust with the handles. Either of these methods can be applied to one or both of the DOF of the device.

The final prototype virtual scene was created in the Panda 3D game engine. A system in Panda 3D was used to create the background of the maze dynamically. This allows for multiple mazes to be setup and used easily without extensive changes to the code. Also an algorithm was created to place the red and green items on the maze in a random position so that each time the maze is used it offers a different experience to the user. A screen capture of the final prototype can be seen below.
The player’s avatar in this virtual scene is represented by a blue circle with a lighter blue rectangle on top. The blue color was chosen to be different from the red and green items that are spread throughout the maze. The lighter blue rectangle on the avatar is used to show the direction that the user is moving.

The green items on the maze are placed either randomly or using a separate text file of x and y coordinates. The avatar is meant to collide with the green items to increase the user’s score and motivate them to do movements using the device. The red items act in the opposite way of the green ones. They too are placed in the maze either via random generation or by a separate text file. The red items lower the score of the user and the
avatar should avoid them, forcing the user to go around and perform rehabilitation movements on the device.

The user interface for the prototype software is basic and simple so that the user can easily understand both their score and the time remaining. The time is displayed as a rectangle that decreases in size and changes color based on the time remaining in the session. Below is a screen capture of the advanced maze in the final prototype software.

![Screen capture of the advanced maze in the final prototype software.](image)

**Figure 47: Advanced Maze for ARHI Final Prototype Software**

The advanced above was the first maze implemented into the final prototype software. The design was created from practitioner feedback on what path density and level of
complexity they wanted. The green and red items that are displayed can be randomly placed or placed by an eternal text file (further explanation in the next section). The diagonal paths that were implemented into this maze presented the largest challenge for subjects and it is because of this high level of difficulty that simple maze was created and implemented.

4.2.4.1 Panda 3D Scripting Systems for the ARHI Software

After gaining the experience of using Panda 3D in the MUVER project it was an obvious choice to use for the ARHI final prototype software. This section presents code segments that were used to implement various features and mechanic of the virtual world.

One of the most important mechanics in the virtual world was to create and maintain an array of red and green items that will populate the maze and that the avatar can collide with. The code segment to create these arrays is below.

```
1. for i in range(self.goodPUtotal+self.badPUtotal):
2. self.listX.append(0)
3. self.listY.append(0)
4. self.listX1.append(0)
5. self.listY1.append(0)
```

**Figure 48: Item Array Code Segment**

Line 1 of the above code segment contains a for-loop that is run for every red and green item that is going to be placed into the virtual environment. The next 4 lines append zero values onto arrays so that they are the appropriate length to hold all the coordinates necessary to create the items. The code segment in Figure 49 changes some of the virtual world properties.
This code segment creates and applies an orthographic lens to the camera that is placed in the virtual world. The orthographic camera is used to make the virtual world appear to be 2D. While some elements of the world are 2D others are 3D and the orthographic camera completes the 2D illusion by not having models converge at the horizon simulating depth.

The collision handler in this virtual environment is nearly identical to the one used in the MUVER final prototype software. The first line creates the collision handler event and the next three lines create collision events for when the avatar collides with in world items. The last line creates the collision traverser that will check each frame from collisions between pieces of collision geometry.

One of the main features of the ARHI final prototype software is that maze can be implemented and switched quickly. This code segment is the backbone of that mechanic.
By using the GeoMipTerrain feature of the Panda engine it becomes very simple to calculate the elevation of a certain pixel on the maze. By using this elevation is becomes very easy to determine if the avatar has collided with a wall. The first line of the code segment creates the terrain object and the next six lines set properties of the terrain to ensure its proper usage. The last line of the code segment places the terrain on the scene graph but without the “terrain.Generate()” command the maze will not be visible on the screen. This is done so that a more simple geometry can be used for the maze so that the virtual environment has a better performance. The code segment for rendering the maze is below.

```python
1. self.level = loader.loadModel("plane")
2. self.level.setPos(512.5, 485, 512.5)
3. self.level.setHpr(0, 0, 0)
4. self.level.setScale(1025, 1, 1025)
5. self.level.reparentTo(render)
6. self.levelTex2 = loader.loadTexture("/c/mazePrototype.jpg")
7. self.levelTex = loader.loadTexture("/c/maze3.jpg")
8. self.level.setTexture(self.levelTex, 1)
```

**Figure 52: Maze Creation and Rendering Code Segment**

Much like the model segment in the MUVER Chapter the loader is used here to import an asset called “plane”. This 3D model is a very simple rectangular prism that is used to display the 2D overhead image of the maze. This is done in line 6 of the code segment. The other lines in this code segment modify the properties of the 3D model so that it is placed correctly in the virtual environment and that it is rendered.
The main reason for creating a GeoMipTerrain object is to use the “getElevation()” function that is built into Panda. It is a simple and fast calculation that returns a number based on the color of a pixel at a creation position. By using black and white maze images it becomes very easy to determine if the avatar is on the white path or in the black area. This code segment uses the “getElevation()” function to place items randomly onto the maze. By checking a randomly created position against a list of conditions the system is able to find positions to place the items in the paths at a minimum distance from and other items. After all the item locations are randomly created, the items themselves need to be loaded and that is completed by using the code segment below.

```
1. good = render.attachNewNode("ballPH"+str(i))
2. good.setX(self.listX1[i]/2)
3. good.setY(473)
4. good.setZ(self.listY1[i]/2)
5. self.goodPU1.instanceTo(good)
6. good.reparentTo(render)
```

**Figure 54: Item Creation Code Segment**

In this code segment a new node is created and added to the scene graph. The 3D model for the item is then attached to node. The next lines modify properties of the model like position and scale. The last line sets the 3D model to be rendered inside the virtual world.
The code segments presented in this section outline the major mechanics and functionality of the final prototype software. After the creation of the virtual scene was complete testing was performed as discussed in the next section.

4.3 Testing of ARHI Software

Testing was performed by three subjects. All subjects completed the maze with both their left and right hands. Data for the torque and position of the device during the test was recorded as well as the total test time. The x and y position of the avatar on screen was also recorded but not used in the analysis. The graph below shows the total time in seconds that each subject took to complete the maze.
There are several factors that impact the time data in the graph above. The first factor is the subject’s familiarity with the maze. The first two subjects had prior experience with the maze and it is expressed by them having lower times for both their left and right hand. Also subject three’s times were affected by the maze experience which resulted in a nearly 20 second drop in time between the first and second test. The other factor of note is the dominant hand of the subjects. For subject 1 and 2 their dominant hand provided a faster run through the maze. The graph below contains the data on the average force seen during the testing.
The average force for the testing showed that the each subject (except for subject 3) had a slight increase in average force from the first test to the second. The dominant hand of the subjects did not play a role in their average force. Also the time to complete the exercise did not affect the average force during the test. The average torque for the testing can be seen in the graph below.
The data once again varied wildly with the third subject. This could be due to resetting the device in between tests or some erroneous data being recorded. As for subjects 1 and 2 they show a much closer variation between tests. It also appears that dominate hand and total time could affect the average torque for the test. Also as with the average force, subject 2 showed slightly smaller average torque than subject 1.

