

Is Finger Motion Tracking a Superior Control Method for Virtual Reality?



UNIVERSITY OF LINCOLN

Mr Liam Wilson

WIL13458211

BSc (Hons) Games Computing

Project Supervisor: Dr. John Murray

University of Lincoln, School of Computer Science

11th April, 2017

Acknowledgements

I would like to express my gratitude towards the following individuals, without whom completing this project would not have been possible.

- **Dr John C. Murray** – for continuous engagement and encouragement throughout the duration of the project, unparalleled availability for meetings and discussion, and help with regards to resources and guidance. The support was astounding and greatly appreciated.
- **Mr Lee Wilson** – for his continuous encouragement and motivation both prior to and during study, helping me to achieve my potential, and supporting me in all my previous accomplishments.
- **Ms Natalia Cossey** – for providing constant motivation and support throughout the duration of the project, allowing me to persevere through the hardships encountered during the development and writing process.

I would also like to thank the following individuals for their friendship, support and inspiration during my time at the University of Lincoln. You provided me with some of my most memorable experiences and I am extremely grateful for all you have helped me to accomplish. I wish you all the best of luck in your future endeavours:

Luke Exton
Adam Plunkett
Andrew Cardwell
George Learmonth

Moad Zardab
James Allington-Kay
Marlon Gilliam
Raymond Tunstill

Lastly, I would like to thank every participant involved in testing the artefact – for the phenomenal feedback which formed a basis for conclusion. Whilst their identities remain anonymous, they should know that their participation is greatly appreciated.

Abstract

Virtual Reality (VR) is a technology that is becoming increasingly popular amongst developers, seeing use across a vastly diverse range of fields, ranging from gaming to medical training. However, despite this diverse usage, the core elements of any VR experience remain persistent, primarily immersion and interaction are essential to the overall effectiveness of a VR application. This project aims to investigate whether enhanced VR controller devices, primarily finger motion tracking devices, prove beneficial to immersion and interaction when experiencing VR.

This investigation and report provides insight into design considerations, including the decision to use a Data Glove rather than infrared means to track hand and finger motion. Furthermore, the creation and implementation of Data Glove hardware and software is outlined, to provide clarity regarding functionality and future works.

This project provides an interesting insight into methods for testing immersion and interaction with devices such as the Data Glove, and concluded that whilst the quality of interaction is not effected by finger motion tracking, the experience immersion is greatly increased.

Contents

1 Introduction.....	7
1.1 Background and Rationale	7
1.2 Aims and Objectives	9
2 Literature Review	10
3 Development Life Cycle	15
3.1 Project Management	15
3.1.1 Methodology	15
3.1.2 Time Management.....	18
3.1.3 Risk Management	20
3.2 Hardware Development	22
3.2.1 Foreword.....	22
3.2.2 Iteration 1 – Component Testing (Arduino Uno + Breadboard)	22
3.2.3 Iteration 2 – Prototype (Arduino Uno)	24
3.2.4 Iteration 3 – Practical Improvements (Arduino Pro Mini)	27
3.2.5 Iteration 4 – Ergonomic Improvements	30
3.2.6 Further Maintenance.....	33
3.3 Software Development Tools.....	34
3.3.1 Integrated Development Environments (IDEs)	34
3.4 Interfacing Software Development	35
3.4.1 Iteration 1 – Arduino Software.....	35
3.4.2 Iteration 2 – Initial C# System	37
3.4.3 Iteration 3 – C# Refactoring.....	40
3.4.4 Testing.....	42
3.5 Scenario Software Development.....	43
3.5.1 Iteration 1 – Initial Scene	43
3.5.2 Iteration 2 – Real-World Scaling.....	44
3.5.3 Further Maintenance.....	45
3.6 Research	46
3.6.1 Participant Recruitment.....	46
3.6.2 Ethical Procedures	47
3.6.3 Design.....	47
3.6.4 Study Procedure.....	48
3.6.5 Results Analysis	48
4 Reflective Analysis	52
4.1 Critical Reflection	52
5 References	53
Appendix A – Project Management	58
Appendix B – Research Resources	64

Table of Figures

Figure 1.1 – Manus VR’s gloves, accompanied by the HTC Vive (Road to VR, 2016).....	8
Figure 2.1 - Illustrated image of tracking components used in the Oculus Touch controller (Game Spot, 2015).....	11
Figure 2.2 – A breakdown of the six degrees of freedom (Leading Ones, 2016).....	13
Figure 3.1 - Illustration of Rapid Application Development (TatvaSoft, 2015).....	17
Figure 3.2 – Iteration plan for the artefact development life cycle.....	18
Figure 3.3 – Differences between planned and actual IE implementation. Snippet of full project Gantt chart (Figure A.1) with timestamp references to GitHub commits (Figure A.2).....	19
Figure 3.4 – Flex Sensor circuitry diagram (Spectra Symbol, 2014).	22
Figure 3.5 – The overall circuitry for the Data Glove, whereby the leftmost diagram shows the circuitry for each flex sensor. Symbols are formatted using IEC-60617 specifications (Autodesk, 2016).....	23
Figure 3.6 – Test implementation using the Arduino Uno. Colour code: Red = Live (5V), Green = GND, and Blue = Output.	23
Figure 3.7 – Adapter cable, designed for use with the Arduino Uno.....	24
Figure 3.8 – Data Glove design (Phase 1).	24
Figure 3.9 – Two sensor pocket implementations, B represents the more favourable design, chosen over A.	25
Figure 3.10 – Electro-static sleeves, created using re-cycled anti-static shielding and electrical tape.	25
Figure 3.11 – Final pocket design, shown on both left (blue) and right (orange) gloves. The design is pictured without anti-static shielding.....	25
Figure 3.12 – Prototyping flex sensor implementation, using copper wiring and header pins.	26
Figure 3.13 – Final prototyping solution, using the Arduino Uno.	26
Figure 3.14 – Data Glove design (Phase 2).	27
Figure 3.15 – Old and new colour coding specification for electrical wires in the UK.	28
Figure 3.16 – Artefact implementation using the Arduino Pro Mini and silicon wiring. The wires no longer use pin-based connections and are instead soldered to the board.....	28
Figure 3.17 – Finished electronics for the artefact, using four flex sensors. Heat shrinks were re-added for additional organisation, safety and support.	29
Figure 3.18 – Data Glove design (Phase 3).	30
Figure 3.19 – 3D model of the Arduino case, created in Blender.	31
Figure 3.20 – 3D-printed box to contain the Arduino Pro Mini.	31
Figure 3.21 – Velcro straps attached to the HTC Vive controller.....	32
Figure 3.22 – Data Glove without electrical components, detailing the addition of more supporting loops and a Velcro square for attaching and detaching the 3D-printed box.	32
Figure 3.23 – Finished hardware artefact, shown with the HTC Vive controller affixed to the users arm.	33
Figure 3.24 – Pseudo code for the Arduino interfacing solution.....	35
Figure 3.25 – Code written in C for creating a connection between the Arduino and a computer. This code is store and executed on the Arduino board.	36
Figure 3.26 – Class diagram representing the association between classes for this interfacing solution.....	37

Figure 3.27 – Flowchart detailing the execution of software interfacing code.	38
Figure 3.28 – Code snippet for unsigned integer flags within the ResistorBounds class.	38
Figure 3.29 – Code snippet taken from source, showing the comparison function, array accessor functions and copy constructor override for the ValueArray class.	39
Figure 3.30 – Finalised class structure for the software interface.	40
Figure 3.31 – Function created for handling errors when opening a serial connection.	41
Figure 3.32 – 3D model of a human hand, built and rigged within Blender.	41
Figure 3.33 – Custom Editor used for calibrating the Data Glove within the Unity editor.	42
Figure 3.34 – A VR room designed for the participant testing process.	43
Figure 3.35 – The objects used in research testing to assess differences in interaction between the HTC Vive and Data Glove.	44
Figure 3.36 – The finalised scene to be used in research testing.	45
Figure 3.37 – The use of masking tape during the user testing process.	46
Figure 3.38 – Results regard the difficulty when interacting with the Data Glove. Mean: 3.15.	49
Figure 3.39 – Results assessing whether real world objects improve interaction. Mean: 2.5.	49
Figure 3.40 – Results of Data Glove immersion compared to the HTC Vive controller. Mean: 4.	50
Figure 3.41 – The resultant impact of real world objects on immersion. Mean: 2.75.	50
Figure 3.42 – Whether the physical presence of the glove detracted from the experience. Mean: 2.1.	51
Figure A.1 – Original project Gantt chart.	58
Figure A.2 – Log of GitHub commits.	61
Figure A.3 – Graph of GitHub contribution frequency from 08/01 to 27/04.	63
Figure A.4 – Graphs of GitHub additions and deletions from 08/01 to 27/04.	63
Figure B.1 – Scanned evidence of consent forms being used.	64
Figure B.2 – Research questionnaire.	65

1 Introduction

1.1 Background and Rationale

Virtual Reality (VR) is a technology that is becoming increasingly popular amongst developers, with use across a vast scope of widely diverse fields ranging from gaming, for example Space Pirate Trainer (I-Illusions, 2016), to medical, such as helping “phobic patients overcome their irrational fear of spiders [and] post-traumatic stress disorder” (Hoffman, 2004). Current VR systems provide an exceptional level of immersion to users that had previously been unachievable due to the available technology of the time. However, technological advancements have now made it a perfect platform for training, simulation and providing an overall better user experience. Despite this, there are several areas for improvement when considering VR systems; motion tracking is an example of this. This project will investigate motion tracking for fingers, focusing on how it can further enhance interaction and control for VR systems, particularly concerning the HTC Vive.

Current VR controllers are limited in their dexterity and functionality, typically only tracking hand movement and button presses, which creates difficulty when attempting to simulate natural and precise interaction with an environment. One example of this is the HTC Vive controller, which simply keeps track of location and orientation, in addition to having several buttons located around the device. These features allow the device to point and interact with objects despite the unnatural feeling experienced by the user. Many existing VR control systems are designed to be held, giving the sensation of holding a rod or a gun. Consequently, developers are often limited both in design and implementation of VR software. This project aims to relinquish these issues and provide developers with the opportunity to develop software and games based around natural human movement. Additionally, the project will consider ease of use, comparing the relatively basic existing technology to a more complex and natural system.

As previously mentioned, VR is applied across many areas ranging from leisurely activities, such as gaming, to far more critical fields, for example medical and crime investigation. Whilst a selection of these activities may benefit from the sensation of holding an object, as provided by existing technology, many may benefit from, or require, more precise and dynamic interaction. For example, consider a VR desktop, finer finger motion tracking would be required when using a virtual mouse or keyboard. In addition to this, tracking complete hand movements has the potential to enhance the immersive experience by allowing users to interact with their environment using natural motions. A further advantage of this is that users already understand how to interact with real-world objects; therefore, minimal guidance would be required to ensure that the user knows how to control the software. Another advantage of a VR glove is the opportunity to elicit accurate haptic feedback, using gentle vibrations throughout the glove to provide the illusion of touch. A final benefit is improved gesture control, providing the potential to enhance interaction with both players and NPCs within VR. Developers could track the position and bend factor of each individual finger, something that is not possible in standard controllers, allowing the user to communicate through a vast number of gestures.

As mentioned, there are many existing VR controllers, although many of them do not address the issue of finger motion tracking. However, some existing technology has attempted to tackle this issue. One such example is the Touch controller, released by Oculus to accompany their existing Rift headset. The Touch controller provides a basic method of tracking finger movement although it does not precisely track motions, instead making assumptions based upon whether certain buttons, located around the grip and top of the controller, are pressed. The Touch controller shares many similarities with the PlayStation Move controller, which also makes assumptions based on whether a button, such as a trigger, is pressed. A more advanced attempt at finger motion tracking takes the form of Leap Motion, which makes use of a specialized infrared sensor to track joint location and orientation. This is one of the best commercially available devices specifically for tracking finer hand movements but has a few issues. Firstly, this technology does not make use of a controller, removing the possibility of implementing features such as haptic feedback. Additionally, relying upon a sensor means that occlusion can occur, the process whereby obstructions cause loss of tracking. Manus VR tackles the issue of infrared sensors and is perhaps the most notable development made towards hand and finger motion tracking. This controller holds particular interest because many of its features will be utilised for this project, and adapted accordingly. Manus VR is a glove-based motion-tracking device, seen in Figure 1.1, which monitors the bend factor for each of the wearers' fingers to manipulate a virtual hand model. Natural interaction occurs because of this functionality. Like Manus VR, the artefact produced for this project will also take the form of a glove, rigged with flex sensors to monitor finger movement.



Figure 1.1 – Manus VR's gloves, accompanied by the HTC Vive (Road to VR, 2016).

1.2 Aims and Objectives

The aim of this project is to investigate the use of finger motion tracking as a means for improving control and interaction in VR systems. To achieve this, a variety of objectives will be completed, as outlined below:

- **Design an artefact, in the form of a wearable glove.**
 - The artefact design will enable accurate tracking of individual fingers.
 - The design will explore several options for measuring finger motion.
- **Create the physical artefact ensuring that digital parsing of raw data can occur.**
 - Initially, the artefact will provide electrical signals. This must undergo Analogue to Digital Conversion (ADC), allowing the computer to read the data.
 - The artefact must function as expected at a physical level to minimize the possibility of hardware related issues when interfacing at a software level.
 - Explore the implementation of haptic feedback within the glove.
- **Interface the artefact with the *Unity Game Engine (Unity Technologies, 2004)*.**
 - Produce a 3D model of a hand using Blender (Blender Foundation, 2002) and port the model into Unity, ensuring that individual manipulation of joints can occur.
 - Accurately map real world movements onto the 3D model within Unity.
 - Implement a method of calibrating the artefact within Unity to enhance the overall accuracy and precision for differing hand dimensions.
- **Create an experimental setup, within Unity, for testing.**
 - Design an interactive scenario capable of evaluating the gloves effectiveness when compared to standard controllers.
 - Implement the scenario within the Unity Game Engine, allowing fair evaluation of the HTC Vive controller against the artefact by enabling control scheme switching.

2 Literature Review

Existing VR controllers pose a variety of limitations which this project aims to explore. The HTC Vive controllers are an exemplary example of this which are only able to “keep track of location [and] track their orientation and the direction in which they are pointing” (Benson et al., 2015). These un-intuitive motion controls cannot sufficiently create precise interaction; despite being combined with an array of buttons spanning the controller chassis. A variety of systems have been developed to reduce these limitations, each differing in their approach to the problem but all aspiring for the same goal of finger motion tracking. Many of these systems are “implemented by applying computer vision techniques to observe and track finger and hand motion” (Cohen and Corradini, 2002) or additionally make use of wearable technology such as the “contact based 5DT Data Glove” (Arkenbout et al., 2015). It is these technologies which inspire and motivate this project, whereby the goal is to assess the feasibility and effectiveness of such devices in VR systems. This section aims to analyse existing research and draw informed conclusions on a variety of aspects associated with the project.

Benson et al. (2015) discuss Hikari Hook, a game developed for the Oculus Rift, a VR platform which, prior to the release of the Oculus Touch controller, lacked any form of hand motion tracking. Despite this, the game was developed based on an understanding that “motion controls add an extra layer of player interaction and immersion” (Benson et al., 2015), thus the team utilised existing motion tracking devices to produce the desired result. Hikari Hook aimed to tackle a key issue for many VR platforms regarding a player’s ability to move freely within the virtual world. The idea involved traversing through a level using a grappling hook, allowing the player to navigate great distances in the virtual world whilst remaining stationary within the physical play space. The research conducted by Benson et al. concluded that a significant number of participants involved with Hikari Hook were unable to play for extended periods of time, expressing symptoms associated with nausea regarding the sudden movements whilst using the grappling hook. This suggests that movements within the virtual world which are not accurately mapped to the real world will cause the player discomfort, and potentially lead to feelings of sickness and nausea. An experiment conducted by Allison et al. (2001) which simulated temporal delay, referred to as lag, within a virtual application suggests a probable cause for the discomfort. The experiment concluded that head tracker induced lag can cause simulator sickness, a form of motion sickness. Furthermore, the paper highlights a popular theory which explains that motion sickness is a product of sensory conflict between visually and vestibular sensed motion (Oman, 1990; Slater et al., 1995; Howarth and Costello, 1997). Thus, it is not unusual that a user would feel discomfort and nausea when visual motion, observed within a Head Mounted Display (HMD), is disjointed from their physical movement. Slater et al. (1995) refers to this phenomenon as “vection”, an illusion of self-motion which describes when a large part of a visual field shifts and gives the impression that the viewer has moved, despite remaining stationary. Furthermore, the paper suggests a correlation between simulator sickness and presence, explaining that an increased sense of presence means that the user pays more attention to the discrepancies between visual and vestibular systems (Slater et al., 1995). This research highlights the importance of a visual and physical union when considering motion within VR, whilst also suggesting that an increased sense of presence can cause issues regarding simulator sickness if the union is broken. Consequently, a conclusion can be made that an implemented finger tracking method must be robust, ensuring minimal possibility for the device to fail which would cause the union to be broken.

Ripton and Prasuehsut (2015) provide an interesting comparison of existing VR systems, analysing four of the most advanced, robust and promising products available to consumers. The comparison first introduces the HTC Vive, a system developed by Valve in collaboration with mobile company, HTC. The headset utilises high end hardware to provide a clearer image than its competitors and allows for 360-degree movement within a VR landscape (Ripton and Prasuehsut, 2015) due to the use of two Infrared sensors for tracking, as opposed to the its closest PC competitor, the Oculus Rift, which uses only one. In addition to this, the HTC Vive system includes two controllers in the form of handheld “wands”, something the Oculus Rift lacked until recently introducing the Touch controllers. These controllers provide a vastly enhanced interactive experience when compared with other systems which utilise only a head-mounted display (HMD). Ripton and Prasuehsut provide evidence indicating that the HTC Vive, whilst being very expensive, provides a VR experience which is only challenged by Project Morpheus, a PlayStation exclusive platform. This leads to the conclusion that the HTC Vive is the best system for this project, whereby the “wands” can be utilised to track the users hand motion by affixing the controllers to the wrist. Fittkau et al. (2015) further support this conclusion, suggesting that the HTC Vive be a candidate for future studies due to improved display resolution and better user experience when compared to the Oculus Rift. However, both this paper and the previous article were written in 2015, prior to the release of the Oculus Touch controllers which perform not only the tracking functionality of the HTC Vive controllers but also boast simple finger motion capture (Amir et al., 2016). Despite this, the Oculus Touch’s improved functionality offers no benefit to this project as the controller simply utilises buttons, as shown in Figure 2.1, to provide basic feedback of whether the users hand is open or closed. This therefore does not satisfy the requirements of this project and thus the HTC Vive’s improved motion tracking capabilities, provided by two sensors rather than one, is more beneficial to the project.



Figure 2.1 - Illustrated image of tracking components used in the Oculus Touch controller (Game Spot, 2015).

Dorfmüller-Ulhaas and Schmalsteig (2001) present a finger tracking solution for virtual applications which supports three-dimensional input, with high precision, using vision-based tracking methods. The study provides a comprehensive analysis of occlusion and self-occlusion, a process that occurs when line of sight is broken between a focus object, such as a hand or controller, and a sensor, such as an infrared camera. This phenomenon has been identified as the most significant issue when considering computer vision-based motion tracking solutions (Kato et al., 2000, Dorfmüller-Ulhaas and Schmalsteig, 2001; Brown et al., 2016). The study suggests the use of retro-reflective markers as a solution to the issue, however concludes that “multiple cameras will be necessary to solve hand-hand occlusions” (Dorfmüller-Ulhaas and Schmalsteig, 2001). The HTC Vive puts this theory into practice, whereby two sensors are utilised at opposing corners of the play space to avoid occlusion of the controllers or HMD by the user. Other attempts to resolve this issue include wrist-worn devices such as Digits, a system designed by Kim et al. (2012). Digits is described as a sensor that recovers the full 3D pose of the user’s hand, enabling a variety of freehand interactions (Kim et al., 2012). Interestingly, it does not require the use of retro-reflective markers, as seen in the previous solution proposed by Dorfmüller-Ulhaas and Schmalsteig. However, the study of Digits concluded that occlusion was still an issue in several failure cases, expressing that “occlusions resulting from crossed fingers, overly bent thumb and handheld objects are problematic for hand pose reconstruction” (Kim et al., 2012). Each of these studies further highlight the lack of robustness in vision-based motion tracking solutions, making it a flawed technique when considered for this project. However, Arkenbout et al. (2015) suggest that a union between contact-based and vision-based sensory input devices provides data completeness during situations where both partial and full visual occlusion occurs. The solution provided by Arkenbout et al. describes a system which fuses data gathered from a Kinect camera and contact-based 5DT Data Glove to achieve astonishing levels of dexterity whilst also solving issues related to occlusion. Interestingly, the paper mentions that the artefact “improved hand posture ... especially at the occurrence of visual self-occlusion of the fingers” (Arkenbout et al., 2015), indicating that the contact-based data glove produced better results in the absence of vision-based data. Furthermore, sensor-based gloves are not susceptible to self-occlusion and have no dependence on lighting conditions (Zubrycki and Granosik, 2014), as they don’t make use of infrared cameras, instead relying exclusively upon in-built electronic sensors. A conclusion can be made regarding this research which suggests that data gloves are the better option when trying to achieve a robust solution to finger motion tracking.