Testing of the ARHI device showed that there are several factors that will affect the time, average force, and average torque. These factors include the dominant hand of the subject and if they have had prior experience using the device or navigating the maze.
Further testing of the device is needed to show how the virtual environment impacts the user’s performance while using the device.

4.4 ARHI SOFTWARE FUTURE WORK

There are two main areas of improvement for the VR software of the AHRI: addition of more mazes and addition of more item mechanics. The difficulty curve for the user was a point of concern in the final prototype software because the two mazes that were created were very complicated. There was concern about the user getting bored with just the red and green items in the maze so more items need to be developed.

The maze progression in the final software must be more gradual for users. Because the software is setup to easily load and display any map this is a simple step that would make the overall software better for both the practitioner and user. A schematic that shows a gradual map progression can be seen below.

![Figure 58: Maze Progression Schematic](image)

The simple progression must not only incorporate a smooth increase in complexity to the mazes but also provide a smooth increase in difficulty to the rehabilitation exercises that match the maze’s shape. This is completed by using the same basic shapes during the
progression of difficulty and adding to them more complex shapes. Along with this maze progression there needs to be more items in the maze to provide motivation and fun to the user. A schematic of one such item can be seen in Figure 59 below.

**Figure 59: Venture and Return Mechanic**

In venture and return mechanic that is seen above is one such item that could be added to the maze. In this mechanic the avatar moves down a length of the maze that is a dead end in order to pick up an item at the end. Once the item is collected by the mechanic is triggered and another object moves down the dead end section of the maze and forces the user to perform a movement under a time constraint. This is useful because it forces the user to move in both directions equally which is important for rehabilitation and range of movement. Other items can also be added to these mazes and three examples of such items can be seen in the figure below.
Currently in the virtual environment there are only items that modify the score of the user. Other items can be added to the virtual environment to modify other properties of the avatar such as its speed, direction, and size. Also items could modify the force that the device exerts or the range of motion of the device. By adding in these other items it not only makes the experience more fun and interesting to the user but the items can be used to influence rehabilitation.

4.5 ARHI Summary

The Active Hand Rehabilitation Interface (ARHI) is composed of a RT control system, a host computer, a two degree of freedom electro rheological fluid based hydraulic system, and virtual reality software. The hardware and controls were created by other members of the Biomedical Mechatronics Laboratory.

The software created was created in Panda 3D and uses several of the built in systems of Panda 3D to facilitate the use of the virtual environment. The software has a maze generation system that makes it simple to manipulate and change mazes. It also has a mechanic to create randomized items on the maze. There are also timing and score mechanics implemented into the system.
Testing was performed on the system using three subjects. They were told to complete the simple maze as quickly as possible and they performed the maze with both hands. Also the force of the device was standard across all three subjects.

Future work for the ARHI software includes creating a smooth maze complexity as well as adding more items and mechanics to the maze so that the user stays motivated and does not get bored.
CHAPTER 5: INTERFACE SOFTWARE FOR THE VREHAB BICYCLE SYSTEM

This project started in December of 2008 and the first prototype, which is discussed here, was completed in July of 2009. The project had three major components: hardware, interface software, and virtual reality (VR) software. The hardware for this project was designed by Richard Ranky and is composed of 3 major systems: handlebars, pedals, and supporting electronics. The interface software is composed of two LabVIEW Virtual Instruments: Signal Interface and the Main Interface. The VR software was created by a third party developer.

5.1 VREHAB BICYCLE SYSTEM HARDWARE

Several hardware and software elements were implemented into a stationary exercise bicycle to monitor physiological parameters of users post-stroke whilst immersing them in a virtual simulation providing visual and haptic feedback. This system is attachable to current commercially-available stationary bicycle systems and interfaces with a personal computer for VR simulation and data acquisition processes. The figure below contains a global schematic of the system.
The signal box (D) acquires and amplifies data from two identical Handlebar Systems (A), two similar Pedal Systems (B), a heart rate monitor system (C). The signal box then sends the data signals to the practitioner interface (F). All components are tethered and powered by the power (E) & signal boxes with the exception of the heart rate monitor which is wireless. The VR software (G) is controlled by the data collected from the users movements on the stationary bicycle.

The handle and Pedal Systems require rigid attachment to the stationary bicycle. The handles have adjustable Velcro fasteners to mount correctly to the range of handlebar
diameters currently available. By using Velcro it is also possible to rotate the Handlebar Systems for a custom rider posture during a rehabilitation session. The pedals systems are fitted with the standard threads for stationary exercise bicycles, and can readily replace the standard bicycle pedals via attachment to the crank arm.

The following sections present information on the design and functionality of the components that compose the bicycle system.

5.1.1 Handlebar System

The two sections below outline the design of the Handlebar System and how it was calibrated. More information on the how the data is collected and manipulated can be found in Section 5.2.2.4.

5.1.1.1 Handlebar System Design

The Handlebar Systems for the right and left hand are identical. Rapid prototyping was used to create the main body of the Handlebar System. Deformable tubing was used in the grip channels and a piezoelectric hydraulic pressure differential sensor was used to collect the grip pressure. A schematic of the Handlebar System can be seen below in Figure 62.
A. Handlebar Housing with embedded channels for tubing
B. Deformable PVC tubing
C. Watertight attachment ports for tubing to connect to reducing elbows (E)
D. Piezoelectric Hydraulic Pressure Differential Sensor
E. Reducing Elbows
F. Handle Caps with embedded channels for tubing and rungs for adjustment straps
G. Tubing Plug

**FIGURE 62: SCHEMATIC OF HANDLEBAR SYSTEM**

The figure above indicates how application of force (blue arrows) on one chamber causes the tube to compress and build up pressure at one end of the pressure sensor. The sensing area of the handle is the surface area of the three exposed tube sections between the handle caps. A key design goal was to minimize the loss of pressure transmission during load application. When a load is applied over the pressure sensing area, any tubing not under direct compression will expand, causing loss of pressure transmission. Therefore tubing under the handle caps has been constrained and tube lines outside of the handlebar housing (C, G) has been minimized and plugged.

The tubing (B) is constrained inside the channels between the handlebar housing and the handle cap (C). The dimensions and cut depth for the channels (C) constrain the tubing
clamped underneath in a slightly compressed state. The channels are designed to constrain the ends of the tube using their minimum bend radius and provide an outlet for the plug and reducing elbow connector. Since the tubes are pre-filled during assembly this compression causes a pre-load on the tube plugs and slight expansion of the unconstrained tube.

Each of the two hydraulic chambers is comprised of a single length of tube which is guided along channels in the housing and handle cap to run back and forth. The section of tube underneath the handle caps is rigidly constrained to prevent loss of pressure from tube expansion. In order to maximize the effective surface area for sensing, three channels were designed into the housing with loops at each end to allow the tubing to follow the bend without kinking. A figure showing a rendering of the final prototype and a picture of the prototype can be seen below in Figure 63.

![Figure 63: Rendering and Physical Prototype of the Handlebar System](image)

The Handlebar System was used in the VR simulation to control the direction of the avatar. For this reason the accuracy of the sensor and the size of the grip area were very important. This design fulfills the needed requirements of both the interface and VR software.
5.1.1.2 Handlebar System Calibration

The handlebars were calibrated to match the force applied over the tubes to the voltage resulting from the pressure in the hydraulic chambers. A calibration paddle was fabricated to match the shape of the three tubes to evenly compress them over the length of the sensing area.

![Diagram of handlebar system calibration](image)

**Figure 64: Calibration Hardware**

The calibration sequence consisted of aligning the paddle surface against the chambers and compressing them by 91N (9% of maximum comfortable loading combined isokinetic grasp with shoulder reaction force). The initial pressure offset was electronically adjusted to zero and the load was applied. Maximum voltage readings from the hydraulic pressure sensor were matched with the tensile force from the load cell.

This calibration created constants that were implemented into the interface software. Each Handlebar System has two calibration constants, one for the top of the Handlebar System and one for the bottom.
5.1.2 Pedal System

The following section contains information about the design of the Pedal Systems and how all the sensors and components are placed. The Pedal Systems are the most complex components in the bicycle system because they have sensors that measure force, tilt, and revolutions per minute as well as containing vibration elements to provide tactile feedback to the user.

5.1.2.1 Pedal System Design

The right and left Pedal Systems are not identical like the Handlebar Systems. The right Pedal System includes a Hall Effect sensor used to calculate the revolutions per minute (RPM) of the crank. A rendered view of the left pedal assembly can be seen in Figure 65 below.

![Figure 65: Render of the Left Pedal Assembly](image)

The Pedal Systems were designed to measure lower extremity forces and range of motion of the user during a rehabilitation session on the bicycle system. The Pedal System requirements include:
• Measure Compressive and Tensile Forces from the Feet
• Measure Range of Motion for Dorsi and Plantar flexion During Exercise
• Measure Rotational Velocity of the Pedal
• Exert Haptic feedback to the Foot
• Securely hold The User’s Without specialized Footwear
• Interface Easily with any commercial Stationary Bike

To allow the pedal to easily attach to a commercial bike, the pedal design was centered around an existing pedal with the standard 9/16” x 20 thread which fits all adult bikes with two and three-piece crank assemblies. The raceways of these pedals have built-in roller bearings and four attachment points for a cage or toe clip. A global schematic of the pedal assemblies can be seen in Figure 66 below.
To provide tactile feedback to the user, vibration elements (A) were mounted on the pedal bindings. The most compact vibrating element was the Precision Microdrives 310-101. Two of these motors were encased and attached to the inside of the bindings with Velcro. In case of any loss of sensitivity to the dorsal side of the user’s foot the vibrating elements could be relocated across the front and back edges of the bindings.
To detect static pedal tilt, it was necessary to mount an accelerometer (F) on the pedal. A range of accelerometers were evaluated and a 2-axis ±5g accelerometer was selected. When mounted, the Y axis of rotation for the accelerometer was aligned with the axis of rotation of the pedal raceway.

To detect RPMs, a latching single phase Hall Effect sensor was used in conjunction with 4 magnet posts. The sensor (I) was mounted on the right pedal 17mm below the raceway axis of rotation and revolved with the pedal around the crankshaft. The four magnet posts were mounted to the exercise bike housing in opposite alternating poles to mark top dead center, 90° CW, 180° CW, and 270° CW.

As the pedal passes next to one of the posts the Hall Effect sensor, it registers the change in polarity of the magnetic field and switches its signal for the digital input. Each time it registers a change in the field it has travelled 90°. The time in between the magnets is used to calculate rotational velocity of the crank.

The pedal rpm input is used to propel the virtual bike in the simulation. Since there is no instrumentation on the stationary bike for gearing or wheel radii this does not yield a velocity value by itself. Therefore the practitioner may adjust a multiplier gain to the user’s rpm to generate an artificial velocity in the simulation.

When the virtual rider turns off from the path the vibrating elements activate to stimulate the rider to return to the designated trail. The combination of haptic and visual feedback has a stronger combined impact than either stimulation alone.
The load cell (J) implemented was a Honeywell Model 13 Subminiature Load Cell. The low profile and durability was an appropriate match to the pedal design when compared to the other sensors examined. The selection criterion was also narrowed by only requiring compression sensing rather than bi-directional.

To secure the user’s foot to the pedal a specific binding (K and L) scheme was required. A readily adjustable, robust, comfortable fit was required to secure the metatarsal-phalanges joint just above the pedal’s axis of rotation. This was successfully achieved by using Flow Flite 4 bindings. The Flow bindings strap across the dorsal side of the wearer’s foot from the base of the Internal and Middle Cuneiform down to the middle of the metatarsals.

5.1.3 Heart Rate Monitor

A wireless heart rate receiver was necessary to drive a component of the virtual environment. The RE07L Wireless Receiver Module and T31 coded elastic chest band were selected. The chest band is worn during exercise with the transmitter in skin contact just below the center of the sternum, detects each heartbeat and outputs for each heartbeat. The combination of these two components fulfills the selection criteria for the heart rate monitor in a cost-effective way:

- Sensitivity unencumbered by activity level
- High reliability under elevated levels of moisture and sweat
- The wearer is not hindered in performing their tasks from discomfort or constrained motion
- Low power and bandwidth requirements
The system in this device has a coded communication to improve noise reduction and cross-talk from other sensors by automatically changing the communication frequency when in close proximity to the chest band. It has an operating range of 80-105cm and operating frequency of around 5kHz. The chest band outputs three pulses for each heart beat detected, of which only one needs to be detected by the receiver to register a heartbeat. The highest heart rate for a healthy human is on the order of 240 beats per minute (4Hz, 250ms window each pulse) which is within the Polar system’s operating frequency of 200ms. As part of the frequency matching sequence, during initial startup of each exercise session the chest strap must begin within 50cm from the receiver for approximately 5 heartbeats.

Before an exercise session, the practitioner will have set the target heart rate of the user. This target heart rate controls the position of a pace rider which the user must keep up throughout the exercise. The difference between the target heart rate and the measured value from the user determines the location of the pace rider relative to the user’s virtual rider. This location may be in front or behind, depending on which value is greater.