Unfortunately, there is research which indicates that vision-based solutions are far superior, in terms of accuracy, when compared to data gloves. For example, Zubrycki and Granosik (2014) mention that flexion sensors, the primary source of data for sensor-based gloves, are likely to change their parameters over time and are susceptible to a 30% resistance difference across different batches. Furthermore, a human hand is considered to have more than 20 degrees of freedom (Sturman, 1991; Erol et al., 2007), whilst a single flex sensor can only track orientation in one dimension. A data glove would therefore require several sensors on each finger to contend with the motion tracking ability of vision-based solutions, which can easily track upwards of six-degrees of freedom, as seen in Figure 2.2. Whilst calibration before use can solve issues regarding difference in resistance and parameters (Zubrycki and Granosik, 2014), most data glove implementations simply cannot contend with vision-based solutions in providing accurate motion tracking. Despite this, the decision to use data gloves is a consequence of previous statements regarding robustness of motion tracking devices as being of utmost importance when wishing to preserve a sense of presence.

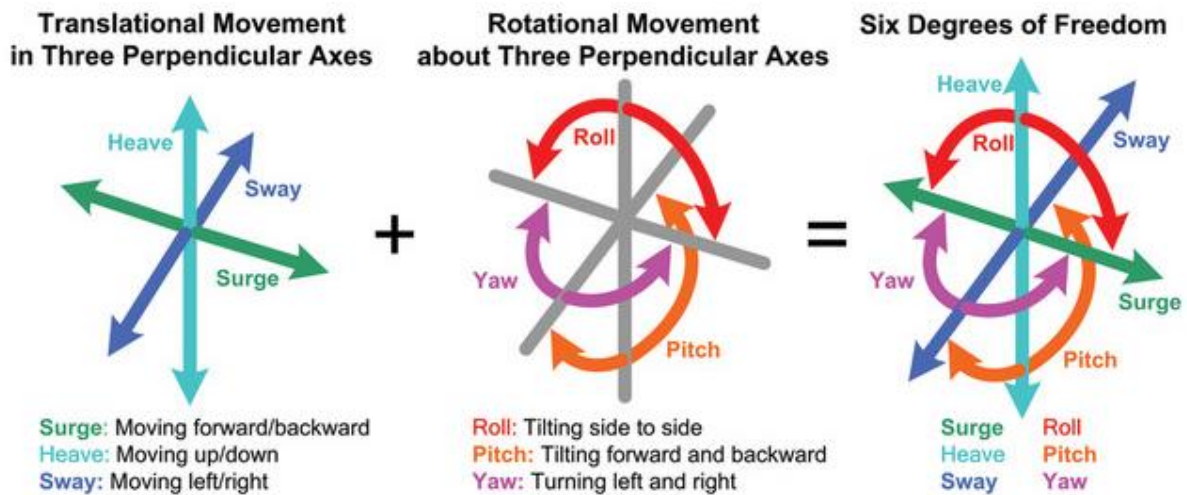


Figure 2.2 – A breakdown of the six degrees of freedom (Leading Ones, 2016).

Interestingly, studies widely differ in their conclusions when analysing the impact of motion controls on user interaction and enjoyment. Existing literature suggests that naturally mapped controls often lead to a greater sense of enjoyment for users (Skalski et al., 2007; Lindley et al., 2008) due to more natural movement and interaction within a game (Lindley et al., 2008). Furthermore, this research also mentions that the advancement of motion controls may not necessarily improve the overall user experience as such advancements do not automatically prompt a more positive response (Limperos et al., 2011). Schmierback et al. (2012) discuss these notions, contextualised towards general Human-Computer Interaction (HCI). The paper compared the use of a steering wheel to a more traditional controller when playing a racing game, to assess whether the differing motor actions impacted upon the user's sense of immersion and interaction with the task. The results of the study suggested that more realistic interaction with the game, in the form of the steering wheel, had a positive impact upon not only the user's enjoyment but also allowed them to feel more immersed in the experience (Schmierback et al., 2012). Further research into motion controls, specific to VR applications, produced similar results, explaining that users felt more immersed when manipulating a scene with their own hands (Renner et al., 2010). The study by Schmierback et al. also suggests that presence and immersion are heavily interlinked, whereby an enhanced sense of presence encourages a greater feeling of immersion for the user (Schmierback et al. 2012). In addition to this, previous research suggest that presence is also heavily influenced by immersion (Slater et al., 1996), indicating that a loop exists in which one sensation can be created and amplified by the other. This further justifies the aforementioned decision to value robustness, and thus a more reliable sense of presence, over the enhanced dexterity of vision-based finger tracking, when considering the artefact design. Despite contributing towards the design of the artefact, this research does not indicate whether finger motion tracking will successfully enhance immersion. However, studies have suggested that players feel disembodied when they cannot see their hands within VR (Eustler et al., 2015), something that will not occur in this project. Therefore, an assumption can be drawn that an interactive visualisation of the user's hand will enhance the overall immersive experience.

Finally, existing literature discusses methods which allow for the effective testing and evaluation of immersion and presence. A common theme throughout papers conducted in this area first start by providing a comprehensive description of immersion or presence, depending upon the term which pertains to the desired testing. Immersion, for example, is commonly described as the state whereby a user is absorbed, or completely involved, within a scenario, captivated within a homogeneous world without external distraction (Jones, 2000; Jennett et al., 2008; Jankowski and Decker, 2012). This was simplified in the Oxford dictionary, defined as a state of “mental absorption” (Gander, 1999) and more recently updated as “deep mental involvement in something” (Anon, 2016a). Presence is more complicated to define. In general terms, presence is a “state or fact of existing, occurring or being present” (Anon, 2016b), however, this definition differs to that of virtual presence, a term used when considering virtual applications. In this instance, virtual presence describes the extent to which a user believes they are elsewhere rather than their physical location, whilst engaging with a computer-generated simulation (Barfield et al., 1995; Slater and Steed, 2000). These definitions indicate similarities between the two sensations but also assure the importance of distinguishing between them. Slater et al. covers this distinction well, expressing that immersion is an objective description of what a particular system provides whereas presence is a state of consciousness, the psychological sense of being in a virtual environment (Slater et al., 1996). Having established accurate descriptions of both immersion and presence, attention can be given towards establishing a number of testing and evaluation methods for each.

Jennett et al. (2008) evaluate various methods for testing immersion within video games. The paper highlights key principles for testing immersion which, despite the methods not directly pertaining to virtual reality, remain valid due to the consistent definition of immersion across both generalised and specific applications. The conducted experiments involved three separate tasks, each amounting to the participant filling out a questionnaire, the contents of which were altered between tasks. A conclusion drawn from the study expressed that immersion applies not only to positive emotions, but also those associated with negativity and uneasiness (Jennett et al, 2008). Suggesting that the questions presented to participants should evaluate not only the ease-of-use but also provide a method of feeding back emotions such as frustration and difficulty during interaction. Brown and Cairns (2004) explain the effect of emotional involvement, both positive and negative, as a barrier to total immersion, identifying a link between gamers who experience a lack of empathy and a resultant lack of immersion. This further supports the requirement for users to express emotional engagement with a virtual reality application as it could indicate that any improved immersion is a result of negative emotions rather than improved interaction. The emotional response of frustration is of particular interest, as this would indicate that the user is experiencing difficulty when interacting with the virtual environment. This might have a psychological effect of the ambiguity of the task presented to the user, which Flow Theory (Csikszentmihalyi, 1996) suggests would result in heightened distraction as the task becomes increasingly arduous. Therefore, the immersive experience would be moderated by distraction, increased difficulty and thus reduced ability to complete the given task, as discussed by Jennett et al. (2008). This consequently means that a questionnaire should allow users to feedback on their ability to interact with their environment, perhaps asking whether interaction was better or worse when using the data glove when compared to the standard HTC Vive controller. Conclusively, the questionnaire produced for this project will examine emotional responses exhibited by the user in addition to analysing the overall ability to interact with objects within the virtual play space. Research suggest that this would produce desirable statistics from which a conclusion can be made.

3 Development Life Cycle

3.1 Project Management

3.1.1 Methodology

The processes and structure of software development are defined by methodologies which play a vital role in how successful a project will be. These methodologies organise the basic processes of software development with the aim of maximising efficiency and quality when creating a specific produce. To do this, methodologies are designed to accommodate for specific project requirements and conditions thus making it crucial to select a methodology which best suits the requirements of the given project. Software Development Methodologies (SDM) can be categorised as ‘Traditional’ and ‘Agile’, whereby the key difference is that Agile methodologies focus on iterative cycles (Khan et al., 2011). This distinction between the two categories holds great importance when determining which methodology to utilise when developing software.

Fowler (2002) further elaborates upon some key characteristics of Agile SDMs explaining that they are adaptive and people-oriented, rather than predictive and process-oriented. Essentially meaning that Agile methodologies accommodate change and cater more towards user requirements, differing from Traditional methodologies which resist changes in user requirements.

When determining the methodology best suited to this project, it was important to consider the aims and objectives, as previously outlined. These would provide a clear definition of any processes which must take place to produce the completed artefact. The first observation indicated that development could be segmented into a number of smaller stages, as outlined below:

- **Hardware** – The development of all software components is considered to be dependent upon the initial creation of the physical artefact, in the form of a Data Glove. This stage describes the process of designing the Data Glove itself, physically connecting electrical components to the a chosen Arduino board, and testing to ensure the artefact works as intended prior to the development of dependent software.
- **Interfacing Software** – The development of interfacing software both within the Arduino IDE and Unity to allow communication between the Data Glove device and the computer. Design regarding user interaction is not necessary at this stage, although the code structure will be considered. This stage to subject to pre-requisites from the hardware phase and thus must only be attempted when the hardware is of an acceptable standard.
- **Evaluation Software** – This stage describes the design and development of a testing scenario within Unity which will promote the fair and accurate evaluation of the artefact against the standard HTC Vive controller device. The testing of this scenario will be dependent upon the completion of previous states to an acceptable standard.

As aforementioned, software development which can be separated into stages will benefit from Agile approaches which allow for iterative development, whereby specific components are developed in an incremental manner and combined into a finished product (Beck et al., 2001; Fowler and Highsmith, 2001; Williams and Cockburn, 2003). This is well suited to the breakdown of stages for this project as each stage is reliant on the completion quality of the previous. For example, it would be more difficult to identify a precise issue with the artefact if successful hardware testing had not taken place prior to implementing software-based interfacing. Traditional methodologies, such as Waterfall, do not accommodate this type of continuous testing, instead forcing bulk execution of all development processes, meaning tests only take place at the end of development (Balaji and Murugaiyan, 2012).

The second consideration is whether the project would benefit from the improved adaptability of Agile methodologies. Whilst the project is user-centric in nature, the requirements of the artefact are not expected to change during development thus making adaptability irrelevant. Therefore, whilst Agile methodologies allow software development to be flexible concerning changes to user requirements (Baskerville et al., 2002; Fowler and Highsmith, 2001; Highsmith and Cockburn, 2001), this provides no benefit to the project. A traditional methodology may therefore prove more suitable, whereby the requirements of the project are clearly defined at the start of development and are not subject to change. However, the project describes a wearable artefact with the aim of uncovering information regarding interaction and immersion as a final result of the study. For this reason, interaction and communication with target users is unlikely to produce any information which cannot otherwise be deducted from a continuous self-testing process. Distinguishing a methodology based on changes in user requirements is therefore unnecessary, as such changes will not occur within the context of this project.

3.1.1.1 Rapid Application Development (RAD)

Rapid Application Development (RAD) is an Agile SDM which emphasises fast development and delivery of high quality systems (Martin, 1992). The methodology builds upon the incremental development model whilst allowing multiple cycles to be executed in parallel, allowing for rapid prototyping in the form of phased deliverables (Beyton-Davies et al., 1999). Fundamentally, Martin's (1992) model of RAD is broken down into 4 phases whereby design and construction is executed iteratively.

The initial phase of RAD involves requirements planning and is only executed once during the life cycle of a project. This stage involves the clear outlining of project requirements and may specify a series of phased deliverables (Beyton-Davies et al., 1999). Forming phased-based deliverables is particularly important when considering RAD as it provides a clear overview of the requirements at each iterative stage. Once the requirements are specified, the design and development stages are executed within a loop until the requirements of the specific cycle have been met. This allows the design of a particular feature to adapt as the project is being developed, provided that the final design still fulfils the previously specified requirements. Finally, consolidation and testing takes place across the system as a whole to ensure that the overall requirements have been successfully met. This is referred to as the 'Cutover' stage, which describes the testing and deployment of combined iterative elements as a finished product. As seen in Figure 3.1, testing is emphasised as a crucial process within the final stage of development.

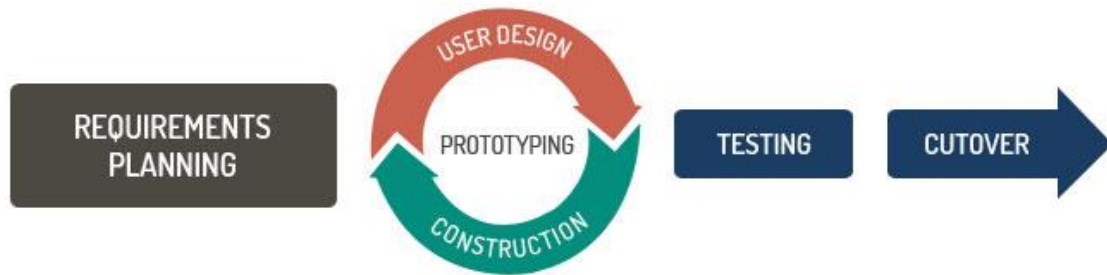


Figure 3.1 - Illustration of Rapid Application Development (TatvaSoft, 2015).

This is well suited to the artefact development of this project which is heavily reliant upon an incremental development process. Additionally, the iterative stage within RAD is highly versatile, allowing development to take precedence over design and vice versa. This provides a lot of flexibility across the drastically different stages of this project, allowing hardware development to benefit from multiple design iterations whilst the smaller interfacing stage can be implemented with minimal design.

Whilst RAD provides a series of advantages which specifically aid this project, it has one drawback. As previously stated, this project will feature three major stages of development which each rely upon the completion of the previous stage as a pre-requisite. However, RAD is most effective when each iterative cycle is independent of the others, which is not the case here. Whilst the methodology can be applied in a sequential manner, allowing cycles to be executed linearly, the benefits with respect to speed of development would be negated. This does not however pose any threat to the development of the artefact, as strict implementation deadlines can be utilised to minimise the risk of failing to meet project milestones. The implementation of such deadlines, within iterative cycles, is referred to as “time-boxing” (Carmel et al., 2010) and is a principle which is regularly associated with the RAD methodology (Beynon-Davies et al., 1999; Howard, 2002).

Conclusively, RAD presents a number of advantages which benefit this project specifically, which have been outlined and justified within this section. For these reasons, the methodology has been chosen for use when developing both hardware and software components, and combining them to form a completed artefact.

3.1.2 Time Management

Time management is a project management process whereby activities are defined, sequenced and assigned an estimated duration with the aim of ensuring a timely project completion time (Kwak and Ibbs, 2002). A number of time management tools and techniques are available to maximise the effectiveness and efficiency of the process, most notable of which is the ‘Gantt Chart’. Developed by Henry Gantt during the early 1900s, this chart facilitates the visualisation of time information (Maylor, 2001), typically depicting project milestones and processes with respect to start and completion times. The chart is frequently used as a planning tool in software engineering whilst also facilitating simple communication between team members, sponsors and stakeholders. As such, Gantt charts are highly beneficial when used for long-term planning (Guide, 2004) and are therefore extremely useful when using traditional methodologies such as Waterfall or Spiral. On the other hand, research suggests that these charts are often ignored or made obsolete shortly after starting projects with a more Agile approach, such as Extreme Programming (XP) (Michele Sliger, 2004). Furthermore, when appropriate, many agile projects involving teams will discard the detailed Gantt chart in favour of Sprint logs and backlogs (Lewis and Neher, 2007; Gustavsson and Rönnlund, 2010), which is often the case when utilising the SCRUM methodology.

Despite using RAD, an agile methodology, a Gantt chart was created for this project, as shown in Figure A.1. However, as the project proceeded, tasks were completed with less conformity to the original deadlines and milestones. This was mainly due to lack of familiarity with the functionality of electrical components. As a result, hardware required adjustment numerous times during the software development process. This section explains the differences between planned and actual execution of the project.

3.1.2.1 Iteration Plan

While the execution time of the project remained the unchanged from that shown in Figure A.1, it became immediately evident that several more iterations would be required to break the artefact development process into more practical phases. The result of this realisation is an iteration plan, shown in Figure 3.2, detailing the order of execution for each stage of development. The numbers shown at each step directly correlate to the iterations presented within this document, providing clarity regarding their chronological execution. It should be noted that the plan does not specify precise completion times, this information is instead taken from the Gantt chart. The main body of implementation for each development type will take place within those specified time frames, as indicated by the darker coloured blocks along the iteration plan timeline.

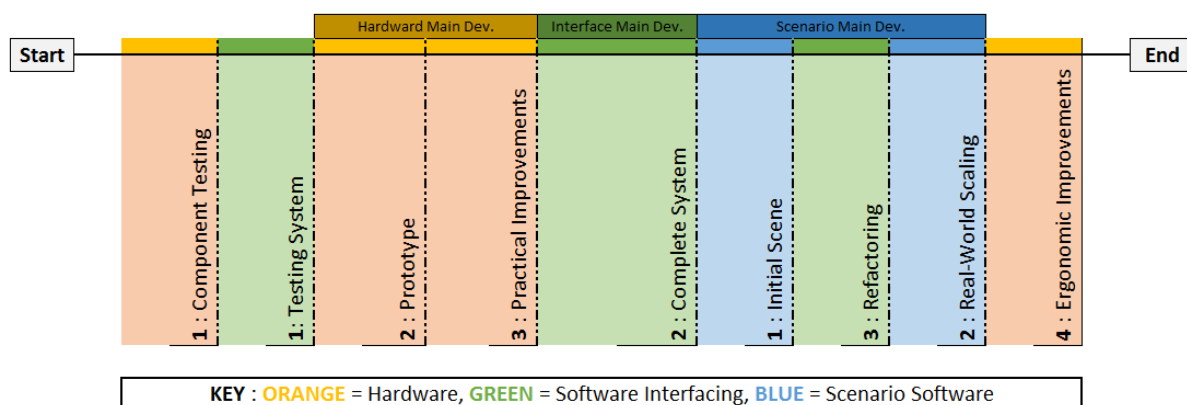


Figure 3.2 – Iteration plan for the artefact development life cycle.

3.1.2.2 Hardware Development

At the start of the project, a large portion of development was assigned to constructing and testing the hardware components of the artefact. Whilst the majority of hardware development took place during this period, a significant number of modifications were made during the implementation of software interfaces between the Arduino and the computer. This is due to a lack of understanding with respect to the electronics used, namely the flex sensors, resulting in the requirement to learn circuitry as the project progressed.

Another reason for adapting the time allocated to hardware development was due to the delayed acquirement of components. Specifically, only two flex sensors were available at the beginning of development meaning that additional phases of hardware development were required to fully build the Data Glove. This is further discussed within the risk management section of this report.

Additionally, the artefact hardware was implemented in several states, initially focusing on correct functionality and later evolving to become more ergonomic and usable. This meant that the initial plan of only one hardware development phase was invalid, resulting in a number of adjustments being made to the overall time plan. However, this mainly effected the scenario development and is thus explained in greater detail below.

3.1.2.3 Scenario Development

The time plan for developing the testing scenario, named the Interactive Environment (IE) within the Gantt chart, was effected as a consequence of adjustments made to hardware development. This reduced the amount of development time allocated to creating the scenario meaning that the implementation was not completed before the assigned milestone, as seen in Figure 3.3. Fortunately, user testing had been allocated time in excess, creating a buffer against issues which could have occurred during development. Consequently, the impact of this issue was mitigated, ensuring that project completion remained on schedule.

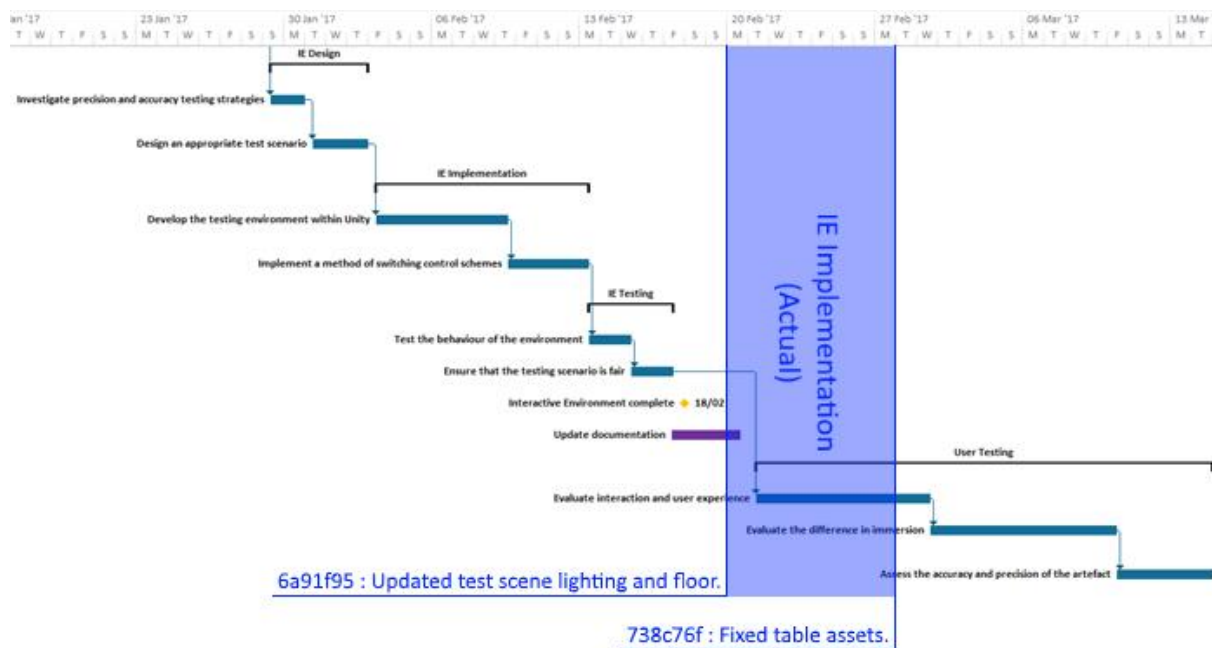


Figure 3.3 – Differences between planned and actual IE implementation. Snippet of full project Gantt chart (Figure A.1) with timestamp references to GitHub commits (Figure A.2).

3.1.3 Risk Management

Risk	Likelihood (1-10)	Impact (1-10)	Mitigation
VR space is unavailable to use due to either hardware issues or sharing of space.	8 - High	9 – Very high <ul style="list-style-type: none"> Without access to testing space, debugging VR related aspects of the project would be impossible. User testing will not be able to take place at the time when the risk occurs. 	<ul style="list-style-type: none"> Re-arrange the project plan in order to compensate for VR debugging time lost, and focus on other tasks that do not require the specialized hardware. Postpone user testing until VR space is obtainable through scheduling. Reduce the scope of testing to compensate for time lost, according to the project plan.
Unable to interface the sensors with the computer system and consequently Unity.	6 - Medium	9 – High <ul style="list-style-type: none"> The project relies heavily upon Unity for testing purposes, therefore an inability to interface with the software would be highly detrimental to the project. 	<ul style="list-style-type: none"> Issues will likely arise from incompatible programming languages used for Unity and the Arduino, therefore Unreal Engine 4 would be used instead which uses C++ instead of C#.
Users experience discomfort, such as motion sickness as this is a common issue related to VR.	7 – High	6 – Medium <ul style="list-style-type: none"> May affect the already limited time available to use the VR equipment. Motion sickness with a particular controller may influence user satisfaction during the next controller test. 	<ul style="list-style-type: none"> Reduce the number of test participants in order to adhere to time constraints. Ask users if they experience motion sickness when travelling and factor their response into evaluation.
VR headset equipment is unavailable to use for testing or implementation.	3 – Low	8 – High <ul style="list-style-type: none"> Without access to a VR headset, debugging motion controls will not be possible. User tests will not be able to take place at the time when the risk occurs. 	<ul style="list-style-type: none"> Re-arrange the project plan in order to facilitate tasks that do not require access to the VR headset for debugging. Re-arrange user testing for a different date in order to reduce impact on evaluation. Reduce the scope of testing for the feature concerned in order to adhere to time constraints.

Arduino experiences a power malfunction and causes irreversible damage to the hardware.	3 – Low	7 – High <ul style="list-style-type: none"> Analog to Digital Conversion (ADC) will not be able to take place using the Arduino, breaking communication between sensors and the computer. 	<ul style="list-style-type: none"> Attempt to replace the Arduino with a similar model or alternatively source a Raspberry Pi and ADC module. Adjust project plan to factor in time lost during replacement, reducing user testing time and also the overall ambitiousness of the project.
Glove equipment becomes damaged, causing a flex sensor malfunction.	5 - Medium	4 – Low <ul style="list-style-type: none"> Motion controls will not map correctly to the 3D model within Unity. Replacement components will incur extra costs which have not been accounted for. 	<ul style="list-style-type: none"> Reduce the scope of the project to only assess motion tracking in fingers where the sensor is functional. Map some of the remaining sensors to multiple fingers to ensure that they still move in user testing. Order replacement equipment and modify project plan, accounting for delivery time where necessary.
Movement of the bend sensor does not accurately represent that of the finger.	2 – Very low	6 – Medium <ul style="list-style-type: none"> User may become disorientated or confused if their joints aren't moving as expected in VR. The project would experience a negative impact on the overall immersion. 	<ul style="list-style-type: none"> If the issue is related to a small selection of sensors, attempt to configure the effected sensors differently and assess the impact. If the issue was unresolved, hardware issue is the likely cause and therefore new sensors will be sourced to replace those which are malfunctioning.
HTC Vive controllers do not accurately represent movement in the wrists when strapped to the forearm.	4 – Low	3 – Low <ul style="list-style-type: none"> Finger tracking still functions as intended, however the hands are rotated incorrectly within the virtual environment. User may feel uncomfortable if their hands do not orientate as expected, bringing testing to a halt. 	<ul style="list-style-type: none"> Adjust positioning on the controller along the forearm in an attempt to remedy the issue without sourcing new hardware. If the above fails, reduce the scale of the project to concentrate purely on finger tracking and exclude immersion tests.

3.2 Hardware Development

3.2.1 Foreword

A software interface between the Arduino and computer is required to test each stage of hardware development. Consequently, all hardware testing results are synonymous with those of the software interface, presented in Section 3.4. Testing sections have therefore been excluded from each iteration within this section and conclusions regarding the effectiveness of hardware implementations should be made considering software testing results presented in Section 3.4.

Due to the prerequisite of a working software interface to test the hardware artefact, a divergence was made from hardware development after the first iteration listed in this section. This led to the early implementation of a working software interfacing solution, presented in Section 3.4.1. Please refer to the time plan presented in Section 3.1.2 for clarity regarding the chronological execution for hardware and software interfacing iterations.

3.2.2 Iteration 1 – Component Testing (Arduino Uno + Breadboard)

3.2.2.1 Design

Minimal design is required at this stage because this iteration only focuses on the creation of a working hardware solution, in its simplest form. However, a clear understanding of the circuitry required for the artefact to function correctly is crucial to the success of the project. Thus, designs created at this stage of the development process are critically important, despite only focusing on the electrical components within the artefact.

The Flex Sensor Data Sheet (Spectra Symbol, 2014) provides useful information concerning the flex sensor and features a basic sensor circuit, as seen in Figure 3.4. This circuitry diagram gives a clear understanding of the input and output requirements for flex sensors which is particularly useful when designing the overall circuitry for the Data Glove. Furthermore, it provides insight into how the flex sensor works. It is observed that the flex sensor is a type of variable resistor which can be used in conjunction with an additional fixed resistor, forming a voltage divider, as shown in Figure 3.4. This combination allows the flex sensor to “take away” a variable amount from an input voltage. The diagram shows that the result of this process is taken as the voltage output, which can then be used to determine the resistance value of the sensor. The exact method used to determine this resistance value is described under Section 3.4, which addresses the software used to interface between the Arduino and computer.

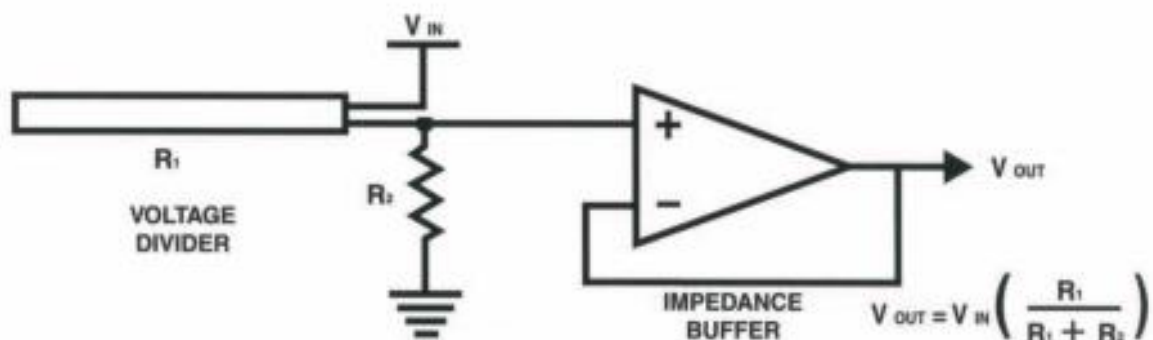


Figure 3.4 – Flex Sensor circuitry diagram (Spectra Symbol, 2014).

Figure 3.5 shows the design for the circuitry of the Data Glove which was produced in accordance with the aforementioned observations regarding the Flex Sensor Data Sheet. In the diagram, the sensors are shown as S_1 and S_n , suggesting that n number of sensors can be used. However, it should be noted that n is limited by the number of analog inputs provided by the Arduino board being used. N is used to show that each sensor is identically connected to the board and consists of the same circuitry, shown on the left side of the diagram. The circuitry representing the flex sensor is heavily simplified in Figure 3.5, as the diagram is designed to show overall functionality of the Data Glove, including its Serial connection to the computer, rather than electrical workings.

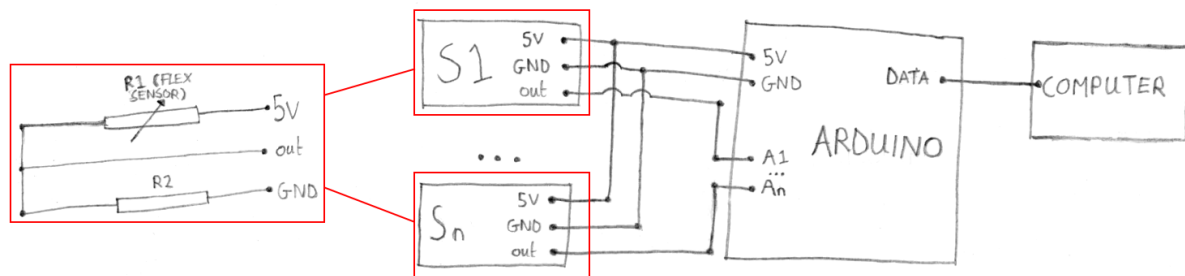


Figure 3.5 – The overall circuitry for the Data Glove, whereby the leftmost diagram shows the circuitry for each flex sensor. Symbols are formatted using IEC-60617 specifications (Autodesk, 2016).

3.2.2.2 Implementation

A 'breadboard' is used in conjunction with an Arduino Uno, as seen in Figure 3.6, to facilitate rapid prototyping and testing at this stage of the hardware development process. Breadboards are one of the most fundamental pieces of equipment when learning how to build a specific circuit (Spark Fun, 2016) because they support solder-less connections. To do this, the breadboard

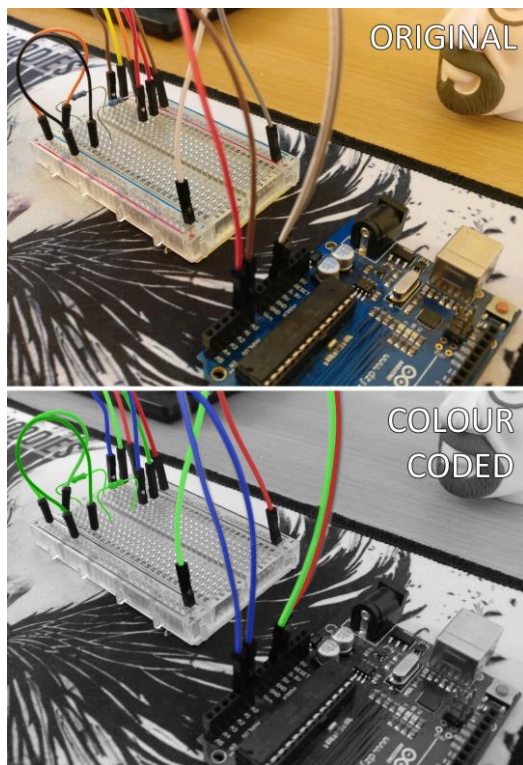


Figure 3.6 – Test implementation using the Arduino Uno. Colour code: Red = Live (5V), Green = GND, and Blue = Output.

provides a vast number of female pin connectors which are interconnected using 'power rails', meaning that a wire can be inserted at any point along a rail to become a part of the circuit. This equipment allowed issues to be identified and fixed at far greater speeds than would have been possible when using solder-based solutions.

Figure 3.6 shows the finalised test implementation using the breadboard, alongside a colour coded version of the image to provide greater clarity with regards to the connections. This implementation is a direct representation of the circuitry that was previously designed, where the blue wire represents outputs taken after variable resistance has been applied across both sensors. The flex sensors are not visible within the image provided, although the connections are visible as vertical green and red wires towards the back of the breadboard. The red wire represents the voltage input of 5V taken from the Arduino VCC pin, whereas the green wire is connected to the GND pin on the Arduino via a fixed resistor. The resistor used in this implementation has a value of 20K Ohm.

3.2.3 Iteration 2 – Prototype (Arduino Uno)

3.2.3.1 Design

Having created a working electrical circuit for the Data Glove, using a breadboard and Arduino Uno, the next stage is to move towards a more functional and permanent design. The breadboard solution presented in Section 3.2.2 is therefore no longer a viable solution, and must be abandoned in favour of a more streamline implementation. However, the hardware is still very much within the prototyping portion of the development process and the solution presented in this design must continue to provide quick and easy troubleshooting. In addition to this, the new solution must be able to support multiple sensors without using the power rails provided by the breadboard. This presents an issue, as Arduino boards only provide one VCC or 5V pin, and similarly only a single GND pin. Therefore, some type of adapter must be created to allow a single pin to support several connections, whilst avoiding permanent solutions such as those involving soldering. The design shown in Figure 3.7 should satisfy these criteria when connected to the VCC and GND pins on the Arduino board.

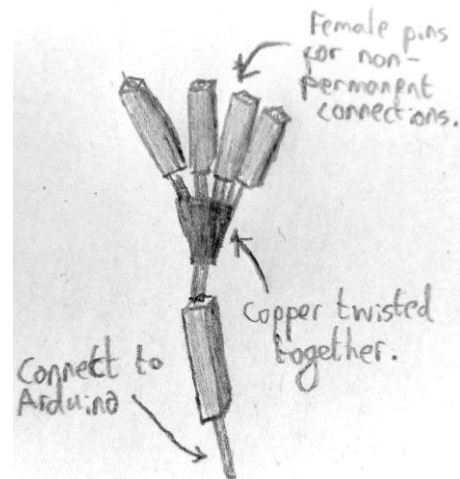


Figure 3.7 – Adapter cable, designed for use with the Arduino Uno.

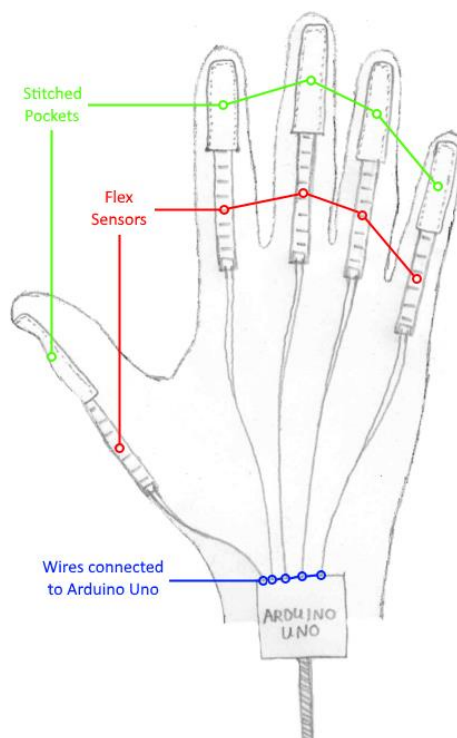


Figure 3.8 – Data Glove design (Phase 1).

This design stage also aims to address the features required to produce a wearable device which is both robust and flexible, allowing free movement for the wearer. Figure 3.8 illustrates the initial design for such a device, detailing the use of several ‘pockets’ to affix the flex sensors to the glove. One key issue for consideration is that the pockets must prevent the sensors from leaving the wearers hand, whilst also allowing them to slide freely. This is important because of the increased physical distance between each sensor and the Arduino when clenching a fist, as opposed to the user holding their hand flat. It is essential to ensure that each sensor remains in a position on the finger where it can provide accurate resistance readings, whilst also avoiding any strain damage being inflicted on electrical components. Allowing the flex sensors to slide freely will ensure that these criteria are met.

One final consideration concerns the type of glove used for the artefact, whereby it must fit to the wearers hand but also accommodate for varying hand sizes. To achieve this, the chosen glove will have a high elasticity.

3.2.3.2 Implementation

During implementation, two attempts were made at creating flex sensor pockets on each finger, adhering to the requirements previously discussed. The first design, labelled as 'A' in Figure 3.9, uses small loops of woollen fabric at the tip and knuckle of the finger. This provided all required functionality, as previously discussed, allowing the flex sensor to slide freely whilst also preventing the sensors from dislodging. However, the material was deemed unsuitable as it was especially prone to wear and tear, quickly becoming frayed which weakened the stitching on the glove. The design, labelled as 'B' in Figure 3.9, used nylon, a considerably stronger material. This eliminated the issues seen within the first design whilst providing equal levels of desired functionality. Unfortunately, this caused a problem, as nylon is a highly conductive material making it unsuitable for use with electrical circuitry. However, anti-static shielding was created to remedy this, fitting around the flex sensor as it was inserted into the pocket. This comprised of electrical tape and recycled anti-static material taken from hard drive packaging, shown in Figure 3.10.

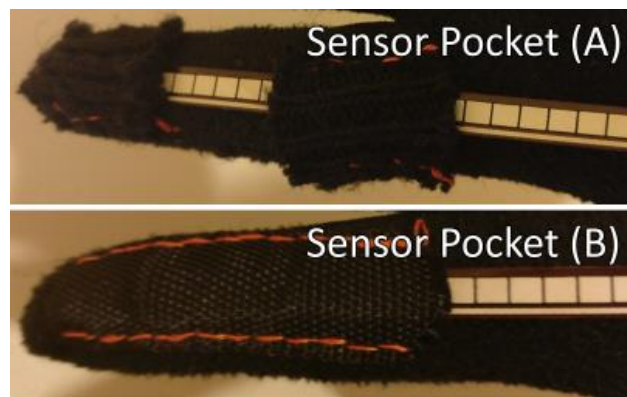


Figure 3.9 – Two sensor pocket implementations, B represents the more favourable design, chosen over A.



Figure 3.10 – Electro-static sleeves, created using recycled anti-static shielding and electrical tape.



Figure 3.11 – Final pocket design, shown on both left (blue) and right (orange) gloves. The design is pictured without anti-static shielding.

Finally, some adjustments were made to the flex sensors to provide greater consistency regarding resistance values and to increase the strength of the component. It was particularly evident during the previous iteration that the sensor was incredibly weak at the connection point, often bending at a very sharp 90-degree angle during use. This interfered with the provided resistance values and an adjustment was therefore made to improve the consistency of results and connection strength. A heat shrink was added at connection points, partially covering each flex sensor in addition to the soldered wires and fixed resistor. This can be seen on the two sensors pictured in Figure 3.12.



Figure 3.12 – Prototyping flex sensor implementation, using copper wiring and header pins.

Figure 3.13 pictures the resultant artefact after this implementation process, utilising an adapter cable on both the VCC and GND pins of the Arduino Uno, as previously discussed. Some noteworthy observations regarding this solution are presented below.

Firstly, the copper wires used are highly in-flexible. This means that the wires are resistance to any adjustment after being oriented upwards due to the pin-based connections. The use of silicon wires may be able to remedy issues regarding flexibility, whilst soldering the wires to the board will eliminate the vertical orientation.

Lastly, the board is far too large and cannot be oriented in such a way that would allow a connected USB cable to run up the arm of the user. Using the Arduino Pro Mini or a smaller model of the board may remedy these issues without losing any of the connections required for this project, namely the VCC, GND and A0-A6 connection points.

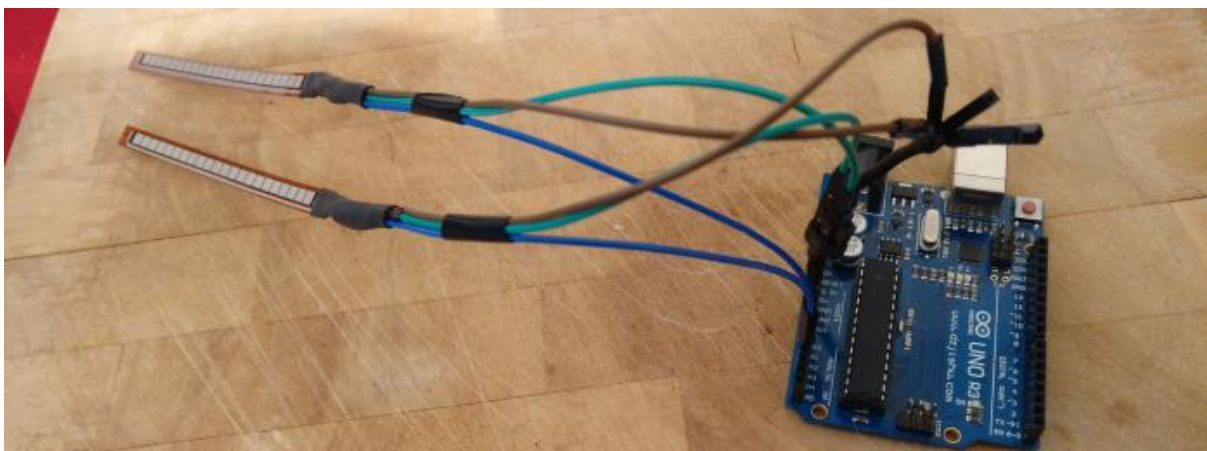


Figure 3.13 – Final prototyping solution, using the Arduino Uno.