5.1.4 Other Hardware Components and Systems

The heart rate monitor, Pedal Systems, and Handlebar Systems require supporting electronics for proper operation. These electronics include signal amplifiers, a data acquisition system, and a signal electronics enclosure.
5.1.4.1 Signal Amplifiers

Strain gauge amplifiers were used to amplify the signal outputs from the pedals’ load cells and handlebars’ pressure sensors. These amplifiers use a full Wheatstone bridge and have an operating range of 8-30V DC with a built-in 5V regulator for the sensor excitation. They can operate in single-ended mode (for the load cells) by bridging the GND and V- terminals or bi-polar mode (for the handle bar pressure sensors) using a negative voltage supply.

**Figure 67: Schematic of SGAU strain gauge signal amplifier**

The SGAU circuit assembly has a fixed gain resistor of 100 ohms in series with a 1K trimmer potentiometer (VRG), allowing the amplifier gain to range from 1000 with the trimmer fully clockwise (100 ohms) to 90.9 with the trimmer fully counterclockwise (1100 ohms). The voltage signal offset may be adjusted by turning the VRO terminal.

The span of the load cell amplifiers were shunt calibrated to each load cell using a 59kΩ resistor to bridge the E- and S- terminals and then adjusting the voltage offset to zero when the resistor was removed.
5.1.4.2 Data Acquisition System

The data acquisition system used in this device was the NI USB DAQ 6008. This device afforded the flexibility of a range of analogue and digital input devices, whilst also supplying a +5V excitation to the smaller devices like the heart rate monitor and vibrating elements. It was connected with a common GND to the amplifier units for the handlebar and pedal force sensors.

5.1.4.3 Signal and Power Boxes

Figure 68: Overall View of Power and Signal Boxes

The power box used in figure 19 was a LOGISYS ATX12V (LOGISYS, Pomona, CA, USA) to provide GND, +5V excitation, and ±12V excitation to the system. Inside the signal box four amplifiers (see section 1.4.1) are connected to the handlebars and pedal load cells. These four elements are tethered to the external sensors through the front of the housing along with the pedal accelerometers, Hall Effect sensor, vibrating elements, and heart rate receiver module.
The USB DAQ system is also contained in the signal box, connected to the analogue &
digital sensor ports, and connected with common GND terminals to the power source and
amplifiers.

5.2 VRehab Software

Two LabVIEW Virtual Instruments (VIs) were created for this project. The first is the
Signal Interface VI and the second is the VRehab Main Interface VI. Both of the VIs use
User Datagram Protocol (UDP) to send information from the VI to the Virtual Reality
Software that was created by a third party developer. The flow of information throughout
the system is outlined in Figure 69 below.

![Diagram of VRehab Software](image)

**Figure 69: Information Communication Diagram**

The system is first configured by the Practitioner using the Configuration Interface which
is a LabVIEW VI. Once the session begins information about the user is transmitted by
the sensors mounted on the bike. The information is sent via the Data Acquisition (DAQ)
card to the LabVIEW VI. Then the data is processed in LabVIEW and then sent via UDP sockets to the third party VR software. Post processing the data is also logged into an Excel Spreadsheet for later analysis. The VR software is then displayed for the user.

5.2.1 SIGNAL INTERFACE

The VR rehab Signal Interface was used to prototype the third party virtual reality simulation. It is set up to send signals using UDP that are identical to the signals that the actual sensors will send on the device. The UDP code that is used in the signal interface is the basis of the UDP code in the UDP Sub VI (discussed in 5.2.2.8). A screen capture of the signal interface front panel can be seen below in Figure 70.
The bike system includes eight sensors (hardware is discussed in Appendix 1) and all of these sensors are emulated using the signal interface. For more information about the UDP code and how the data is sent to the VR simulation see 5.2.2.8.
5.2.2 VR Rehab Main Interface

The VR Rehab Main Interface (hereafter referred to as simply “interface”) has several components and objectives. The interface is used to acquire all the sensor data from the DAQ card as well as do any signal processing necessary to the data. It is also used to display that data in real-time as well as log the data into two different files for later evaluation. The last objective of the interface is to send modified data to the VR simulation so that is can provide accurate and updated visual feedback to the user. Because of the complicated nature of these objectives it was important to use the Sub VI feature of LabVIEW to simplify and streamline the interface. A Sub VI can be compared to an object in Object Oriented Programming (OOP). These Sub VIs are all inside the main while loop that is executed from when the interface is started until it is stopped by the practitioner. The loop contains a counter that is used in several of the Sub VIs as well as a delay timer to have the loop run at 100Hz under ideal conditions (the loop may be slowed due to computation speed or other factors). Also inside that main while loop are all the controls and displays that the practitioner may use during a session.

The data is acquired using tasks set up to the appropriate lines and ports of the DAQ card. This data is then split and sent to the corresponding Sub VIs for any necessary processing or manipulation (all of the Sub VIs are discussed in the following sections). Then the data is sent through the data logging and communication Sub VIs to be recorded and used in the VR simulation respectively. Below, in Figure 71, is a screen capture of the interface’s front panel.
FIGURE 71: VRHAB MAIN INTERFACE FRONT PANEL

The front panel will be broken up into elements for ease of explanation. The first elements can be seen in Figure 72 below.

FIGURE 72: LOOP AND DATA LOGGING CONTROLS
The arrow above the stop button starts the interface, the grayed out stop sign icon is the emergency stop button and it stops the interface without cleanly finishing the loop. The larger stop button is used to stop the interface between sessions during normal operation. The zero sensors button is used to normalize all the sensor readings to prevent drift as well as look at comparison data. The file name and record button are used for naming and recording the data files (the record buttons functionality is discussed further in 5.2.2.7). The following figure (Figure 73) shows the controls and display of the handlebar information.

The filler bar is used to adjust the sensitivity of the handlebars in the VR simulation. The graph displays how the handlebar data changes over time. The left and right handlebar numeric displays show the real-time readout of the data. The override button is used to null the handlebar signal being sent to the VR simulation. To the right of the handlebar

**Figure 73: Handlebar Display and Controls**

The filler bar is used to adjust the sensitivity of the handlebars in the VR simulation. The graph displays how the handlebar data changes over time. The left and right handlebar numeric displays show the real-time readout of the data. The override button is used to null the handlebar signal being sent to the VR simulation. To the right of the handlebar
display and controls on the front panel is the pedal force display and controls as seen in Figure 74 below.

![Pedal Force Display](image)

**Figure 74: Pedal Force Display and Controls**

The graph shows the change in the pedal force data over time and the two numeric displays beneath the graph show the real-time data. The override button nulls the signal being sent to the VR simulation. Beneath the pedal display and controls is the display for the pedal angle as seen in Figure 75 below.

![Pedal Angle Display](image)

**Figure 75: Pedal Angle Display**
The pedal angle is displayed using two gauges that span from 90° to -90°. There are no controls associated with the angle on the front panel. To the right of the pedal angle display there are the revolutions per minute (RPMs) and velocity controls and display as seen in Figure 76 below.

![Velocity and RPM](image)

**Figure 76: Velocity and RPM Controls and Display**

The graph displays the change in rpm and velocity data over time and the numeric displays beneath the graph show the real time data. The velocity button forces the velocity data being sent to the VR simulation to never drop below 5. The gain is used to increase the velocity for the VR simulation based on the rpm calculation (discussed further in Section 5.2.2.3). To the right of the velocity and rpm controls and display is the heart rate and vibration element controls and display as seen in Figure 77 below.
The heart rate is displayed by a rectangular light that is illuminated in red when a beat is detected. It is also displayed as BPM beneath the light display. The vibration button is used to change control of the vibration elements between manual and simulation control. Manual control is carried out by manipulating the left and right filler bars. Simulation control means that the vibration data is read from the VR simulation via UDP communication. To the right of and beneath the heart rate and vibration controls and display is the minimum and maximum controls and display as seen in Figure 78 below.

The minimum and maximum is displayed for each sensor and can be reset using the reset button. The save button saves the current values into a spreadsheet file.
5.2.2.1 MIN/MAX SUB VI

To calculate the minimum and maximum values of each sensor a Sub VI was created. This is the only Sub VI that is used multiple times in the interface, because of this it is setup to take in any data type and output it’s minimum and maximum. A screen capture of the block diagram of the Sub VI can be seen below in Figure 79.

![Block Diagram](image)

**Figure 79: MIN/MAX SUB VI Block Diagram**

Signal In block is the data that will be calculated. The Reset block is used to reset the minimum and maximum incase of bad readings or multiple trials. Once the data is brought in using the Signal In connection it is then compared to using the greater-than or equal (≥) block for the maximum and the less-than or equal (≤) block for the minimum. A comparison block is then used to evaluate the correct values. The values are then sent out of the Sub VI via the Signal Max and Signal Min connections so that they can be displayed on the front panel or saved to a data file.
The heart rate signal is received from the DAQ card as a Boolean value (true signifying a heartbeat, else the signal is false) because of this several calculations need to be made to the signal before the heart rate can be determined in beats per minute (BPM). A screen capture of the Sub VI can be seen below in Figure 80.

**Figure 80: Heart Rate Sub VI Block Diagram**

The heart rate data comes in through the Data In connection block. A comparison is then made to determine when the signal changes. When the signal changes (from false to true) a counter increments by one and is then sent out of the Sub VI via the Heart Beat Count block. The lower portion of the Sub VI is used to calculate the time between beats so that the BPM can be calculated and displayed. Two timer values are first subtracted (this subtraction calculates the time in milliseconds since the interface was started) and the signal is used to determine the time or two successive beats. This is accomplished by using feedback nodes to store the two values. After the values are found they are
subtracted to get the difference between the beats and this subtraction is used in a division with 60000 to change the units from milliseconds to beats per minute. The heart rate is then sent out of the Sub VI via the Heart Rate connection block for display and data logging.

5.2.2.3 Hall Effect Sensor Sub VIs

The system uses a Hall Effect sensor to calculate the rpm value of the right pedal. The crank of the stationary bicycle is rigidly connected so that the left and right pedals are always 180° apart. Because of this it is only necessary to collect the rpm data of one pedal because the other pedal will have an identical reading. The Hall Effect sensor used in the system is a latching sensor, meaning that the sensor value changes polarity (true or false) when it encounters a magnetic field of satisfactory strength. A Sub VI was created to handle the calculations for the rpm data and the velocity data. The Hall Effect sensor Sub VI block diagram can be seen below in Figure 81.
The signal (true or false) is brought into the Sub VI via the signal connection block. Two timer values must also be brought into the Sub VI as well as an initialized array, the loop count (number of loops performed) and the dimension size that is used to create a running average of the real-time rpm data. The Hall Effect signal is compared against the previous reading to determine when a polarity change happens. The time between changes is then calculated using a method similar to that of the heart rate monitor (see Section 5.2.2.2). Once the time is determined in milliseconds it is used to divide a constant of 15000 (four magnets are installed in the system so the 60000 used for the heart rate calculation needed to be divided by four) to calculate the rpm reading. There
are two other calculations done to the rpm data before it is averaged: erroneous data is filtered and a timeout function.

The rpm data is compared against a known limit of 120 RPMs so that any data exceeding the known ceiling is discarded. The timeout function is setup so that if no signal is read for 1000 milliseconds the rpm data is overridden and set to zero. This calculation could cause problems in the very low rpm range (1-15) but its main use is to lower the averaged value smoothly.

The averaged rpm calculation takes the loop to loop rpm data and places it in an array. The entire array is then averaged to create the running average rpm data. This data lags the loop to loop rpm data but acts as momentum when it is used in the VR simulation. The averaged data is then used to calculate a velocity which does not represent the bicycle’s actual velocity because it is stationary. A Sub VI was created to do the velocity calculations and the block diagram of that Sub VI can be seen in Figure 82 below.

![Diagram](image)

**Figure 82: Velocity Sub VI Block Diagram**

The rpm average data is brought into the Sub VI and is then multiplied by the gain multiplier. The gain multiplier is setup so that a gain value (0-100) is added to the
constant 1 to create a percent multiplier. The velocity is then sent out to be recorded in the data file and is also used in another calculation that can force the velocity to be non-zero. The velocity button adds a constant value, currently set to 5, to the velocity so that even if the user is not pedaling the bike in the VR simulation will continue to move.

5.2.2.4 Handlebar Sub VIs

The handle bar sensors give an analog voltage (ranging from -5 to 5) signal that is read into the VRehab interface for both the right and left handlebars. The handlebar voltage must then be used in a calculation to find the force in pounds. This force is then used to determine a change in direction of the bicycle in the VR simulation. A Sub VI was created to carry out this calculation and the block diagram of that Sub VI can be seen in Figure 83 below.

**Figure 83: Main Handlebar Sub VI Block Diagram**

The handlebar data is calibrated so that there are different calibration constants depending on the sign of the data. The constants were determined experimentally using a load cell to compare known forces against the voltage output of the pressure sensors. So the
handlebar data’s sign is determined using a greater than or equal to zero comparison and that controls which constant is used to find the force. Once that force is calculated the data is sent out of the Sub VI so that is can be recorded and it is also sent to a second Sub VI for more manipulations before it is sent to the VR simulation. The block diagram of this Sub VI can be seen below in Figure 84.

The force data for the handlebars is subtracted and then compared against the dead zone constants. The dead zone constants were determined experimentally by testing the responsiveness of the handlebars. After the dead zone calculation the force data is subject to the same gain calculation that was used in the velocity calculations (see 5.2.2.3). The handlebars also have an override function so that the actual data can be overridden and forced to zero. This is used in the VR simulation to make the bicycle go in a constant direction.

**Figure 84: Handlebar Secondary Calculation Sub VI Block Diagram**
The load cells installed into the left and right pedals produce an analog voltage signal much like the handlebar pressure sensors (see 5.2.2.4). The block diagrams of the Sub VIs that contain the load cell calculations can be seen in Figure 85 and Figure 86 below.