3.2.4 Iteration 3 – Practical Improvements (Arduino Pro Mini)

3.2.4.1 Design

At this stage, a large portion of the design and development has taken place and all the necessary functionality has been determined and implemented within the artefact. However, this iteration aims to address the issues discussed in Section 3.2.3.2 regarding the practicality of the glove. Figure 3.14 shows a very similar design to the previous iteration with a few subtle differences. Firstly, the Arduino Uno is replaced by the Arduino Pro Mini, a device which is considerably smaller than the previous board whilst providing the same functionality. Secondly, the connections for each wire are more clearly defined, showing the disconnect between live, ground and output wires. This diagram represents the goal, in terms of aesthetics, for this stage of the development process, whereby each wire will be routed through the front of the Arduino to create shorter and neater connections than those seen in the previous iteration. These changes layer over previous implementations to form the final design of the artefact with regards to its electronics and components.

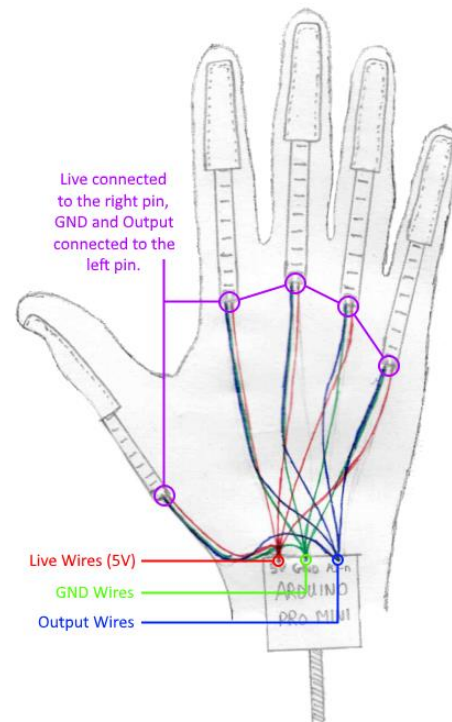


Figure 3.14 – Data Glove design (Phase 2).

At this stage, the aim is to ensure that the artefact offers similar functionality to that seen in the Manus VR glove, allowing the user to take full advantage of the finger motion tracking provided by the glove when performing gestures and actions. The design accommodates the use of silicon wires and the Arduino Pro Mini to improve upon the previously lacking ergonomics of the artefact.

3.2.4.2 Implementation

In order to begin this stage of implementation, the connections between wires and the flex sensors needed to be dismantled. This involved carefully cutting and removing the previously applied heat shrinks before de-soldering each wire from its associated flex sensor, ensuring not to damage the fixed resistors as it would be ideal to re-use them. After this, work could begin on upgrading all the wires to a silicon variant and connecting them to the Arduino Pro Mini board which replaced the Arduino Uno that was previously being used.

This process involved firstly soldering 5 silicon cables together for both the ground and live connections, providing a more permanent version of the previously used adapter cable. It was important to ensure that each wire could reach the flex sensor with some additional slack to allow them to slide within their nylon pockets, as previously discussed. Secondly, each flex sensor was secured to all of the required wires, providing them with input and output connections. Red cables were used to represent live wires, whilst black was used for ground and white for the connections to the Arduino Analog pins, as seen in Figure 3.16. Ideally, these colours would conform to the new UK specification for electrical wiring, shown in Figure 3.15. However, the silicon wires were not available in these colours during implementation, resulting in use of a custom colour coding system.



Figure 3.15 – Old and new colour coding specification for electrical wires in the UK.

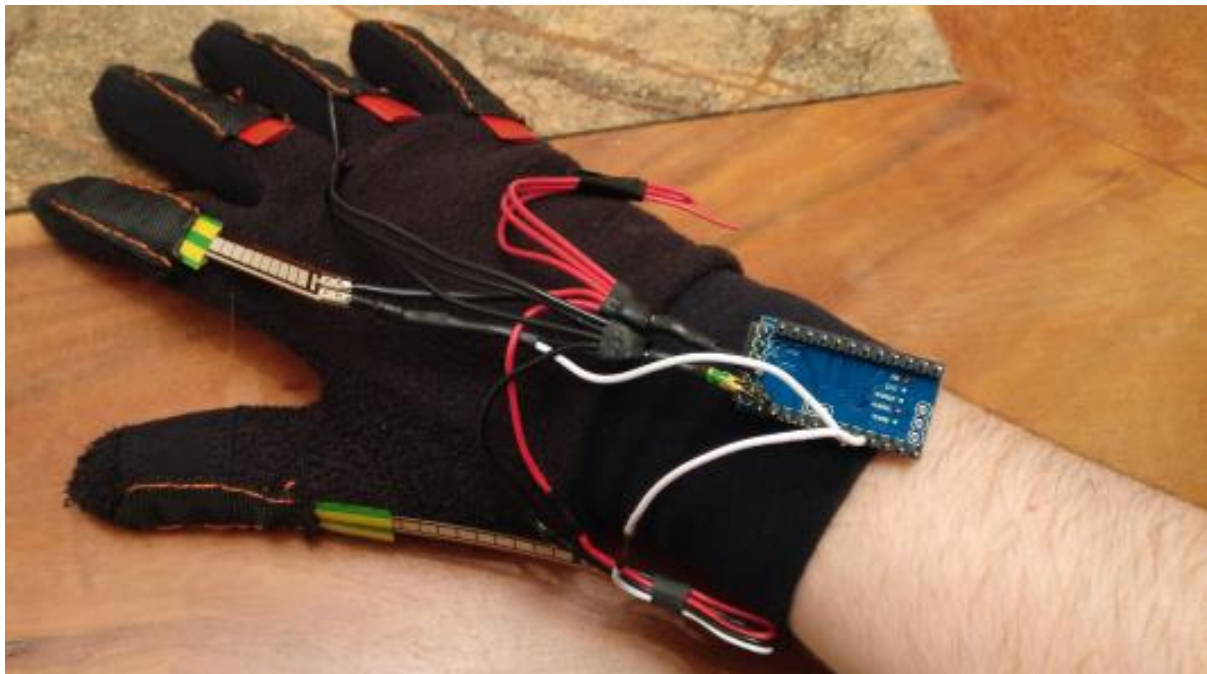


Figure 3.16 – Artefact implementation using the Arduino Pro Mini and silicon wiring. The wires no longer use pin-based connections and are instead soldered to the board.

As a finishing touch, heat shrinks were added to provide shielding and strength to the soldered connections, shown in Figure 3.17. This addition was also made in an attempt to solve the issue of flex sensors being raised and slightly twisted, as seen on the sensor attached to the index finger in Figure 3.16. Unfortunately, this did not fully resolve the issue and only prevented the twisting. Further efforts were not made to fix the raising problem in this iteration, as the glove still provided a satisfactory level of functionality for development and testing purposes. However, this will need to be revisited in a later iteration to ensure that each sensor performs optimally when operated by participants in the user testing stage.



Figure 3.17 – Finished electronics for the artefact, using four flex sensors. Heat shrinks were re-added for additional organisation, safety and support.

In addition to highlighting the use of heat shrinks to provide a more robust connection between sensors and wires, Figure 3.17 shows four connected sensors rather than the two seen in previous Figures. The reason for this is that the two additional sensors were only delivered at this stage of development, although connecting them to the existing artefact was just as simple as connecting the first two. Unfortunately, a fifth sensor never arrived, meaning that the sensor used for the ring finger would also need to control the fifth digit, or little finger. The decision was made to combine these specific digits due to the coupling between cortical neurons in both the ring and little finger meaning that the brain has difficulty properly distinguishing their individual movement (Aleesh Sankaran, 2012). This difficulty can be exploited, as an assumption can be made that the user will always move the ring and little finger in unison.

3.2.5 Iteration 4 – Ergonomic Improvements

3.2.5.1 Design

As this is the final iteration of hardware development, all existing issues must be resolved to produce a fully functional and practical artefact in preparation for the research investigations, which take place at the end of the project. Subsequently, this design stage will focus on the ergonomics of the artefact, detailing a case to be used for securely holding the Arduino. Figure 3.18 shows the design for such a case, detailing how each output wire will feed into the front and connect to Analog pins on the Arduino. This will conceal the routing required to neatly secure each wire.

Secondly, an addition has been made to the design which will address the issue of each sensor being raised outside of the currently existing pockets. Loops will be added to increase the amount of support applied to each sensor further up the wearers' hand, ensuring that each sensor remains flat across the points of the glove which impact the accuracy of resistance values.

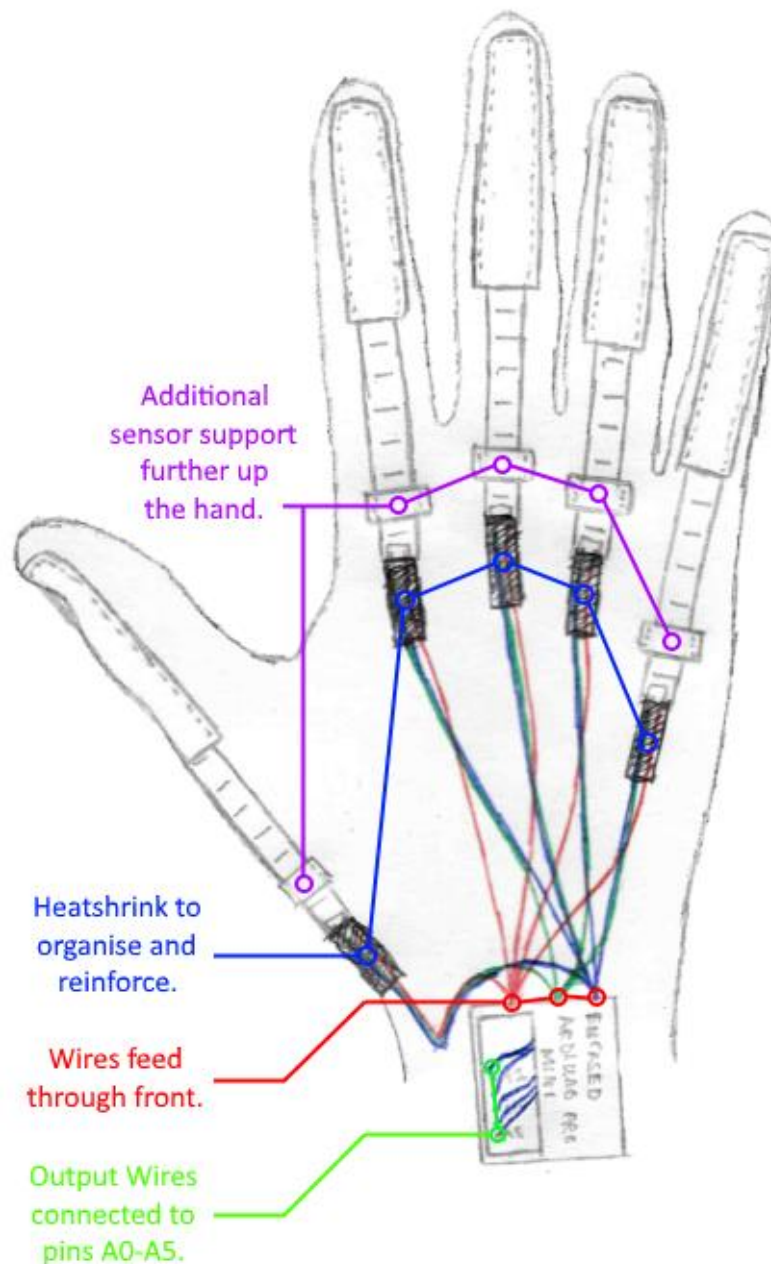


Figure 3.18 – Data Glove design (Phase 3).

3.2.5.2 Implementation

Blender was used to produce the model for the Arduino casing which would later be 3D-printed. This software provides the user with a broad array of tools to facilitate the 3D modelling process, including the edge scale GUI element which was used to ensure that the model was created to the correct dimensions. The high level of precision when creating the case for 3D printing was crucial, as it ensured that the Arduino had all of the necessary support when placed inside. Figure 3.19 shows the case within Blender, providing edge lines to demonstrate the achieved accuracy. The case measures exactly 2.4cm x 2.4cm x 4cm within the 3D modelling software, however this required scaling when converted to STL format due to the inconsistency between dimensions in Blender and 3D Builder. 3D Builder is simply a software used to view STL files on Windows, and allows models to be sent to a 3D printer. The model was scaled to 100x the original size and printed. Figure 3.20 shows the result of this process with a few adjustments made post-printing. Firstly, superglue was used to affix a standard small hinge to the 3D printed model, allowing it to open and close without having to detach the top and bottom each time. Secondly, foam padding was used as a protective lining inside of the case, this was mostly added as a precaution to ensure that the Arduino did not take any damage if it happened to move within the box. The padding also helped to negate minor discrepancies regarding the space within the box, buffering an additional few millimetres on each face to provide a snug fit for the Arduino board.

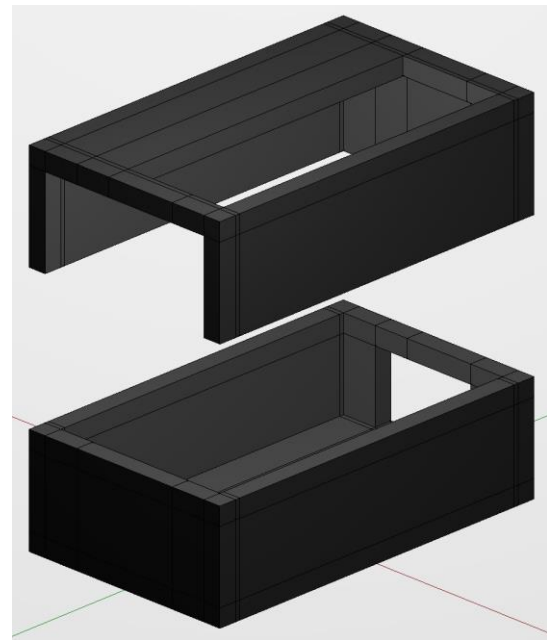


Figure 3.19 – 3D model of the Arduino case, created in Blender.



Figure 3.20 – 3D-printed box to contain the Arduino Pro Mini.

Lastly, although it is not visible in the image above, Velcro was glued to the underside of the case, allowing it to be easily connected and disconnected from the glove after used. This addition was made in the interest of portability, convenience and safety, allowing the electronic components of the artefact to be safely removed when transporting the Data Glove between locations.



Figure 3.21 – Velcro straps attached to the HTC Vive controller.

To prepare the HTC Vive controller for use with the glove, Velcro straps were added which could fit around the circumference of the wearer's arm, shown in Figure 3.21. This allowed the controller to be securely attached and detached from the underside of the users are, providing all of the necessary tracking for the hand of the user. However, a few functionality issues occurred when using this method. Firstly, the Velcro straps would sometimes fail to encompass the arm of the user, particularly in cases where the wearer had a larger profile of the upper forearm. However, using the

safety strap, provided on the HTC Vive controller as standard, provided a solution to this issue, allowing all users to comfortably use the device regardless of the circumference of their forearm. The second observed issue had a larger effect on the overall performance of the glove. In particular, the rotation of the forearm is not synonymous with the rotation of the wearer's hand. It was often the case that minor rotations applied across the length of the forearm would be amplified at the wrist of the user, creating a visual discrepancy when applying the rotation of the controller to the 3D modelled hand in VR. A fix for this issue was not implemented due to time limitations, although the use of smart mathematical functions may be able to fix this issue.

Figure 3.22 showcases the final additions to this implementation process, where four loops have been stitched just above the knuckle of the hand, to ensure that the sensor remains flat across all important areas. This ensured that the flex sensors remain secured to the fingers of the wearer to improve the accuracy of resistance readings. The image also shows a Velcro square fixed to the wrist portion of the glove.



Figure 3.22 – Data Glove without electrical components, detailing the addition of more supporting loops and a Velcro square for attaching and detaching the 3D-printed box.

The purpose of this square is to attach the Arduino casing at a point which does not impede the ability for each flex sensor to slide within its pocket, whilst also ensuring that wires do not become untidy at the front of the Arduino. This is done by providing the appropriate amount of slack on each wire, capable of reaching each sensor with a minimal amount of slack.

The finished artefact, shown in Figure 3.23, satisfies all aforementioned hardware requirements, with the exception of the minor issue concerning wrist rotation. However, this issue should not impact upon the research results for this project which simply aims to analyse finger motion rather than the mapping of entire arm movements and orientations. The only other concern relates to the in-ability for the case to close and lock, causing the Arduino to shift slightly during use. However, due to the precise scale of the USB connection hole in relation to the USB connector itself, the Arduino was unable to become dislodged. These compromises were made in the interest of time limitations and would have been solved had the resources been available to do so.

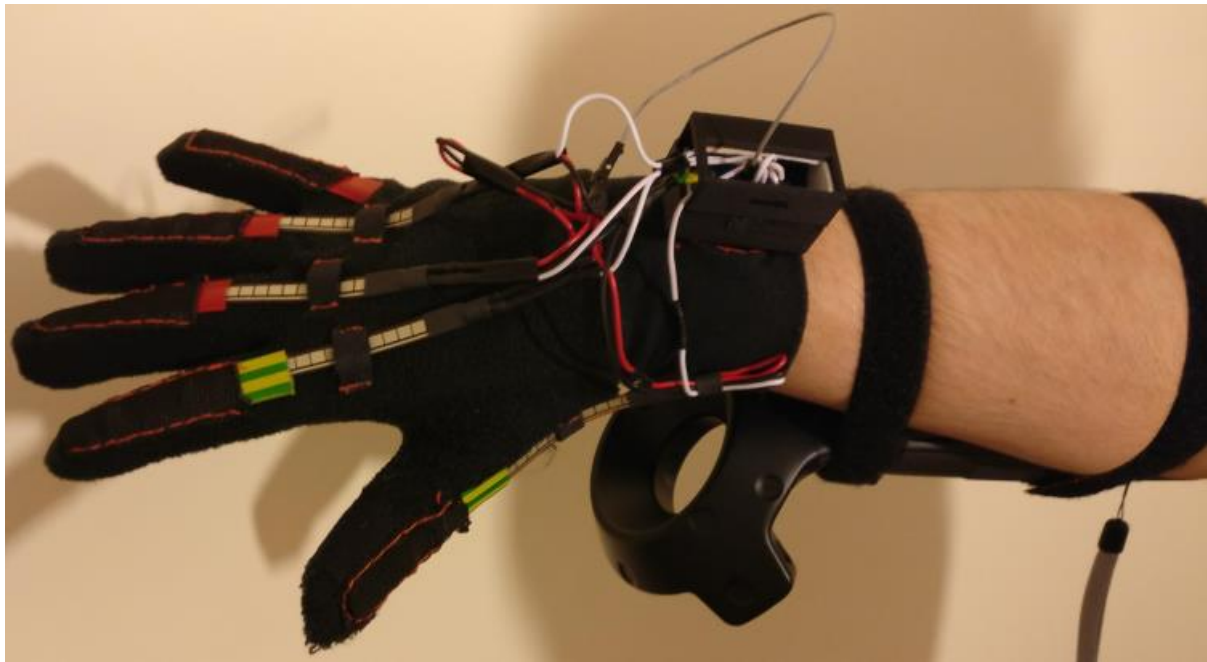


Figure 3.23 – Finished hardware artefact, shown with the HTC Vive controller affixed to the users arm.

3.2.6 Further Maintenance

No additional maintenance was required prior to user testing as the finalised hardware satisfied all of the minimum requirements to be suitable for use. However, a repair was necessary due to an incident which occurred midway through participants using the artefact. This was due to excessive force being used to remove the artefact from the hand of the user, causing the live connection to be broken between all sensors and the Arduino board. This was temporarily fixed by inserting a pin header into the point where each live wire converged and connecting it to the backup VCC pin on the Arduino board. This quick maintenance is pictured in Figure 3.23, represented by the grey cable elevated above the main body of the Data Glove, and allowed user testing to continue uninterrupted for all remaining experiments.

3.3 Software Development Tools

3.3.1 Integrated Development Environments (IDEs)

3.3.1.1 Visual Studio 2015

Visual Studio 2015 is an IDE from Microsoft which offers a vast range of features to facilitate the ability to develop, test and debug at phenomenal speeds. These qualities are ideal for the purposes of this project due to the chosen RAD methodology, which relies heavily upon the programmer's ability to rapidly prototype software. Perhaps the most desirable feature offered by each version of Visual Studio is Intellisense, a utility which provides real-time error checking and provides the ability to more easily locate required functions and variables by presenting the user with a drop-down menu. This is particularly useful when coupled with a game engine, such as Unity, or other service that offers a vast library of functionality because it reduces the requirement to constantly revise any online documentation. In summary, Visual Studio, like most large-scale IDEs, allows development to take place at a quick pace due to its vast array of utilities to aid the writing of code, something that is simply unavailable in many barebones applications such as Mono Develop and Notepad Plus-Plus.

3.3.1.2 Arduino IDE

The Arduino IDE is significantly more lightweight when compared to the likes of Visual Studio and Mono Develop, facilitating the writing of basic C programs for use on Arduino devices. However, a developer would not be expected to write humongous scripts and class-based solutions within the Arduino IDE due to the small storage capabilities on Arduino devices, preventing large programs from being stored. The interface is relatively basic, offering only a basic console output for debugging any written program and basic file IO functionality alongside an upload feature for sending the code to a board connected via USB. However, the Arduino IDE offers some unique functionality which is unavailable on alternative solutions such as Notepad Plus-Plus. The upload utility is exclusive to the IDE and allows programs to be easily compiled and uploaded to Arduino devices connected to the computer via USB cable. This capability makes up for the minimalistic interface and lack of advanced debugging utilities, providing a clear justification for using the IDE.