**Figure 85: Main Pedal Load Cell Sub VI Block Diagram**

**Figure 86: Pedal Load Cell Override Sub VI Block Diagram**

In Figure 85 above the data for the load cells is multiplied by a calibration constant in the same way as the handlebar pressure sensor data (see 5.2.2.4). The data is then sent out the Sub VI to be recorded and also sent to the override Sub VI, Figure 86. The override
Sub VI simply uses a Boolean button to force the data to zero before it is sent to the VR simulation.

### 5.2.2.6 Vibration Element Sub VI

To give the user of the system force feedback vibrating elements are installed in the bindings on the pedals. Figure 87, below, is the block diagram of the vibrating element Sub VI.

![Vibration Element Sub VI Block Diagram](image)

**Figure 87: Vibration Element Sub VI Block Diagram**

A pulse-width modulated (PWM) signal is created to control the vibrating elements via a transistor which regulates the voltage going out to the vibrating elements. The values initially come into the Sub VI via either the user interface or from a UDP signal sent from the VR simulation. The remainder of a division of the current loop count and 10 is used to get Boolean values that create the PWM signal. This signal is then packed into an array and sent through the output array to the DAQ card and then eventually to the vibrating elements.
The interface can create two different kinds of data output files: the main data file and the minimum and maximum data file. The Sub VIs that record the data and create the files are discussed in the next sections.

5.2.2.7.1 Main Data Recording Sub VI

The interface front panel has a single button (see Figure 72) that is used to start and stop recording as well as saving the data file (in .dat format). In order to use a single button a finite state machine (FSM) had to be created to implement all the functionality. Figure 88 below is a diagram of the states that are needed and what value from the front panel buttons triggers them.

![Figure 88: Data Recording Finite State Diagram](image)

The front panel button is a Boolean and thus can only have a true and false state. When the button is false the FSM is in its default state (not recording or saving data). When the button is switched to true recording begins and data is saved every loop while the button is true. Once the button is again pressed and turned to false two states happen during the transition: saving the data file and resetting all the values so that the FSM returns to the
default state. The block diagrams of the FSM can be seen below in Figure 89, Figure 90, and Figure 91.

**Figure 89: Default State Block Diagram**
The default state of the FSM initializes all the arrays that hold the data as well as the index that maintains the location of the arrays. The comparison block at the top of the case structure reads the button signal each loop and waits for the button to go from false to true. Once the button goes to true a constant of one is sent to the state change block and forces the FSM into the data recording state, the block diagram of which can be seen below in Figure 90.
Once the FSM changes to state 1, record data, each sensors signal that is sent from its respective Sub VI is collected into an array. The index used for all the arrays is also
incremented each loop while the record data state is active. Once the front panel button is switched to false a constant of 2 is sent to the FSM and it is forces into the third state: file creation and reset. The block diagram of the third state can be seen Figure 91 below.

**Figure 91: File Save and Reset State Block Diagram**
The last two states are merged into one for simplicity. The file is saved by collecting all the sensor data arrays into another array (creating a two dimension array) and then using a create spreadsheet file block that has a file name that is concatenated from the date, time, file type, and users name (which is input from the front panel). After the data is saved all the arrays are reset as well as the array index. The state is also set to zero forcing the FSM back into the default state.

5.2.2.7.2 Minimum and Maximum Data Recording SUB VI

The front panel of the interface displays the minimum and maximum values for each sensor. These values can also be saved into a file. The saving functionality for the file is controlled by a button on the front panel. The block diagram of the Sub VI that writes the file can be seen below in Figure 92.
The data is brought into the Sub VI as bundles with the minimum and maximum data for each sensor together. The data is then unpacked and converted into a string. After the conversion the title of each sensor is added to an array that holds the data. Then all of the data is collected into a large array and then saved using a create spreadsheet file block.
The file name for the spreadsheet file is a concatenation of the date and time as well as
the file type and the user’s name as entered from the interface front panel.

5.2.2.8 UDP Sub VI

Data is sent from the interface to the VR simulation by using UDP communication.
There is no control over the sending of the data, the UDP ports are open once the main
loop is started. The block diagram of the Sub VI can be seen in Figure 93 below.
Figure 93: UDP Communication Sub VI Block Diagram

All of the signals are brought into the Sub VI and then multiplied by the integer multiplier which is used to amplify the signals. Next all the data is converted into strings so that it can be sent to the VR simulation. All of the ports were standardized based on
what was used in the signal interface. When the main loop is stopped all the UDP ports are closed.

5.2.3 VR SIMULATION

The third piece of software that was created for the VRehab system is the VR simulation that was created by a third party developer. The purpose of the VR Simulation is to provide the user of the system with visual feedback on their performance. Before the VR Simulation can begin a series of choices need to be made for the session, these choices are chosen from the VR Simulation Menu. A screen capture of the menu can be seen in Figure 94.

![VR Simulation Menu](image)

**Figure 94: VR Simulation Menu**
The first option available on the menu is to choose the location in the virtual environment that will be the starting point. The default is the beginning of the loop but the simulation can also be started at the checkpoints at are spread throughout the virtual environment. The second option on the menu is to choose the difficulty. This affects the width of the path that the user’s avatar (the user’s representation in the virtual environment is referred to as an avatar) must traverse. The next two options control the distance of the user’s target. This is an identical avatar whose speed is controlled by the user’s heart rate. The last two options allow the practitioner to either start the simulation with these settings or quit the program. Once the session is started the simulation begins, a screen capture of the VR simulation during a session can be seen below in Figure 95.
In the upper right corner of the simulation a map of the virtual environment is displayed. The large red dot on the map is the user’s avatar and the other markers denote the checkpoints on the route through the virtual environment. Below the map the current heart rate of the user is displayed. The sandy tan area of the virtual environment is the path that the user traverses and the human riding a bike model in the middle of the screen is the user’s avatar. Data that is sent from the UDP Sub VI of the interface to the VR Simulation is used to control the user’s avatar. The rpm data controls the avatar’s speed, the handlebar force data controls the avatar’s direction, and the pedal load cells control the tilt of the player.
5.3 Testing of the VR rehab System

Testing of the VR rehab system was done by testing the Handlebar System and Pedal System. This section is segmented into the protocol for the testing, the data collected, and the analysis of the data. Some tests were performed in conjunction with the virtual scene and all tests used the interface software to collect data and configure the system.

5.3.1 Testing Protocol

The testing protocol is different for the Pedal and Handlebar System. The Handlebar System offers more possibilities for testing because it is used more like an input device. The Pedal System protocol is more straightforward but still provides useful data for analysis.

5.3.1.1 Handlebar System Testing Protocol

A series of tests were conducted to validate the design of the Handlebar System and also evaluate its capabilities as an input device for the virtual environment.

- Grip Oscillations from Steady Pedaling: During normal pedaling motion oscillating loading patterns in the isokinetic grasping forces occurring from trunk rotation and flexion/extension of the legs have been observed [31]. To assess symmetry and loading between the two handles, force patterns were recorded and compared for left & right hands. The subject pedaled without viewing the VI or simulation and pedal as symmetrically as possible. Three 1-minute trials were conducted with a 1 minute break between trials.
• Navigation in a Virtual Environment: Within the simulation the subject pedaled and navigated the centerline of the track. Checkpoint 2 was selected as the starting point for the simulation because it includes the most turns immediately following the start. The subject navigated for 1 minute, was allowed to rest for 1 minute, then the simulation was reset for a new trial. Three trials were recorded.

• Linear Increase in Isokinetic Grip: While sitting in the system but without pedaling, the subject grasped the handles without viewing the VI or simulation. Over a 30-second period they linearly increased their isokinetic grasp on both handles up to 70% of their maximum comfortable force. Subjects were allowed to take three unrecorded practices beforehand. Then three 30-second trials were conducted with a 1 minute break between trials.

5.3.1.2 Pedal System Testing Protocol

A test was conducted to examine the loading pattern of the left and right feet during steady pedaling.

• Steady Symmetrical Pedaling: During normal pedaling motion characteristics oscillating loading patterns have been observed [31, 32] [31][30][29] to assess symmetry and loading between the two pedals, force patterns were recorded and compared for left & right feet. A healthy subject pedaled as symmetrically as possible without viewing the VI or simulation.
5.3.2 VR rehab testing data

This section is separated into sub sections based upon the test that was performed and the data was gathered.

5.3.2.1 Handlebar system data

All data for the following three tests has been collected via attachment to a recumbent stationary bicycle. Data was collected from a subject who was a right-hand dominant healthy adult male with no previous cognitive or physical impairments. It is important to note that the stationary bike used for this testing is fitted with a friction brake which cannot be disengaged, but was kept consistent for friction forces.

5.3.2.1.1 Grip oscillations from steady pedaling:

Oscillating force readings on the handles were anticipated. The frequencies of the left and right handle forces were expected to be the same and that frequency should correlate to the pedals’ rpm. Figure 96 below displays the revolutions per minute (rpm) of the crank along with the handle forces. The results from Trial 2 are shown below, and trends are representative of all three trials in this test. The data has been sorted to remove the ramp-up period data for the pedaling motion. RPM for this test was steady averaging 93 rpm.
As seen in Figure 97 below, amplitudes of peaks for left and right hands are close but have consistent differences. The left hand peaks (compressive forces) are consistently lower by roughly 20% but troughs (tensile forces) are comparable for both.
The characteristic shapes of both red and blue curves have clear increases and slower decreases in force for each period. Positive values indicate an upward ‘pulling’ force on the underside of the handle. Negative values indicate a downward ‘pushing’ force on the topside of the handle. Loading patterns for left and right handles have been isolated below.

**Figure 98: Graph of Right Hand Force During Steady Pedaling**

**Figure 99: Graph of Left Hand Force During Steady Pedaling**

For all three trials the loading patterns differed from left to right hands. But the trends for each hand were consistent from one trial to the next.
5.3.2.1.2 Linear Increase in Isokinetic Grip:
The results were anticipated to be a roughly linear increase from zero to a force 70% MCL, then a sharp drop off. Below are averaged results for the three loading trials, normalized and aligned at the peaks for the 30 second cycle.

Figure 100: Graph of 0-70% MCL for Grasping During Rest Averaged

Peak forces perceived at 70% MCL were within 2.8% of each other (18 lbs, 80 N) for both hands. The left hand exhibited a smoother transition to peak and smoother drop (unloading) to zero. Right hand had a sharper increase, decrease. Both curves have a 2.5 sec resting value at the end.
A healthy adult female subject pedaled in steady motion at 38 rpm for the 4 minute duration of data collection.

Figure 101 below is a close up view of left & right pedal forces during symmetrical pedaling. Peak loadings are ½ period apart for the 180° offset between the crank arms. Positive values indicate compression forces, negative values indicate tensile forces. Peak compression force magnitudes were smaller for the left foot and maximum tensile forces were greater for the left foot than the right.

**Figure 101: Graph of Force During 5 Seconds of Steady Symmetrical Pedaling**

Normalized curve samples are shown below for one rotation of both pedals.
Curves were normalized at zero force point as crank TDC. The amplitude of forces for both pedals is 22.5 lbs for compression, with greater tensile forces exhibited by the left pedal. The analysis of the pedal data is in the following section.

5.3.3 Analysis and Discussion

The section again is separated by the system that was tested. The Handlebar System analysis includes the testing during steady pedaling and the linear increase of force on the system. The Pedal System analysis includes testing from the steady symmetrical pedaling.
5.3.3.1 Handlebar System Testing Analysis and Discussion

The Handle System testing discussion has two sections: grip oscillations from steady pedaling and linear increase in force.

5.3.3.1.1 Grip Oscillations from Steady Pedaling Analysis and Discussion

Oscillating force readings on handles were recorded. Figure 99 displays the revolutions per minute (rpm) along with the handle forces. RPM for this test was steady averaging 93 rpm which is within range for a healthy recommended cadence (88-95rpm)[31].

The amplitude of the forces could be an indication of asymmetry in the pedaling forces transmitted through the trunk. Abnormalities in pedaling motion can affect the loading pattern on the handles.

Examining the curves in Figure 97; the right hand lifts (pulls) up harder and faster than the left (indicated by the shorter rise time to force peaks), then drops abruptly (250ms) and decreases further for a larger downward pushing force. The pattern of the right hand is 250ms of peak & drop, then 250 ms of slow change in force from pulling to pushing. The left hand had an equal period to the right, but different shape and division between these actions. In the left hand each large loading is followed by a second faster loading of almost equal amplitude. There is a 370 millisecond (ms) primary load and unload action (slower than the right hand) and a 100 ms secondary load & unload action ending with net zero force on the handle and moving into the ‘pushing’ phase. Results from studies in strength pedaling have shown to stabilize trunk postures despite deficits in the lower extremities [33].
All three trials had loading patterns which differed from left to right, but patterns for left and patterns for right were consistent between trials for magnitude, and shape, and period.

5.3.3.1.2 Linear Increase in Isokinetic Grip Analysis and Discussion
This test evaluates the feeling of symmetry and comfortable loading between two hands and as an input device for zero to 70% of MCL.

The left hand in Figure 100 had a smoother transition to peak and smoother drop down for unloading than the right. The right hand had a sharper increase and decrease very close peak forces. Both curves have a 2.5 second resting period at the end. The unloading patterns were almost identical for both handles. The dip at the peak force for the right hand could indicate that subject needed to adjust their grip. This coincides with studies relating comfort and maximum force output from high loading rate [34].

It is important to note that since there is uneven surface area for the two hand surfaces in contact with the sensing area, some of the force from the higher-loaded lateral side will be counteracted by the pressure on the smaller medial side.