3.4 Interfacing Software Development

3.4.1 Iteration 1 – Arduino Software

3.4.1.1 Design

The core interfacing system will consist of a very small program, written in C using the Arduino IDE, to ensure that it can be stored on the very limited storage space provided by Arduino boards. This program will be uploaded to and executed by the board, meaning that the hardware is limited both in speed and available functionality, only allowing basic Analog-to-Digital Conversion (ADC) to take place in addition to most mathematical operations.

```
Input: sensorCount = N, VCC = 4.98, rDiv = 20000.0

start
  Initialise a serial connection
  while connection = true
    for i = 0; i < sensorCount; i++
      flexADC = Read analog pin A0 + i
      flexV = flexADC * VCC * 1023.0
      rString = rDiv * (VCC / flexV - 1.0)
      Print rString + ","
    end
    Print newline
  end
end

Output:
"XXXX.X,XXXX.X, ..."
XXXX.X,XXXX.X, ..."
```

Figure 3.24 – Pseudo code for the Arduino interfacing solution.

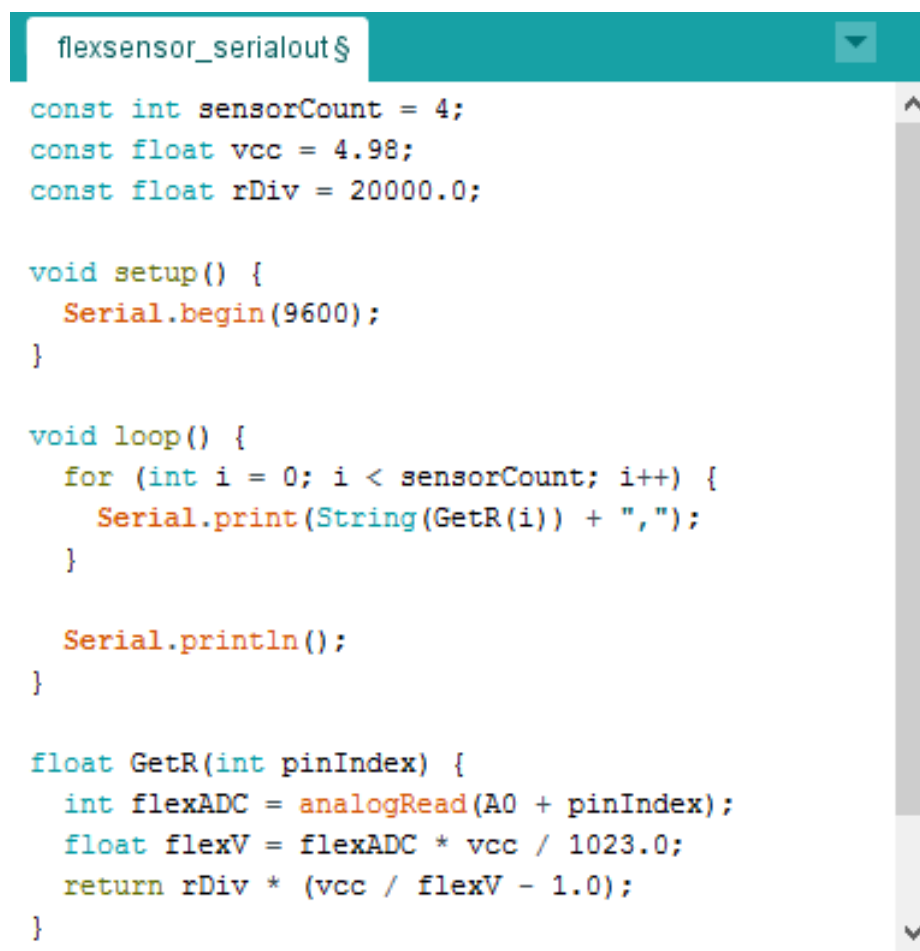
Design at this stage can remain relatively simple due to the small size and complexity of the code required to interface between an Arduino board and a computer. Consequently, the design in this section consist only of pseudo code, shown in Figure 3.24, and does not require a class diagram because C is not a class-based programming language, unlike its object-oriented counterpart, C++. This pseudo code details the process of creating a serial connection and converting the analog data provided to digital information. The digital output can then be used to determine resultant voltages by multiplying

the provided output by two constants, the initial 4.98V input provided by the Arduino and some constant value of 1023. The multiplication by 1023 is not entirely necessary although it should provide more accurate results after the next stage of the process. Finally, one more formula is used to convert this voltage to a resistance reading. This formula involves the division of potential voltage (4.98) by the previously calculated voltage value. This produces a value which represents the output voltage as a percentage of the original, and can be multiplied by the known resistance of the fixed resistor to provide the final resistance of the flex sensor. These formulas were taken from an online working solution (SparkFun, 2016).

3.4.1.2 Development

The designed pseudo code was implemented using the C programming language and saved within a 'ino' file, native to Arduino development. This meant that the program could be compiled and executed on any of the supported boards within the IDE, simply by uploading the program to any Arduino model connected via a USB cable. At this point of the development process, the model being used was the Arduino Uno which had no problems accepting the compiled program.

Some functional differences exist between the original pseudo code and the actual implemented solution, pictured in Figure 3.25. Namely, the connection is made within a 'setup' function, which only runs once after the Arduino receives power via USB cable. Additionally, the while loop shown within the pseudo code is replaced by the 'loop' function, which executes continuously whilst the Arduino is provided with power. These changes do not affect the underlying order of execution and simply show that the original assumption of all code running from one function is incorrect. Therefore, the code runs as expected when utilising the hardware artefact presented in Section 3.2.2, providing varying readings based on the current state of each connected sensor. These observations regarding functionality form the entirety of testing conducted at this stage, with the purpose of ensuring that both the basic hardware and software interfacing solutions work as intended.

A screenshot of the Arduino IDE interface. The title bar of the code editor window reads 'flexsensor_serialout\$'. The code is written in C and includes constant declarations for sensorCount, vcc, and rDiv. It defines a setup() function for serial communication and a loop() function that iterates through sensor readings. A GetR() function is also defined to calculate the resistance from the ADC reading.

```
const int sensorCount = 4;
const float vcc = 4.98;
const float rDiv = 20000.0;

void setup() {
  Serial.begin(9600);
}

void loop() {
  for (int i = 0; i < sensorCount; i++) {
    Serial.print(String(GetR(i)) + ",");
  }

  Serial.println();
}

float GetR(int pinIndex) {
  int flexADC = analogRead(A0 + pinIndex);
  float flexV = flexADC * vcc / 1023.0;
  return rDiv * (vcc / flexV - 1.0);
}
```

Figure 3.25 – Code written in C for creating a connection between the Arduino and a computer. This code is store and executed on the Arduino board.

3.4.2 Iteration 2 – Initial C# System

3.4.2.1 Design

In addition to the code implemented in Section 3.4.1, C# code is required to allow the Arduino to communicate with simulations developed within the Unity 3D Game Engine. This section talks about the design of such a system using a programming pattern known as ‘Adapter’, something that is conventionally used when trying to merge two existing systems together. The pattern is defined as a means of allowing an object with a foreign interface to cooperate with a new or existing system (Bahn and Jacobsen, 2002). This is precisely how the pattern will be applied in this iteration, performing the necessary conversions on resistance values, thus providing a clean frontend for the software interfacing solution.

A class diagram was created to facilitate the design stage of this implementation, seen in Figure 3.26. The diagram presents a design containing four classes, three of which will be used to create the wrapper whilst the final class, labelled ‘DataGloveController’ contains far less code and forms the top level of the interfacing solution. Specifically, the DataGloveController class will inherit from Unity’s existing ‘MonoBehaviour’ class which enables it to be attached as a Script to an object within the game scene. This is not the case with the other three classes which comprise of pure C# code and hold no association to the Unity code libraries, meaning that the classes are flexible and can be ported into any C# program in the future. In terms of class structure, both ‘ValueArray’ and ‘ResistorBounds’ will be encased within the ‘ArduinoInterface’ class, forming two sub classes with direct association to the core class. Specifically, these two sub classes will act as containers for resistance values whilst containing additional functions and validation measures, such as the Boolean value of “_ready” seen in the ResistorBounds class.

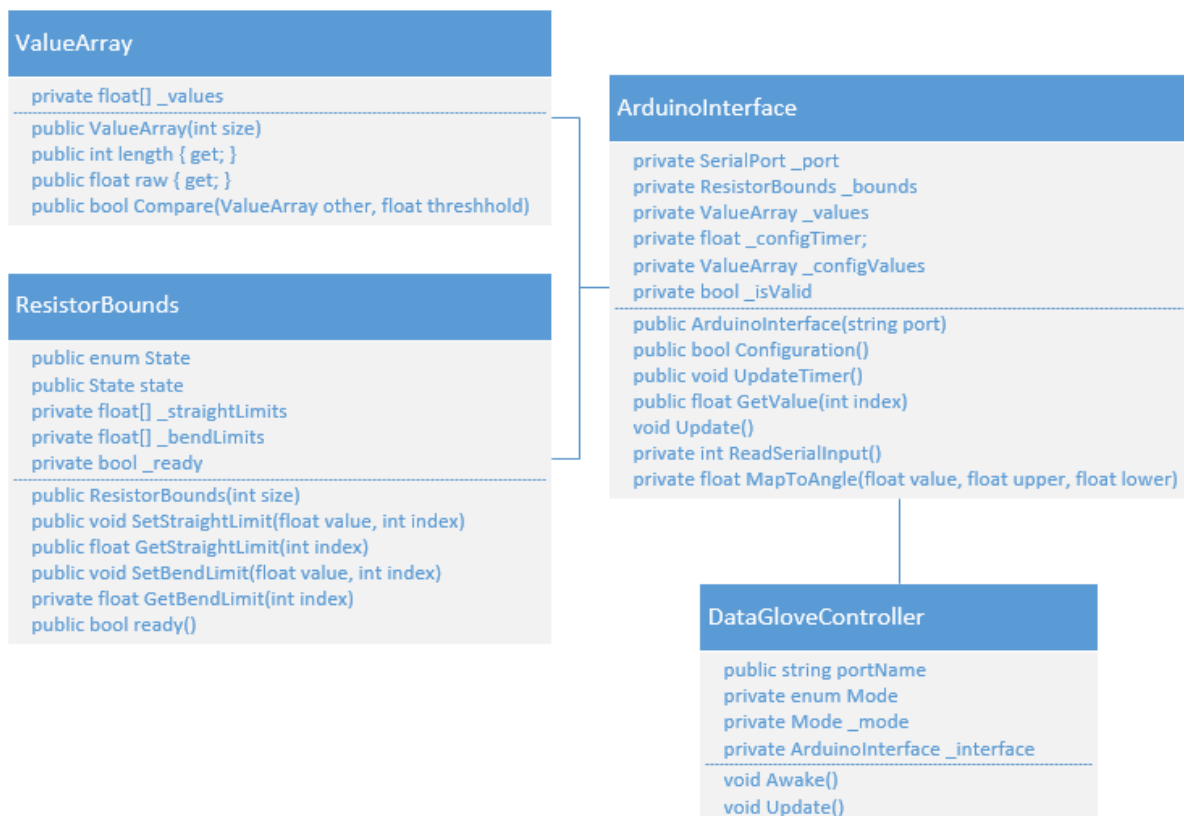


Figure 3.26 – Class diagram representing the association between classes for this interfacing solution.

In addition to the class diagram, the flowchart shown in Figure 3.27 details the intended execution for the interfacing software. The program will contain two runtime loops, as highlighted by the orange and green boxes around specific processes. The first loop is called the configuration loop and will encompass all necessary functionality for calibrating the hardware artefact. This will work by firstly checking if configuration is required, an option that will be presented to the user allowing them to decide if it is necessary. If this returns true, a configuration timer will be set to 0 ready for calibration to take place. The loop itself will continuously check for any large movements by the user and reset the timer if any such actions occur. However, if the user maintains a steady gesture long enough for the timer to expire, the current resistance values will be recorded across the glove and stored as one of the two limits to be used later. This will occur twice, allowing a state to be recorded for both an open and closed hand. Once the configuration process is complete or is otherwise shown to be unnecessary, the program will move onto the main loop. This loop consists of functions related to the reading and conversion of serial input data from the Arduino. Constantly updating the stored values as the user flexes their fingers and thus provides different resistance values. The calculations performed within this loop are reliant upon the correctly calibrated hardware artefact which is why the configuration loop is necessary.

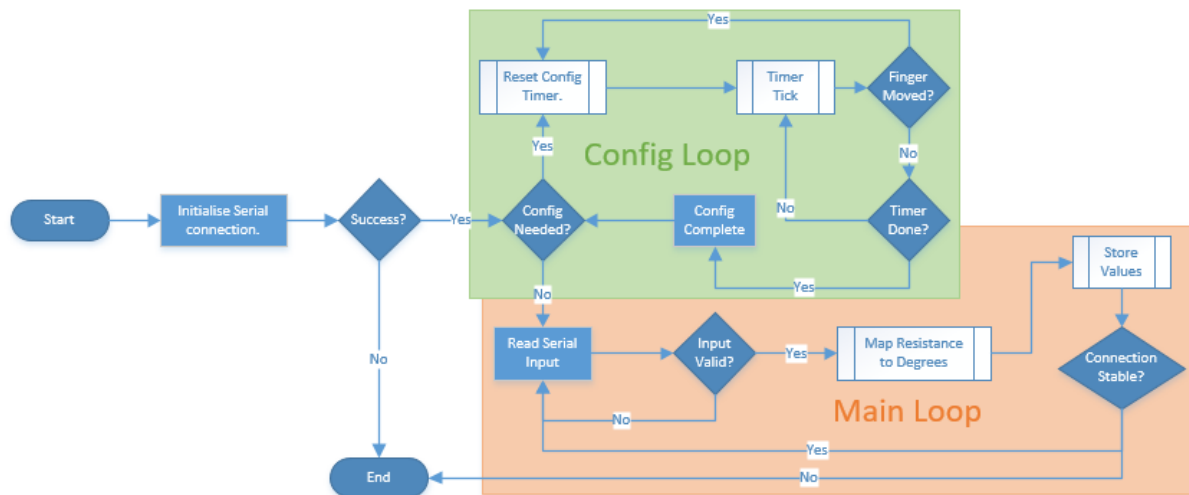


Figure 3.27 – Flowchart detailing the execution of software interfacing code.

3.4.2.2 Development

Whilst much of the development adhered to the suggested class structure for this iteration, several adjustments were made in the interest of writing clean and efficient code. The changes are detailed below and visually aided by code snippets taken from the source files.

- **ResistorBounds** – The ‘State’ enumerator proved to be far too cumbersome for its intended use, resulting in a highly impractical amount of conditional ‘if’ statements being used through the programming solution. This was replaced by unsigned integer flags, allowing a conclusion to be made regarding the current state of each ResistorBounds object based on a single integer comparison to the variables shown in Figure 3.28. Therefore, a value of ‘2’ implied that resistor limits for both an open and closed hand were provided.

```

12     public static readonly uint STRAIGHT_R_SET = 1;
13     public static readonly uint BEND_R_SET = 1;
14     public static readonly uint BOTH_R_SET = STRAIGHT_R_SET + BEND_R_SET;

```

Figure 3.28 – Code snippet for unsigned integer flags within the ResistorBounds class.

- **ResistorBounds** – Whilst writing the code for this element of the software interface, it quickly became clear that it would be better suited as a struct rather than a class. This was caused by a lack of functions unrelated to variable access and protection, meaning that the benefits provided by using a class were not being exploited by the current implementation.
- **ValueArray** – Several operator overload functions were added to this class, allowing two objects of this type to be compared to and assigned to each other. Specifically, the copy constructor was overwritten alongside the addition of a compare function for checking similarity between two ValueArray objects, as seen in Figure 3.29. This code snippet also shows two modified array access operators, allowing the array of values contained within the class to be accessed via the same syntax as it used for a standard array.

```
135 public bool HasVariance(ValueArray other, float threshold)
136 {
137     int size = Length;
138     if (other.Length < size)
139         size = other.Length;
140
141     for (uint i = 0; i < size; i++)
142     {
143         if (Mathf.Abs(_values[i] - other[i]) > threshold)
144             return true;
145     }
146
147     return false;
148 }
149
150 public float this[uint i]
151 {
152     get { return _values[i]; }
153     set { _values[i] = value; }
154 }
155
156 public float this[int i]
157 {
158     get { return _values[i]; }
159     set { _values[i] = value; }
160 }
161
162 public ValueArray(ValueArray other)
163 {
164     _values = new float[other.Length];
165     for (uint i = 0; i < _values.Length; i++)
166         _values[i] = other[i];
167 }
```

Figure 3.29 – Code snippet taken from source, showing the comparison function, array accessor functions and copy constructor override for the ValueArray class.

Despite these changes being made, the quantity of code written for the software interface began to grow exponentially, eventually leading to a poorly structured implementation. Consequently, the next iteration will involve refactoring all the code written in this section, armed with the newly acquired knowledge of exactly how the system should be structured to yield the most efficient and manageable code possible.

3.4.3 Iteration 3 – C# Refactoring

3.4.3.1 Design

After discovering issues with the previous implementation of the software interface, the iteration was abandoned in favour of re-writing the entire solution. To do this, a new class diagram was required, as shown in Figure 3.30, detailing a far more organised implementation which has been split into several additional classes. These classes will be much shorter than those previously used, adhering to the typical approach to development within Unity whereby a new class, or script, is created for every different functional component.

Additionally, the classes shown in Figure 3.30 are better integrated within the Unity ecosystem, making use of functions and data types exclusive to the Unity Engine libraries, such as ‘Transform’ and ‘Quaternion’. Consequently, far less will need to be done in terms of casting and converting variables, drastically reducing the amount of code to be written.

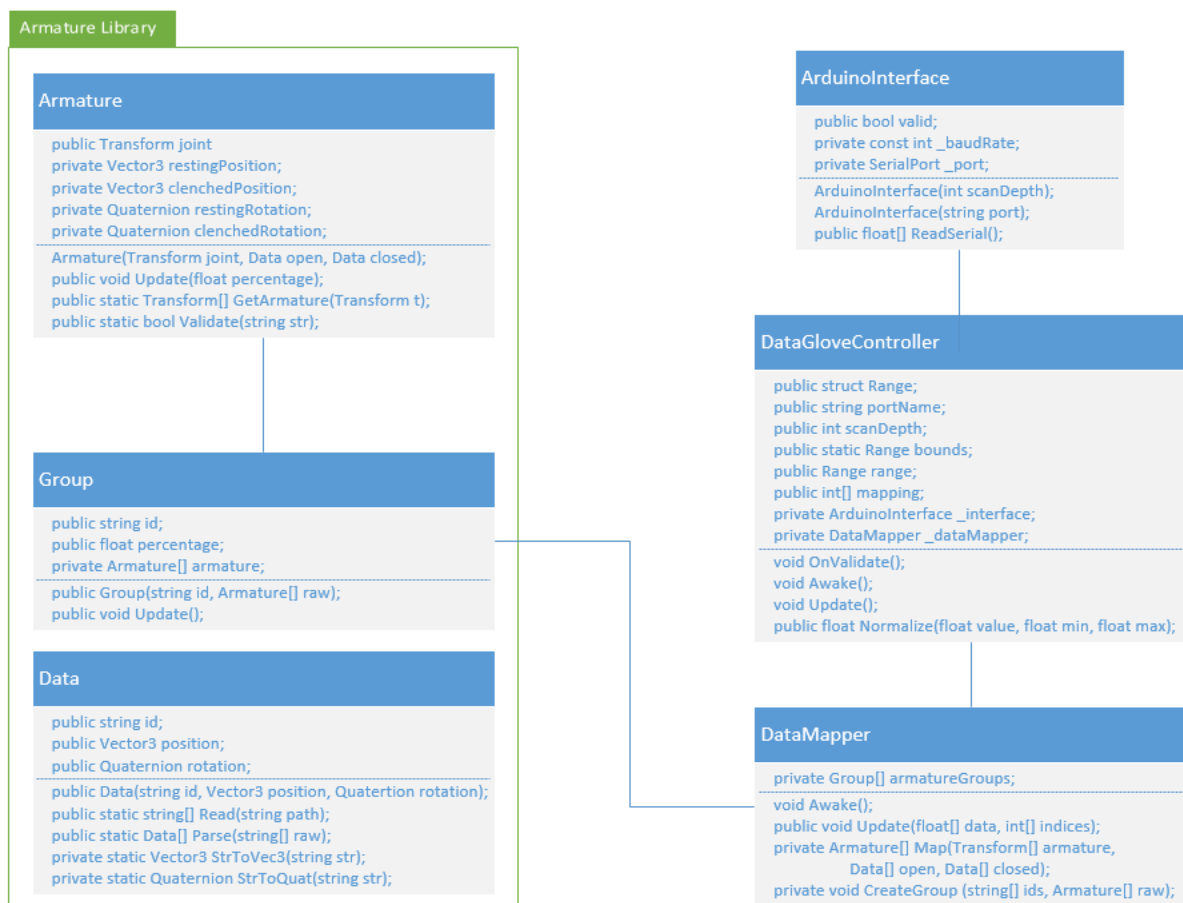


Figure 3.30 – Finalised class structure for the software interface.

A small library will be written to facilitate the manipulation of armature within the world space, something which was not implemented in the previous iteration because the solution was scrapped before reaching this stage of development. The library is necessary because Unity does not offer any armature manipulation tools by default, instead relying upon animations being completed within 3D modelling software, such as Blender, prior to importing the Asset. Pre-built animations are not suitable for the purposes of this project as the user must have individual control over each finger rather than the entire set. The library will consist of three classes, providing custom data

types in addition to a series of helper functions to support armature manipulation. The library will be capable of detecting a network of armature, referred to as a skeleton, and saving the state of each bone within the network with respect to position and orientation at any given time. With this foundation in place, animation will be able to occur by interpolating between two states, otherwise known as a “poses”, a technique employed by Unity’s in-built animator.

3.4.3.2 Development

As with the previous iteration, this development process did not stray far from the designed class diagram, implementing all of the suggested functions and variables. However, one function was added due to issues experienced when opening serial connections where the provided functions for checking if a connection is established were not producing the expected result. This function, shown in Figure 3.31, simply encased the opening of the port within a try-catch statement, ensuring that any errors were caught and communicated across the rest of the program at runtime. The function also helped to catch any errors if the serial connection was interrupted, preventing error messages from being output to the console and thus eliminating lag associated with excessive error outputs in Unity. The reduced frame rate would otherwise potentially induce motion sickness.

```
55 private bool OpenPort(string name, int baudRate, out SerialPort port, bool log = false)
56 {
57     port = new SerialPort(name, baudRate);
58
59     try { _port.Open(); }
60     catch (Exception e)
61     {
62         if (log)
63             Debug.LogError(e.Message);
64         return false;
65     }
66
67     return true;
68 }
```

Figure 3.31 – Function created for handling errors when opening a serial connection.

In addition to creating the artefact library, a 3D hand model was built and rigged within Blender, allowing each finger to be independently controlled. The process of doing this involved vertex manipulation to build up a realistic hand shape, before creating an armature skeleton and painting vertex weights across the surface of the 3D model. Vertex painting is a way of visualising how much influence each bone within the armature skeleton will have over a specific part of the mesh. Once finished, the 3D model was exported into Unity where each individual bone could be manipulated freely within the scene. The armature library exploits this ability, simply recording the ‘Transform’ information of each bone during a given pose and using that to produce controllable animations.

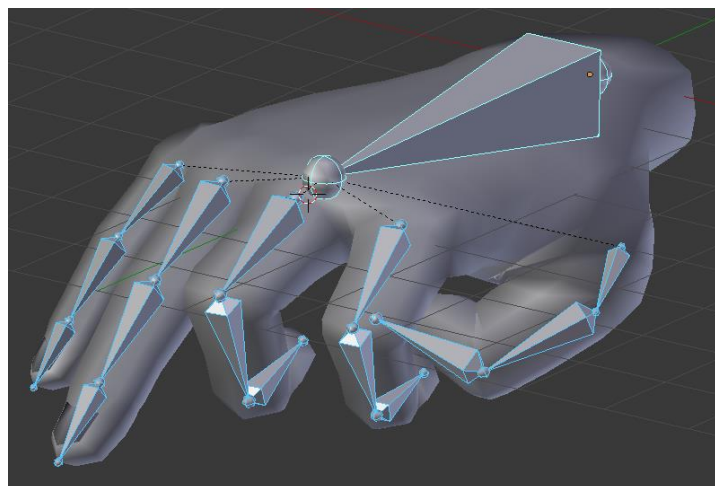


Figure 3.32 – 3D model of a human hand, built and rigged within Blender.

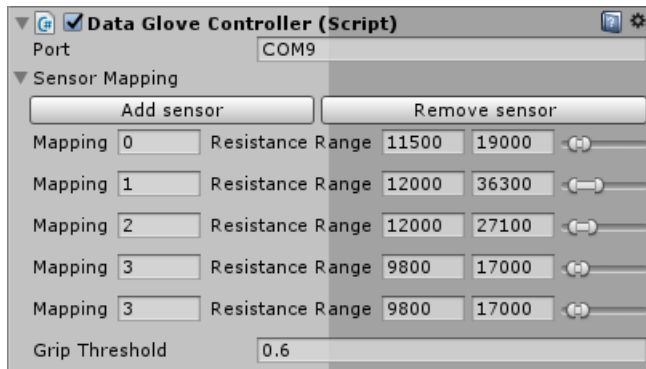


Figure 3.33 – Custom Editor used for calibrating the Data Glove within the Unity editor.

to calibrate the artefact, including the ability to map each sensor to any number of fingers, as seen in the last two mapping fields each showing '3' as their target sensor. However, testing this panel led to the conclusion that it is rather counter intuitive, and a solution which uses the sensors themselves to calibrate the glove would be more ideal. This is discussed further in Section 4.2.

Unlike the previous iteration, the class design pictured in Figure 3.30 does not account for any interactive calibration for the Data Glove, meaning that an alternative must be implemented to ensure that the glove provides the desired results. Figure 3.33 pictures a solution to this issue in the form of a custom inspector panel within the Unity editor, providing precision control over the resistance limits for the glove. This provides all the necessary tools needed

3.4.4 Testing

3.5 Scenario Software Development

3.5.1 Iteration 1 – Initial Scene

3.5.1.1 Design

This design process describes the setup of a scene within Unity using two different VR packages, namely Steam VR and Newton VR, which achieve the same functionality with a few differences. Fundamentally, Newton VR offers a more refined interaction experience than that of Steam VR. It is bundled with a wide variety of interaction scripts to suit most situations using the standard HTC Vive controller, vastly increasing the speed in which a working scene can be implemented. For these reasons, Newton VR has been chosen to use in the scene concerned with testing the usability of the HTC Vive controller. However, the enhanced interaction scripts would offer no advantages when used in the Data Glove scene as they are written for specific use on the HTC Vive controller and would require significant modification before they would work on the in-scene hand model. Hence using the basic Steam VR package for the scene concerned with testing the project artefact.

Having determined the tools which will be used to allow player control within each scene, the next step is to design the layout of the scene with regards to lighting, walls and other obstacles. Although a large portion of this will take place in the next iteration which focuses on adding research critical objects to the scene. Therefore, there is a lot of room for flexibility when creating the initial scene in this iteration. The only concern at this stage is to ensure that both scenes are synonymous with each other, meaning that the floor, walls, ceiling and lighting should not change in either scenario.

3.5.1.2 Development

Two identical rooms were created within Unity to satisfy the requirements for this iteration. The decision to use a room rather than an open space was a product of two thought processes. Firstly, a room minimises the chance of the participant becoming distracted by the vastness of the world they are placed into, ensuring that they focus on the task and complete it in a timely manner. Secondly, the room closely represented the dimensions of the VR learning space that would be used in the demonstrations, creating a more natural feel to the environment. The room is shown in Figure 3.34, where all of the front most walls are semi-transparent for demonstration purposes.

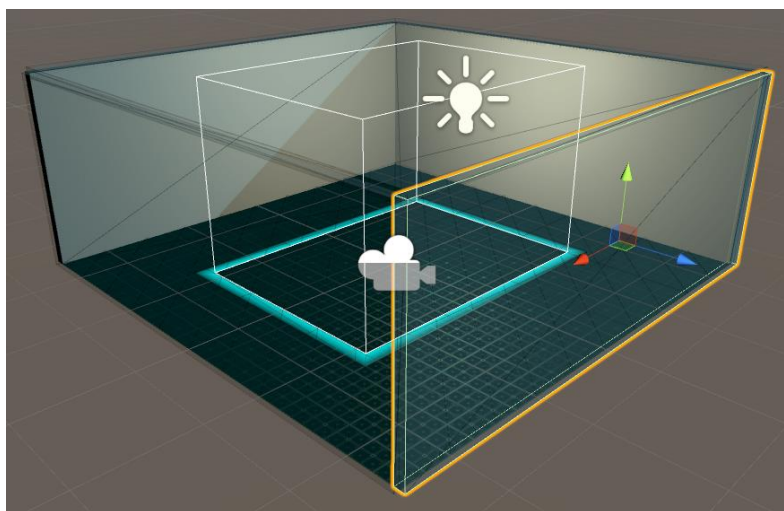


Figure 3.34 – A VR room designed for the participant testing process.

3.5.2 Iteration 2 – Real-World Scaling

3.5.2.1 Design

To create scenarios which would produce meaningful results, several objects needed to be chosen which would support a range of interactions. Firstly, object should be able to be picked up and thrown within the VR environment, allowing the user to get a feel for how easily they can perform basic actions in VR. Secondly, objects should be added which require an element of precision to interact with. This could be in the form of a calculator or keyboard, but must be suitable for both the HTC Vive controller and the Data Glove whilst retaining its dimensions in both tests. These aims should be relatively easy to accomplish. However, the third testing scenario requires virtual objects to be scale 1-to-1 with their real world counterparts, creating an increased level of difficulty for the modelling process. The number of models produced should not matter, provided that there is at least one for each type of interaction, although redundancy would be useful to ensure that the simulation can continue much longer should the user lose a virtual object.

Due to the use of real world objects, design is not a particularly large part of this iteration, simply outlining a set of rules to be followed to create valid testing scenarios.

3.5.2.2 Development

Developing the models for this implementation was particularly tricky but by using the scaling tools within Blender, as done previously when creating the Arduino casing, the process was completed relatively quickly. The dimensions of each real world object were acquired using a ruler and simply applied to meshes within Blender to provide the 1-to-1 scaling. A tumbler, can and mug were created to represent all of the real world objects used. Additionally, a large scale typing pad was created for use in VR, although this was fictional and had no real world counterpart. These objects were placed on a table in both Unity scenes, as shown in Figure 3.35.



Figure 3.35 – The objects used in research testing to assess differences in interaction between the HTC Vive and Data Glove.

Figure 3.36 presents the completed scene with the front most walls once again shown as partially transparent. This scene is duplicated for use with both the Newton VR and Steam VR, to ensure that both tests take place under equal circumstances. When standing in the scene, everything seemed to be the correct size, however minor adjustments may be necessary when switching over to the VR learning space for research testing. One virtual object which may cause some problems is the table, which was not measured but instead modelled from side, front and top view reference images. This may mean that the table is not the correct width, height or depth. This will however become clear once the project is moved into the VR learning space.

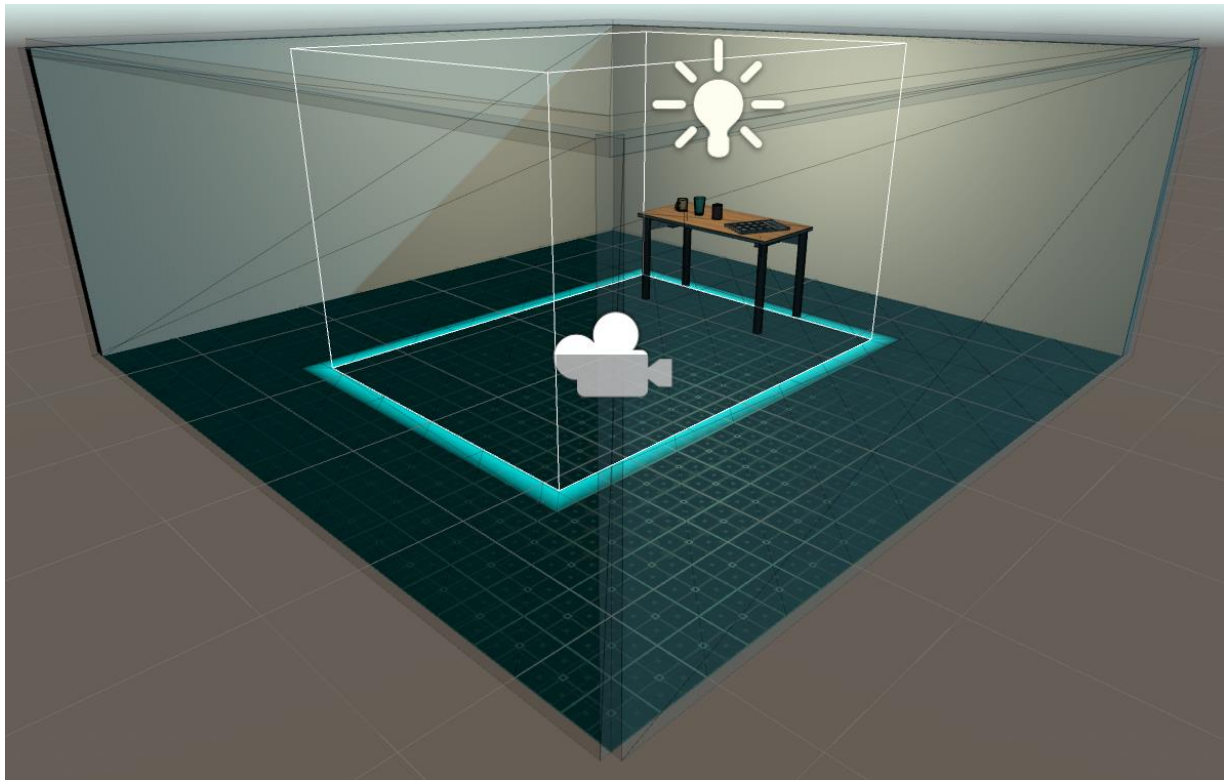


Figure 3.36 – The finalised scene to be used in research testing.

3.5.3 Further Maintenance

Once the project was loaded into the VR learning space, in preparation for research testing, each object had to be aligned with their real world counterpart, including the table. At this point, the table required scaling down in all axis, but soon fit to the exact dimensions of its real world counterpart. Each of the objects were the correct size and simply required lining up on the table.

After these minor adjustments, the artefact was ready to begin research testing. However, as the tests went on, the table and objects started to become misaligned, requiring constant adjustment throughout the day. The cause for this is unknown but may be due to the fact that two HTC Vive devices occupy the VR learning space, perhaps causing infrared interference. Nevertheless, testing continued and the use of masking tape across the floor and top of the table, shown in Figure 3.37, proved to be incredibly useful between participants.



Figure 3.37 – The use of masking tape during the user testing process.

3.6 Research

3.6.1 Participant Recruitment

In general, the participant recruitment process is considered to be the most difficult part of any research process (Ferland and Fortin, 1999; Walson, 1999; Chang et al., 2004) with recent estimates suggesting that 85% of trials do not form a conclusion on schedule due to low participant accrual (Blanton et al., 2006). However, this study presents a fairly lenient participant recruitment criteria meaning that a wider range of individuals are suitable to engage with the user testing process. Specifically, the only recruitment criteria was that the group should contain a mix of varying experiencing regarding VR, accounting for those who have never used the technology before in addition to those who have specifically used the HTC Vive in the past. This was done to review the differences in results between those who have experienced VR technology already and those who have not.

For the purposes of this study, the difference in results with relation to gender and age was not measured and the associated participant information asked for on the questionnaire will simply be used in the interest of gauging the demographic information of participants.

With these criteria, the goal is to engage with at least 15 participants, but more would be ideal to improve the reliability of results.

3.6.2 Ethical Procedures

Before allowing participants to engage in the study they were presented with information about the study, the risks associated with participating in VR related research and an assurance that their personal information would remain anonymous. They were asked to read the form and tick several boxes to confirm that they have understood the information presented to them, before signing to provide their consent to participate in the study. In the interest on anonymity, only one consent form has been provided to evidence their use, with the name and signature of the participant blurred to protect their identity. This consent form has been scanned and provided as Figure B.1.

3.6.3 Design

A questionnaire will be produced in order to gather the results of each testing scenario, and will consist mostly of Likert scales to measure the level of interaction experienced by each participant during the study. The reason for this is that interaction can be easily quantified across a specified scale, simply requiring the participant to rank how easy it was to interact with their environment in both the HTC Vive and Data Glove scenarios. Likert scales are capable of producing objective data, typically consisting of a scale between 1 and 5, allowing the user to answer anywhere between strongly disagreeing and strongly agreeing (Gliem and Gliem, 2003) with a provided statement. Interestingly, Likert scales are very flexible and allow 4, 5 or 7 points along its axis, depending on the desired result of the study. For the purposes of this project, a 5-point scale was used to allow participants to provide neutral responses to each question, indicating that they do not prefer either the HTC Vive controller or the Data Glove. The numerical answers provided by each participant will then be used to determine the mean quality of interaction for each device.

With appropriately worded statements, the Likert scale can also be used to understand the level of immersion experienced by each participant, although this statistic is likely to be influenced greatly by a number of variables. Firstly, users who are more prone to motion sickness will be more likely to experience nausea when participating in the experiment. The individual will then experience discomfort which impacts upon the immersive experience, as previously discussed in Section 2. Secondly, immersion is heavily influenced by external distractions such as sound and movement, which means that any unforeseen noises could interfere with the immersive experience. These variables create difficulty when attempting to gauge immersion, regardless of the method of data analysis that is used. For this reason, steps must be taken to minimise the risk of interference and motion sickness within the simulation.

In addition to the vast number Likert scales used, some qualitative analysis will take place using text fields which will allow participants to express their thoughts regarding a particular part of the experiment. For example, when asked about difficulties during interaction, the participant will be asked to specifically name areas of the experiment that proved most challenging. The data gathered from these questions will be used to more accurately pinpoint areas where interaction was particularly lacking, allowing comparisons to be made between different types of interaction. This will produce subjective information about the experiences of each participant during the study, although common answers may be considered to be factual rather than being based on personal thoughts of an individual, forming objective data.

The design considerations described in this section resulted in the production of the questionnaire shown in Figure B.2, created using Google Forms.

3.6.4 Study Procedure

When conducting the study, participants were directed to a VR learning space located at the University of Lincoln, and asked to read and complete the aforementioned consent form. Once the participant had consented to the study, they were given a short verbal briefing of each of the three scenarios that would take place. The participant would be observed during each scenario in the interest of safety and to witness any difficulties during interaction which could help to identify areas for improvement regarding the Data Glove.

The first scenario tasked the user with interacting with four different objects within VR, using the HTC Vive controller. The scene was setup to cover a number of interactions, including picking up and throwing an object in addition to precise typing on a 3D modelled pad. The second scene mimicked the first, however it would allow the participants to use the Data Glove for interaction instead of the HTC Vive controller. This formed the basis for the second scenario, where the user was simply asked to once again interact freely until they were finished. The final scenario repeated the second but introduced real life physical objects which were of the same scale as the 3D models. Tape was used to mark the exact location of where each object should be placed in every test, ensuring consistency between participants.

After completing all three scenarios, the participant would be asked to fill out a questionnaire, shown in Figure B.2, in order to provide feedback. This feedback was particularly important to forming a research conclusion, so participants were asked to consider each answer carefully before submitting the questionnaire.

3.6.5 Results Analysis

Of the 20 participants involved in the study, only 4 had no prior experience using VR devices, whilst 16 participants had engaged with a type of VR platform, 14 of which had used the HTC Vive. This results in a ratio of 1:5 participants who began the test with no previous experience using VR systems, which is not ideal for identifying variance in ability between experienced and inexperienced users when operating the Data Glove. As a consequence of this significantly disproportionate ratio, these statistics were not factored into the final conclusion, ignoring all the investigations into whether or not previous VR experience has an effect on the participant's ability to use the Data Glove.

On the following page are the results of each Likert scale used in the questionnaire with their means represented the dark blue line on each graph. The result is to be considered insignificant at any instance where the mean value is between 2.75 and 3.25, as this would represent a statistically neutral mean.

"It was more difficult to interact with objects using the Data Glove rather than the HTC Vive controller."

(20 responses)

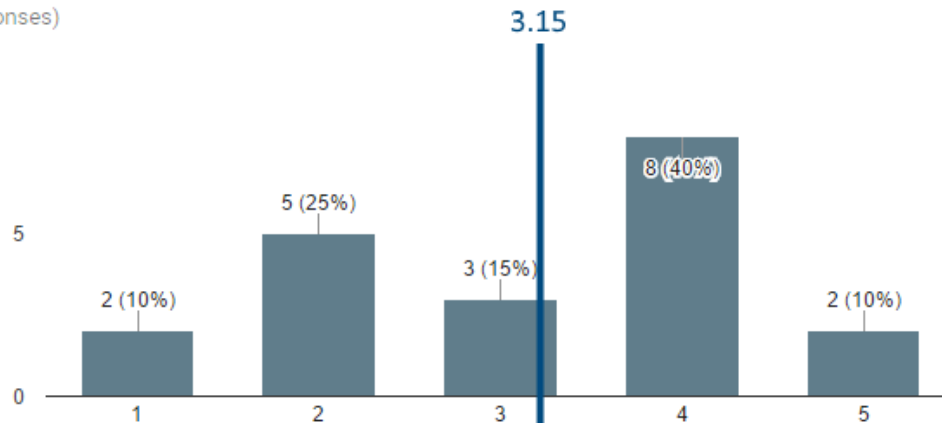


Figure 3.38 – Results regard the difficulty when interacting with the Data Glove. Mean: 3.15.

In Figure 3.38, the mean of the results was 3.15, it is considered to be insignificant. However, it can be noted that participants did not favour either device with regards to interaction, finding them both equally as difficult to use within the VR environment. Unfortunately, these results may have been effected by a lack of clarity in the provided statement. This is because the statement does not distinguish the second and third scenario. This may prove to be significant as 16 participants said that some interactions were particularly challenging when using the data glove. Of which, 7 explained that the physical objects were the cause of interaction difficulties. However, upon inspecting the individual questionnaire submissions, only 2 of the 7 participants ranked the data glove as proving worse interaction that the HTC Vive controller. This relinquishes the concern with regards to statement clarity and means that the result remains insignificant.