5.3.3.2 Pedal System Testing Analysis and Discussion
The pedal data has been aligned according to the same zero loading point for pedal TDC (indicated by 330° CW from TDC). When evaluating the data for pedal forces it is important to consider that since the pedals are connected rigidly through the crank, any deficits or impairments of a weaker leg can be overcome by the stronger leg. The higher
compression forces from the right leg may not just be higher strength from the right leg but may be attributed to a deficit in the left leg (also referred to as ‘negative work’) [33].

Maximum compressive forces took place at 36° and 90° CW from TDC for the left and right feet, respectively. The right foot exhibits peak compressive forces at the appropriate rotation angle but the left foot reaches peak compression sooner by almost 30% of the cadence cycle [31, 33]. Peak tensile forces were greater for the left foot than the right by 5.5 lbs and took place 15% later on in the pedal cycle. The locations of these peak tensile forces are at 240° and 294° CW from TDC for the right and left feet, respectively.

The loading pattern for both feet also displays an unusual secondary rise in compression at BDC (180°) which differ between each other by 11% magnitude and 4% cadence timing. This pattern could possibly be attributed to the stick-friction from the stationary bike when the normal forces drop to zero and tangential forces are close to zero.

5.4 VRRehab Future Work

There is possible future work for the VRRehab software in the areas of software communication, interface design, and sensors for the Pedal System. Future work for the VR software is not discussed here as it was developed by a third party.

Currently the software communication is handled by UDP in both LabVIEW and the VR software. The second version of the VRRehab system would include changing moving the data logging and acquisition from LabVIEW to a custom program. A DAQ card will still be used but the software can collect data directly from the device instead of using
LabVIEW. This will reduce the cost of the software and also simplify communication and display of data and controls for the system.

In the current version of the system there is an interface for the practitioner and the VR software to give the user feedback. Because there are two separate software programs running during a session it is cumbersome and counter-intuitive to switch between them to make necessary adjustments to the system and see the data in real time. A future version of the software would have integration of the controls the practitioner needs with the VR software so that there is no need to switch between programs.

The current version of the pedal uses an accelerometer and Hall Effect sensor to get rpm and tilt data. In a future version of the Pedal System the rpm and pedal data would be collected by using an infrared (IR) sensor system. By placing reflectors on the point of rotation of the pedal and front and back an IR receiver places in line of sight of the reflectors would be able to collect the data needed to calculate the tilt and RPMs. This is advantageous because it not only reduces the number of sensors needed for the system but it would also produce higher accuracy data.

5.5 VRRehab System Summary

In this project a prototype VR rehabilitation system was created by mounting custom hardware that controls a VR simulation onto a standard stationary recumbent bicycle. The hardware consisted of two identical handlebar assemblies, two pedal assemblies, a heart rate monitor, power supply, data acquisition system, and supporting electronics.
The software for the VR rehabilitation system includes a signal interface, a main interface, and a VR simulation.

The hardware has sensors that transmit data to the main interface. The main interface then conditions the data (as well as saving it to an external file) and sends it via UDP communication to the VR simulation. The Handlebar Systems measure the gripping force the user exerts and this measurement is used to control the user’s avatar in the VR simulation. The Pedal Systems have sensors to measure pedal tilt, force, and crank RPMs. The pedal force is used to modify the user’s avatar’s vertical orientation. The crank RPM measurement is used to control the velocity of the avatar in the VR simulation. The heart rate of the user as collected by the heart rate monitor is used in the VR simulation to control the user’s target avatar.

Testing of the system was performed by on both the Handlebar System and the Pedal System. The testing successfully showed the accuracy of the systems as well as the issues with symmetrical pedaling.

The areas of possible future work in the VR Rehab system include changing several of the sensor technologies as well as more integration between the interface software, the VR software, and the hardware. Overall the first prototype system has proven to be a successful proof of concept Bicycle Rehabilitation System.
CHAPTER 6: CONCLUSIONS

This thesis presented work completed on three projects: design of software for a Multiple User Virtual Environment for Rehabilitation (MUVER), design of software for the Active Hand Rehabilitation Interface (AHRI), and design of interface software for the VRehab Bicycle Rehabilitation System. In all of these projects a virtual reality (VR) system was described for use in rehabilitation.

Background on other VR systems and their uses as well as studies completed with them was presented. The overall conclusion is that the use of advanced input devices and virtual environments can be successfully used to augment motor rehabilitation for patients post stroke.

The hardware chosen for the MUVER project was the P5 Glove device. It was chosen because it is cheap and easy to interface with VR software. The initial prototype virtual scene for the MUVER concept was created in the Blender game engine but a final prototype scene was created using the Panda 3D game engine. Once completed testing was performed using six subjects. The results of the testing showed positive evidence that the MUVER concept can help with rehabilitation. Testing of the P5 Glove was performed in order to analyze the accuracy of the sensor data. It was found that while the finger bend data had satisfactory accuracy the position data was very inaccurate. This inaccuracy could be due to the sensors themselves or the placement of the infrared tower. Future work for the MUVER project includes implementing more features into system as well as creating a custom input device that will correct the deficiencies of the P5 Glove.
The ARHI uses novel custom hardware designed by members of the Northeastern University Biomedical Mechatronics Laboratory. The device has two degrees of freedom for hand and wrist movement. It actuates this movement by using a linear and a rotary hydraulic electro rheological fluid based actuator. Closed loop control was implemented on the hardware to increase performance and haptic feedback for the user. The virtual environment for the ARHI was created using Panda 3D. Information was sent from the custom to the virtual environment by using user datagram protocol (UDP) code in both the control program (created using LabVIEW) and the Panda 3D virtual environment. A system was created in the virtual environment to allow for quick setup and implementation of custom mazes created by practitioners. Testing was performed on a simple maze by three subjects. By analyzing the total time as well as average force and torque it was shown that the ARHI shows promise as a VR system. Future work for the project’s software includes creating more mazes to provide users with a steady complexity and more items for use in the level to keep the user interested and motivated.

The last project presented was the VRehab Bicycle system. Custom hardware systems were created for this VR system including a handlebar and pedal sensor package. The hardware and VR software for this project was complete by Richard Ranky and a third part developer respectively. In order to ensure proper data flow and communication between the sensors and the VR software a LabVIEW based interface software was created. The interface software was able to acquire data and then perform calculations on it before either logging it to a separate file or sending it via UDP. Testing was performed on both the Handlebar and Pedal Systems. This testing showed that the hardware design
provided accurate data that was then conditioned and sent to the VR software via the interface software. Future work for the interface software includes adding better data logging and communication algorithms as well as simplifying hardware systems to lower the amount of needed sensors.

The three projects presented all show promise as VR systems to assist with rehabilitation. All projects require future work and testing to move beyond the proof of concept phase. However, the data collected so far has been invaluable as an aid in design for these complex and technologically advanced systems.
REFERENCES