"The presence of real world objects improved interaction." (20 responses)

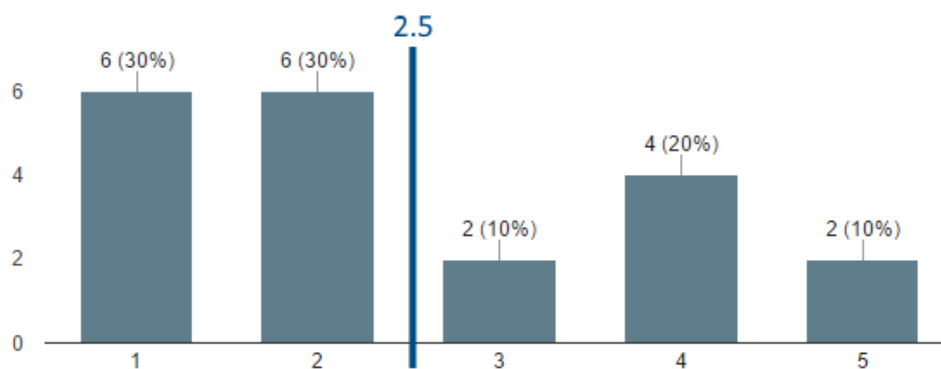


Figure 3.39 – Results assessing whether real world objects improve interaction. Mean: 2.5.

In Figure 3.39, the mean of the results is 2.5. This falls just outside of the range of insignificant results, suggesting that real world objects did not offer any benefit to interaction and were instead a hindrance. This is likely due to the constant issue of real-world objects moving away from their virtual counterparts, creating confusion. However, the table used was not prone to this issue as it was heavily enough to remain stationary during each testing process. With additional time, another test could be run to analyse the effectiveness of static real-world objects rather than the lightweight objects used in this project. This may provide conclusions which indicate the benefits of static physical objects in conjunction with VR.

The mean of 4 for the results shown in Figure 3.40 provides the most conclusive result of the entire research process. This result suggests that participants categorically felt that immersion was improved when using the Data Glove in comparison to the HTC Vive controller. A conclusion can therefore be made that despite the lack of improved interaction, the use of finger motion tracking does indeed provide superior levels of immersion when using VR. However, the use of physical objects which can move within real world space does not offer any benefits in terms of immersion. This is further evidenced by Figure 3.41 which shows an insignificant result regarding the impact of physical objects on immersion. The significant number of negative comments made towards the third scenario provide further justification for this conclusion.

"Using the Data Glove (without physical objects) made me feel more immersed when compared to the HTC Vive controller."

(20 responses)

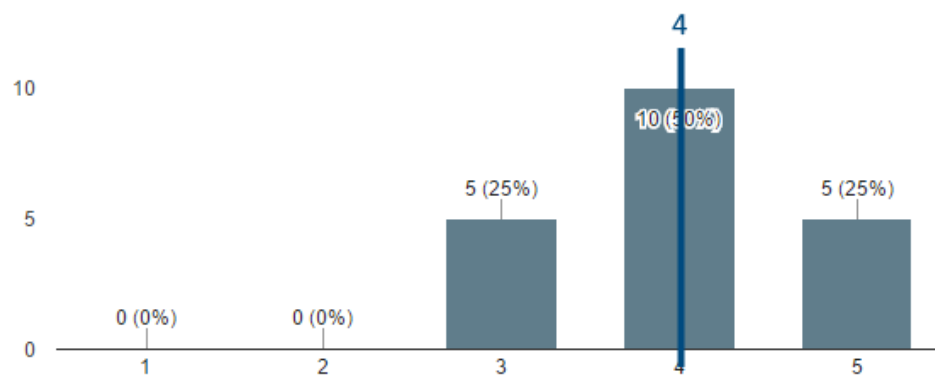


Figure 3.40 – Results of Data Glove immersion compared to the HTC Vive controller. Mean: 4.

"The presence of real world objects improved immersion." (20 responses)

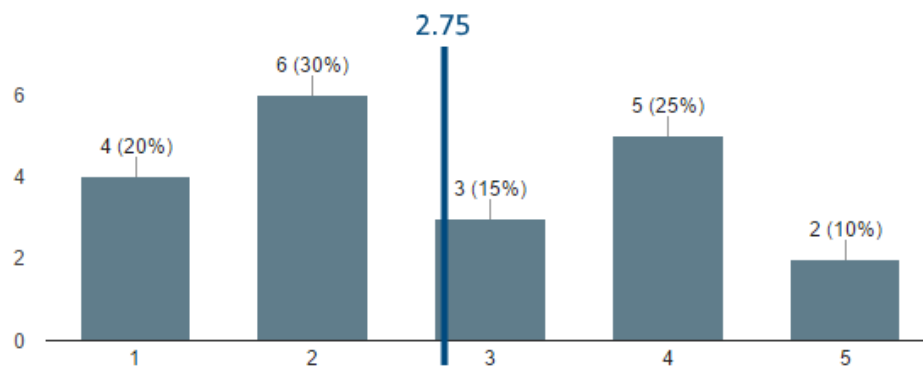


Figure 3.41 – The resultant impact of real world objects on immersion. Mean: 2.75.

Lastly, it was found that the physical presence of the data glove did not subtract from the experience, scoring a mean of 2.1 in Figure 3.42. This is particularly unexpected as predictions would dictate that the cumbersome nature of the attached HTC Vive controller would restrict the wearer's movement, resulting in a worse experience. However, the participants clearly felt that this was not the case, strongly disagreeing with the statement presented in the questionnaire.

"The physical presence of the Data Glove device detracted from the experience."

(20 responses)

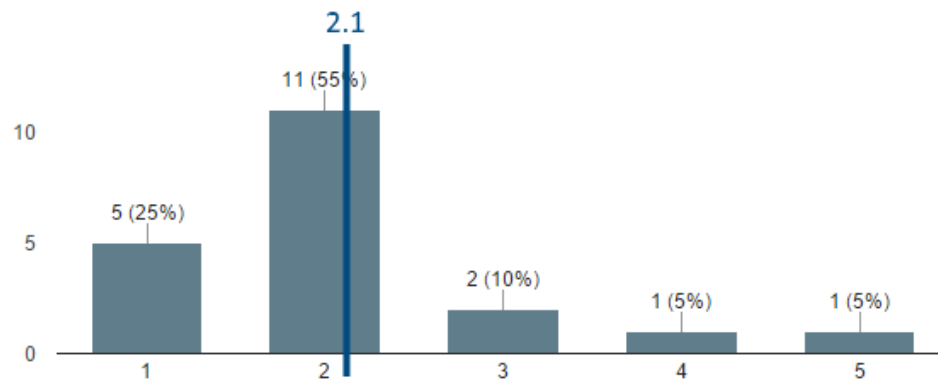


Figure 3.42 – Whether the physical presence of the glove detracted from the experience. Mean: 2.1.

4 Reflective Analysis

4.1 Critical Reflection

I strongly believe that the chosen methodology played a large part in the timely completion of the project. It worked well around other assignment deadlines and also allowed development stages to be executed a way that made more sense than the previously suggested Gantt chart. I think that if I were to start the project again, I would have specified the smaller iterations within the original Gantt chart to cut down on delays at the start of the project.

I am incredibly pleased with the final artefact as the development process challenged me to quickly learn a series of unfamiliar skills, especially with regards to the electronics. Having no prior experience dealing with electronics was incredibly daunting at the start of the project, however, I became increasingly more confident as the project progressed. The artefact as a whole has exceeded my original expectations and provided all of the necessary functionality to make the project a success. However, I believe that more time could have been spent researching the electronics prior to development which may have allowed the process to finish at an earlier stage and require less maintenance towards the end of the project. This would have also allowed some user testing to take place prior the research stage. An advantage of this would be that calibration issues could have been identified at an earlier stage of the development process, rather than allowing them to surface during the research phase.

With regards to the research processed within the project, I feel that the results were heavily influenced by the calibration issues which may unfortunately deem the research conclusion invalid. Given 20-20 hindsight, I would have made interactive calibration a priority to avoid these issues and retain the validity of the study. However, I feel that the main conclusion made, regarding the impact of finger motion tracking on immersion, maintains its validity regardless of the calibration issues. The reason for this is that calibration issues would only sway results in favour of the HTC Vive controller, meaning that the Data Glove providing higher immersion despite the issues.

In conclusion, I feel that a better initial project plan could have resulted in a smoother project execution, with far fewer issues experienced at the research stage. However, the project has been invaluable to me personally as I have learnt valuable skills that the Games Computing course would have otherwise not taught me. Therefore, in terms of the finished artefact, main research conclusion and personal gain, I feel that the project was a success.

5 References

1. Aleesh Sankaran (2012) Of the fingers in the human hand, why is the ring finger the most difficult to control? [online] Available from <https://www.quora.com/Of-the-fingers-in-the-human-hand-why-is-the-ring-finger-the-most-difficult-to-control> [Accessed 26 April 2017].
2. Allison, R.S., Harris, L.R., Jenkin, M., Jasiobedzka, U. and Zacher, J.E. (2001). Tolerance of temporal delay in virtual environments. In: Virtual Reality, Tokyo, Japan, 13-17 March New York, USA: IEEE, 247-254.
3. Amir, M.H., Quek, A., Sulaiman, N.R.B. and See, J. (2016) DUKE: Enhancing Virtual Reality based FPS Game with Full-body Interactions. In: Advances in Computer Entertainment Technology, Osaka, Japan, 9-12 November New York, USA: ACM, 35.
4. Anon (2016a) immersion – definition of immersion in English. [online] Available from <https://en.oxforddictionaries.com/definition/immersion> [Accessed 13 April 2017].
5. Anon (2016b) presence – definition of presence in English. [online] Available from <https://en.oxforddictionaries.com/definition/presence> [Accessed 13 April 2017].
6. Arkenbout, E.A., de Winter, J.C. and Breedveld, P. (2015) Robust Hand Motion Tracking through Data Fusion of 5DT Data Glove and Nimble VR Kinect Camera Measurements. *Sensors*, 15(12), 31644-31671.
7. Autodesk (2016) IEC-60617 Symbol Preview. [online] Available from <https://knowledge.autodesk.com/support/autocad-electrical/learn-explore/caas/CloudHelp/cloudhelp/2016/ENU/AutoCAD-Electrical/files/GUID-7871E6EF-24D5-467E-9B74-321FEDC9DFDA-htm.html> [Accessed 21 April 2017].
8. Bahn, S.R. and Jacobsen, K.W. (2002) An object-oriented scripting interface to a legacy electronic structure code. *Computing in Science & Engineering*, 4(3), 56-66.
9. Balaji, S. and Murugaiyan, M.S. (2012) Waterfall vs. V-Model vs. Agile: A comparative study on SDLC. *International Journal of Information Technology and Business Management*, 2(1), 26-30.
10. Barfield, W., Zeltzer, D., Sheridan, T. and Slater, M. (1995) Presence and performance within virtual environments. In: Virtual environments and advanced interface design. New York, USA: ACM, 473-513.
11. Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R. and Kern, J. (2001) Manifesto for agile software development. [online] Available from <http://nlp.chonbuk.ac.kr/SE/ch05.pdf> [Accessed 18 April 2017].
12. Benson, E., Breen, J., Knapp, C., Halloran, S. and Han, A. (2015) Hikari Hook: Game Development with the Oculus Rift and Handheld Motion Controllers. PhD. Worcester Polytechnic Institute. Available from <https://web.wpi.edu/Pubs/E-project/Available/E-project-110815-212712/unrestricted/OsakaMQP15Paper.pdf> [Accessed 12 April 2017].

13. Beynon-Davies, P., Carne, C., Mackay, H. and Tudhope, D. (1999) Rapid application development (RAD): an empirical review. *European Journal of Information Systems*, 8(3), 211-223.
14. Blanton, S., Morris, D.M., Prettyman, M.G. and McCulloch, K. (2006) Lessons learned in participant recruitment and retention: the EXCITE trial. *Physical Therapy*, 86(11), 1520.
15. Blender Foundation (2002) Blender. [software] Amsterdam, Netherlands: Blender Foundation. Available from: <https://www.blender.org/foundation/> [Accessed 12 April 2017].
16. Brown, D., Renney, N., Stark, A., Nash, C. and Mitchell, T. (2016) Leimu: Gloveless music interaction using a wrist mounted leap motion. In: *New Interfaces for Musical Expression*, Brisbane, Australia, 11-15 July, 3-4.
17. Brown, E. and Cairns, P. (2004) A grounded investigation of game immersion. In: *Human factors in computing systems*, Vienna, Austria, 24-29 April New York, USA: ACM, 1297-1300.
18. Carmel, E., Espinosa, J.A. and Dubinsky, Y. (2010) "Follow the Sun" Workflow in Global Software Development. *Journal of Management Information Systems*, 27(1), 17-38.
19. Chang, B.H., Hendricks, A.M., Slawsky, M.T. and Locastro, J.S. (2004) Patient recruitment to a randomized clinical trial of behavioral therapy for chronic heart failure. *BMC Medical Research Methodology*, 4(1), 8.
20. Corradini, A. and Cohen, P.R. (2002) Multimodal speech-gesture interface for handfree painting on a virtual paper using partial recurrent neural networks as gesture recognizer. In: *Neural Networks*, Honolulu, USA, 12-17 May New York, USA: IEEE, 2293-2298.
21. Csikszentmihalyi, M. (ed.) (1996) *Flow and the psychology of discovery and invention*. New York, USA: Harper Collins.
22. Dorfmüller-Ulhaas, K. and Schmalstieg, D. (2001) Finger tracking for interaction in augmented environments. In: *Augmented Reality*, New York, USA, 29-30 October New York, USA: IEEE, 55-64.
23. Erol, A., Bebis, G., Nicolescu, M., Boyle, R.D. and Twombly, X. (2007) Vision-based hand pose estimation: A review. *Computer Vision and Image Understanding*, 108(1), 52-73.
24. Eutsler, B., Gaertner, J., Pollino, E., Robinson, N., Moges, M., Tadimetri, S. and Yerrabandi, K. (2015) Virtual Reality Glove. In: *Gulf-Southwest Annual Conference*, Texas, USA, 6-8 March Washington, USA: ASEE.
25. Ferland, D. and Fortin, P.R. (1999) Recruitment strategies in superiority trials in SLE: lessons from the study of methotrexate in lupus erythematosus (SMILE). *Lupus*, 8(8), 606-611.
26. Fittkau, F., Krause, A. and Hasselbring, W. (2015) Exploring software cities in virtual reality. In: *Software Visualization*, Bremen, Germany, 27-28 September New York, USA: IEEE, 130-134.
27. Fowler, M. (2001) The new methodology. *Wuhan University Journal of Natural Sciences*, 6(1), 12-24.

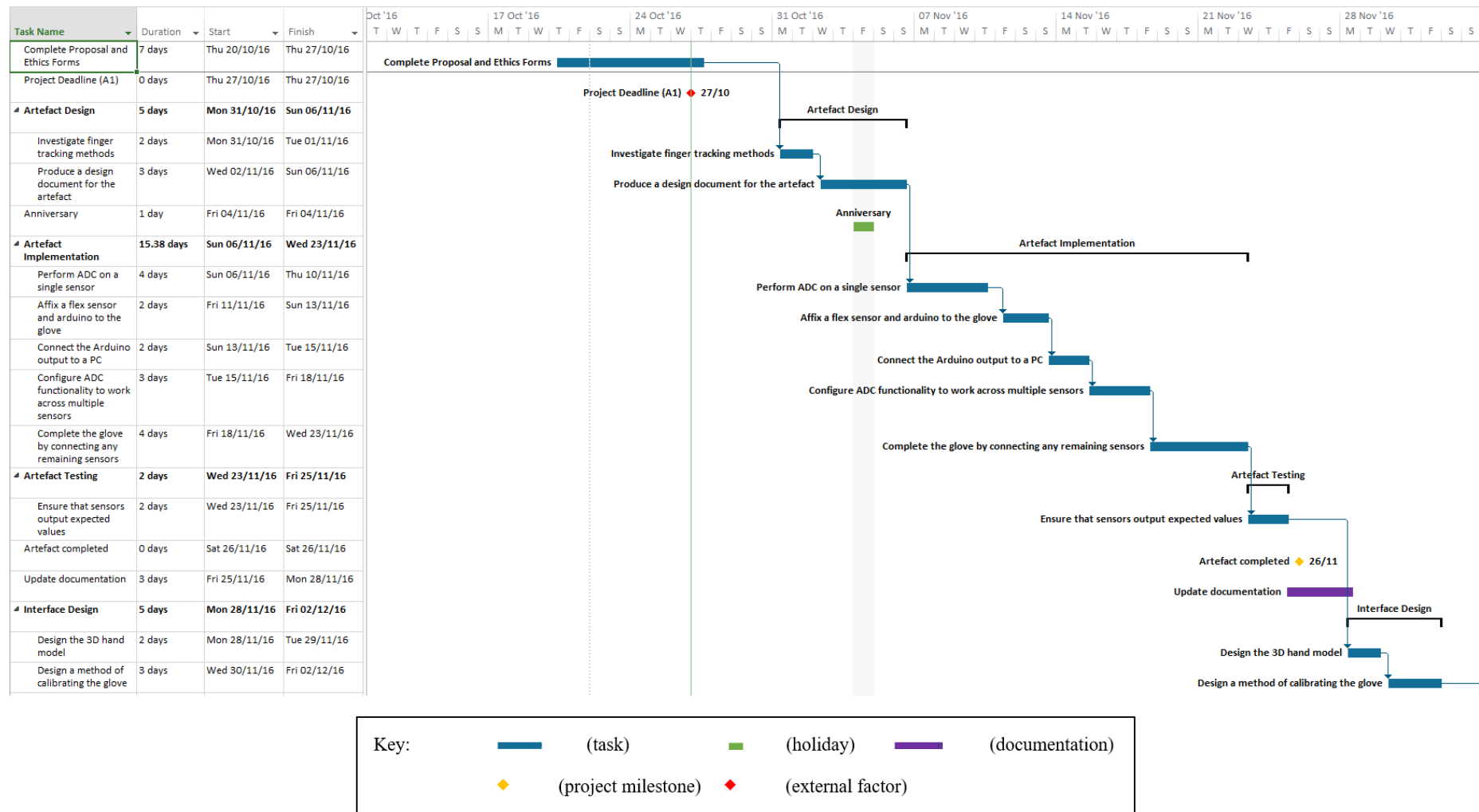
28. Fowler, M. and Highsmith, J. (2001) The agile manifesto. *Software Development*, 9(8), 28-35.
29. Game Spot (2015) Oculus Reveals Its Unusual Rift VR Controller, Oculus Touch. [online] Available from <https://www.gamespot.com/articles/oculus-reveals-its-unusual-rift-vr-controller-ocul/1100-6428010/> [Accessed April 14 2017].
30. Gander, P. (1999) Two myths about immersion in new storytelling media. [online] Available from http://www.pierregander.com/research/two_myths_about_immersion.pdf [Accessed April 13 2017].
31. Gliem, J.A. and Gliem, R.R. (2003) Calculating, interpreting, and reporting Cronbach's alpha reliability coefficient for Likert-type scales. In: *Midwest Research-to-Practice Conference in Adult, Continuing, and Community Education*.
32. Guide, P.M.B.O.K. (ed.) (2004) A guide to the project management body of knowledge. A Guide to the Project Management Book of Knowledge. Pennsylvania, USA: Project Management Institute.
33. Gustavsson, T. and Rönnlund, P. (2010) Agile project management in public events. In: *IPMA World Congress, Istanbul, Turkey, 1-3 November Amsterdam, Netherlands: IPMA*.
34. Hoffman, H.G. (2004) Virtual-reality therapy. *Scientific American*, 291, 58-65.
35. Howard, A. (2002) Rapid Application Development: rough and dirty or value-for-money engineering?. *Communications of the ACM*, 45(10), 27-29.
36. Howarth, P.A. and Costello, P.J. (1997) The occurrence of virtual simulation sickness symptoms when an HMD was used as a personal viewing system. *Displays*, 18(2), 107-116.
37. Highsmith, J. and Cockburn, A. (2001) Agile software development: The business of innovation. *Computer*, 34(9), 120-127.
38. I-Illusions (2016) Space Pirate Trainer. [online game] Brussels, Belgium: I-Illusions. Available from <http://www.spacepiratetrainer.com/#spthomepage> [Accessed 12 April 2017].
39. Jankowski, J. and Decker, S. (2012) A dual-mode user interface for accessing 3D content on the world wide web. In: *World Wide Web, Lyon, France, 16-20 April New York, USA: ACM*, 1047-1056.
40. Jennett, C., Cox, A.L., Cairns, P., Dhoparee, S., Epps, A., Tijs, T. and Walton, A. (2008) Measuring and defining the experience of immersion in games. *International journal of human-computer studies*, 66(9), 641-661.
41. Jones, S. (2000) Towards a philosophy of virtual reality: Issues implicit in "consciousness reframed". *Leonardo*, 33(2), 125-132.
42. Kato, H., Billinghurst, M., Poupyrev, I., Imamoto, K. and Tachibana, K. (2000) Virtual object manipulation on a table-top AR environment. In: *Augmented Reality, Munich, Germany, 5-6 October New York, USA: IEEE*, 111-119.
43. Khan, A.I., Qurashi, R.J. and Khan, U.A. (2011) A comprehensive study of commonly practiced heavy and light weight software methodologies. [online] Available from <https://arxiv.org/pdf/1111.3001> [Accessed 17 April 2017].

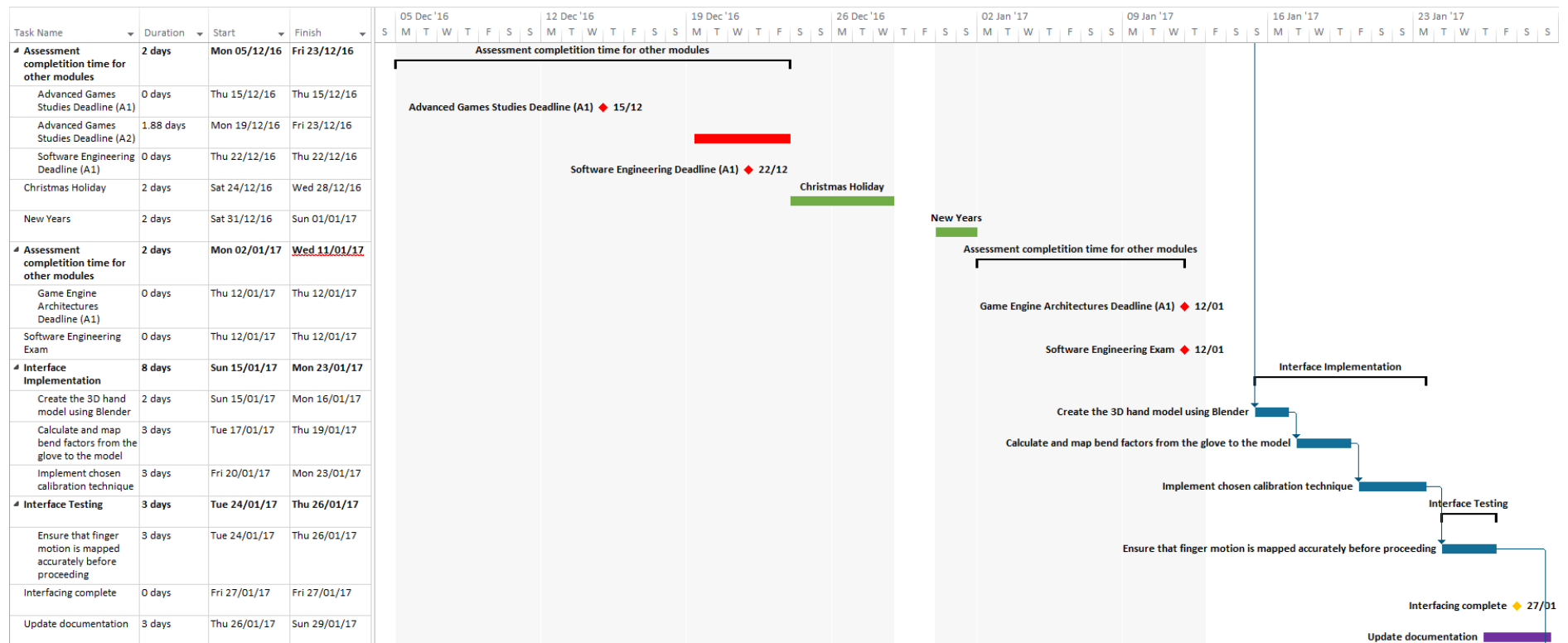
44. Kim, D., Hilliges, O., Izadi, S., Butler, A.D., Chen, J., Oikonomidis, I. and Olivier, P. (2012) Digits: freehand 3D interactions anywhere using a wrist-worn gloveless sensor. In: User interface software and technology, Cambridge, USA, 7-10 October New York, USA: ACM, 167-176.
45. Kwak, Y.H. and Ibbs, C.W. (2002) Project management process maturity (PM) 2 model. *Journal of management in engineering*, 18(3), 150-155.
46. Leading Ones (2016) Intro to VR: Degrees of Freedom. [online] Available from <http://www.leadingones.com/articles/intro-to-vr-4.html> [Accessed 12 April 2017].
47. Lewis, J. and Neher, K. (2007) Over the Waterfall in a Barrel-MSIT Adventures in Scrum. In: Agile, Washington, USA, 13-17 August Tennessee, USA: Agile Alliance, 389-394.
48. Limperos, A.M., Schmierbach, M.G., Kegerise, A.D. and Dardis, F.E. (2011) Gaming across different consoles: exploring the influence of control scheme on game-player enjoyment. *Cyberpsychology, Behavior, and Social Networking*, 14(6), 345-350.
49. Lindley, S.E., Le Couteur, J. and Berthouze, N.L. (2008) Stirring up experience through movement in game play: effects on engagement and social behaviour. In: Human Factors in Computing Systems, Florence, Italy, 5-10 New York, USA: ACM, 511-514.
50. Martin, J. (1991) Rapid application development. London, UK: Macmillan Publishing.
51. Maylor, H. (2001) Beyond the Gantt chart: Project management moving on. *European Management Journal*, 19(1), 92-100.
52. Oman, C.M. (1990) Motion sickness: a synthesis and evaluation of the sensory conflict theory. *Canadian journal of physiology and pharmacology*, 68(2), 294-303.
53. Ripton, J. and Prasuethsut, L. (2015) The vr race: who's closest to making vr a reality. [online] Available from <http://www.techradar.com/news/world-of-tech/future-tech/the-vr-race-who-s-closest-to-making-vr-a-reality--1266538>. [Accessed 14 April 2017].
54. Road to VR (2016) Manus VR's gloves in action using Valve's lighthouse tracking for hands and arms. [online] Available from <http://www.roadtovr.com/manus-vrs-gloves-in-action-using-valves-lighthouse-tracking-for-hands-and-arms/> [Accessed 12 April 2017].
55. Schmierbach, M., Limperos, A.M. and Woolley, J.K. (2012) Feeling the need for (personalized) speed: How natural controls and customization contribute to enjoyment of a racing game through enhanced immersion. *Cyberpsychology, Behavior, and Social Networking*, 15(7), 364-369.
56. Skalski, P., Tamborini, R., Shelton, A., Buncher, M. and Lindmark, P. (2011) Mapping the road to fun: Natural video game controllers, presence, and game enjoyment. *New Media & Society*, 13(2), 224-242.
57. Slater, M., Usoh, M. and Steed, A. (1995) Taking steps: the influence of a walking technique on presence in virtual reality. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 2(3), 201-219.
58. Slater, M., Linakis, V., Usoh, M., Kooper, R. and Street, G. (1996) Immersion, presence, and performance in virtual environments: An experiment with tri-dimensional chess. *Virtual Reality Software and Technology*, Hong Kong, China, 1-4 July New York, USA: ACM, 72.

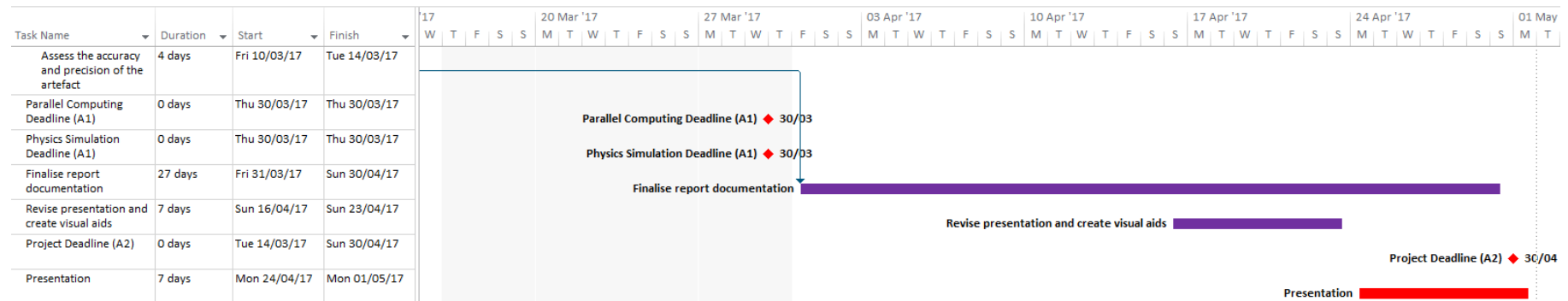
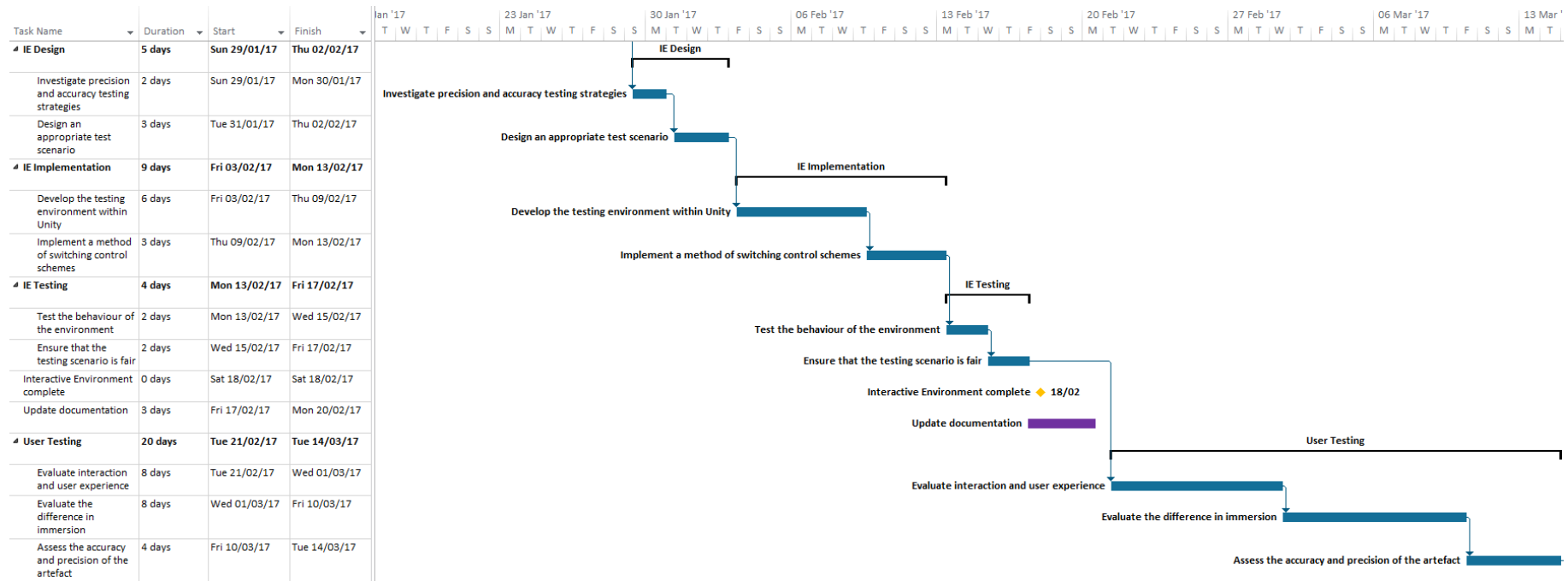
59. Slater, M. and Steed, A. (2000) A virtual presence counter. *Presence: Teleoperators and virtual environments*, 9(5), 413-434.
60. Sliger, M. (2004) Fooling around with XP. *Better Software*, 6(5), 16-18.
61. SparkFun (2016) Flex Sensor Hookup Guide. [online] Available at <https://learn.sparkfun.com/tutorials/flex-sensor-hookup-guide> [Accessed 26 April 2017].
62. Spectra Symbol (2014) Flex Sensor Data Sheet. [online] Available from <http://www.spectrasymbol.com/wp-content/uploads/2016/12/FLEX-SENSOR-DATA-SHEET-v2014-Rev-A.pdf> [Accessed 24 April 2017].
63. Sturman, D.J. (1991) Whole-hand input. PhD. Massachusetts Institute of Technology. Available from <https://pdfs.semanticscholar.org/8132/f3696a183700962099f5e85a9b891f8d5f98.pdf> [Accessed 12 April 2017].
64. TatvaSoft (2015) Top 12 Software Development Methodologies & its Advantages. [online] Available from <http://www.tatvasoft.com/blog/top-12-software-development-methodologies-and-its-advantages-disadvantages/> [Accessed 18 April 2017].
65. Unity Technologies (2004) Unity Game Engine. [software] San Francisco, USA: Unity Technologies. Available from <https://unity3d.com/> [Accessed 12 April 2017].
66. Walson, P.D. (1999) Patient recruitment: US perspective. *Pediatrics*, 104(3), 619-622.
67. Williams, L. and Cockburn, A. (2003) Guest Editors' Introduction: Agile Software Development: It's about Feedback and Change. *Computer*, 36(6), pp.39-43.
68. Zubrycki, I. and Granosik, G. (2014) Using integrated vision systems: three gears and leap motion, to control a 3-finger dexterous gripper. In: *Recent Advances in Automation, Robotics and Measuring Techniques*. Cham, Switzerland: Springer International Publishing, 553-564.

Appendix A – Project Management

Figure A.1 – Original project Gantt chart.







Key:

- (task)
- (holiday)
- (documentation)
- (project milestone)
- (external factor)

Figure A.2 – Log of GitHub commits.

27/04/2017 Commits · necronDOW/DataGlove_Dissertation



This repository Search Pull requests Issues Gist

[necronDOW / DataGlove_Dissertation](#) Private Unwatch 1 Star 0 Fork 0


Code Issues 0 Pull requests 0 Projects 0 Wiki Pulse Graphs Settings

Branch: master



Commits on Apr 4, 2017

-  Slight adjustment. 5a99c22
necronDOW committed 22 days ago
-  Configured for testing. 8e800c6
necronDOW committed 23 days ago


Commits on Apr 3, 2017

-  Added NVR and finished gripping interaction. 6342e15
necronDOW committed 23 days ago


Commits on Apr 2, 2017

-  Code adapted to accept per-sensor resistance ranges. 590a5d4
necronDOW committed 24 days ago
-  Minor changes. 6e6aea1
necronDOW committed 24 days ago


Commits on Mar 9, 2017

-  Added blend file for data glove case. c39e0e4
necronDOW committed on 9 Mar



Commits on Mar 7, 2017

-  .ino moved to folder. f0a7bbe
necronDOW committed on 7 Mar



Commits on Mar 5, 2017

-  Minor re-factoring. ad43773
necronDOW committed on 5 Mar



Commits on Mar 4, 2017

-  Updated version and added editor scripts. ... c0b50fe
necronDOW committed on 4 Mar
-  Dependencies added. ... abfa8d3
necronDOW committed on 4 Mar

Commits on Mar 1, 2017

-  Minor re-factoring and interface. ... 4fe7d32
Marly1995 committed on 1 Mar
-  Refactoring and Arduino Interface. ... 4217f78
necronDOW committed on 1 Mar


Commits on Feb 27, 2017



-  Fixed table assets. 738c76f
necronDOW committed on 27 Feb
-  Added table assets 98a5c3e
necronDOW committed on 27 Feb
- Added table assets.

https://github.com/necronDOW/DataGlove_Dissertation/commits/master 1/2


27/04/2017



Commits · necronDOW/DataGlove_Dissertation

necronDOW committed on 27 Feb


c9ba80f



<> Commits on Feb 24, 2017


Added table.
necronDOW committed on 24 Feb



cce1a06


<> Commits on Feb 20, 2017



Minor changes.
Marly1995 committed on 20 Feb


1b59b92



Updated test scene lighting and floor.
Marly1995 committed on 20 Feb


6a91f95



Updated README.md ...
Marly1995 committed on 20 Feb

a4b4cf7


Added debugging for all finger movement.
Marly1995 committed on 20 Feb



f0386f5


Updated README.md ...
necronDOW committed on GitHub on 20 Feb



711c9ad

<> Commits on Feb 19, 2017


Merged.
necronDOW committed on 19 Feb



31c9768


Big Changes ...
necronDOW committed on 19 Feb



c91dc3e

<> Commits on Feb 7, 2017


Saved scene.
necronDOW committed on 7 Feb



9b019c6

Finished resistance... Needs testing.
necronDOW committed on 7 Feb


2152594



<> Commits on Feb 5, 2017


Added base code for configuring glove.
necronDOW committed on 5 Feb



38f1374


<> Commits on Jan 14, 2017



Added two classes for interfacing with the Arduino.
necronDOW committed on 14 Jan


f553a83

Added core Unity project.
necronDOW committed on 14 Jan

c6a30b7

Updated README.md
necronDOW committed on GitHub on 14 Jan

f86e407

Initial commit
necronDOW committed on 14 Jan



3b35da3



Figure A.3 – Graph of GitHub contribution frequency from 08/01 to 27/04.

Jan 8, 2017 – Apr 27, 2017

Contributions: Commits ▾

Contributions to master, excluding merge commits

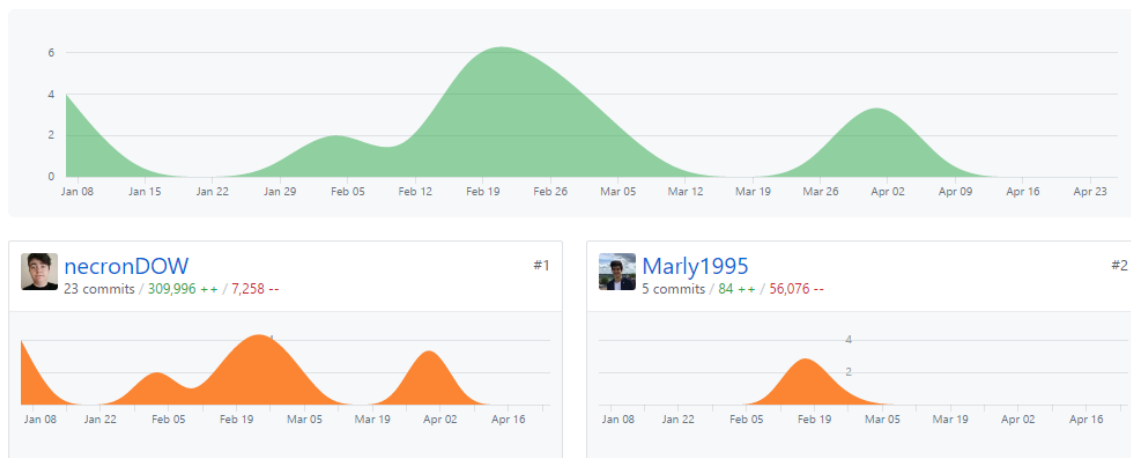
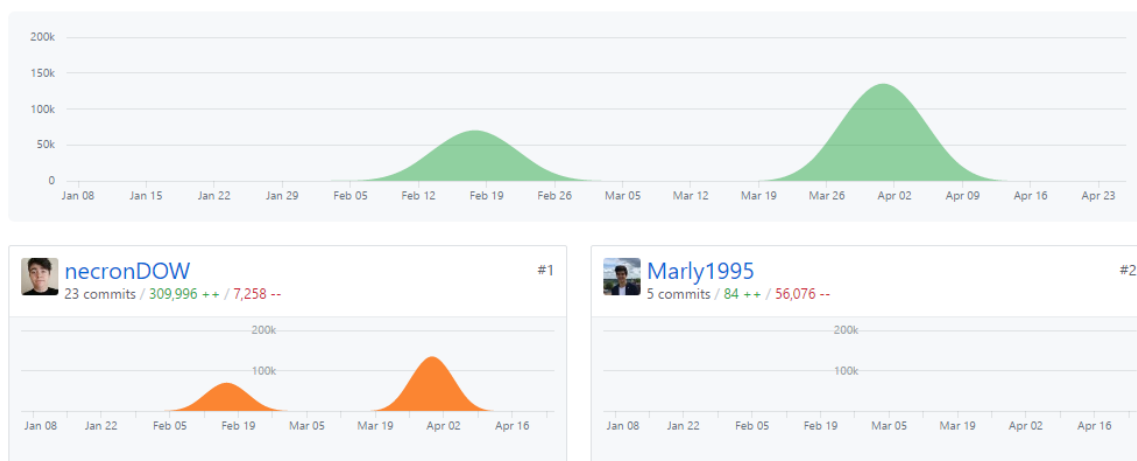


Figure A.4 – Graphs of GitHub additions and deletions from 08/01 to 27/04.

Jan 8, 2017 – Apr 27, 2017

Contributions: Additions ▾

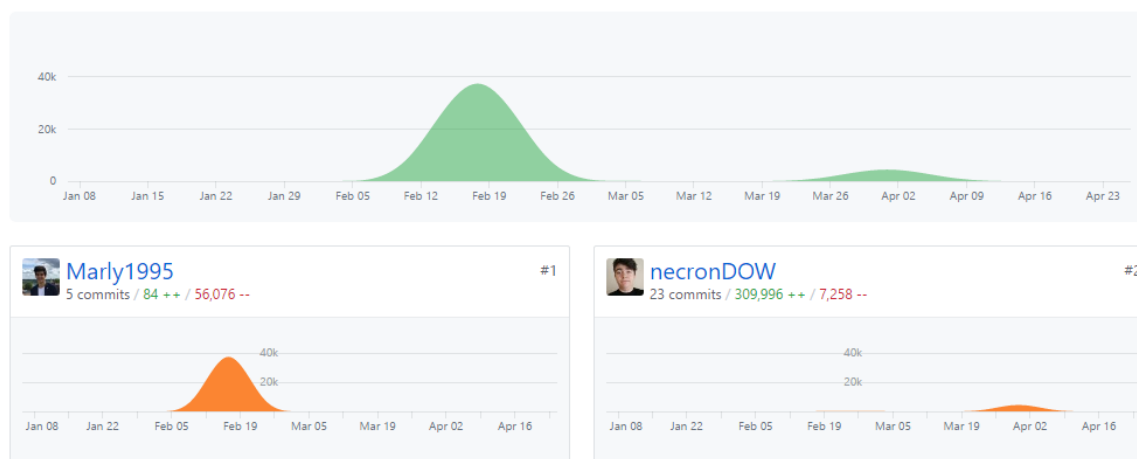
Contributions to master, excluding merge commits



Jan 8, 2017 – Apr 27, 2017

Contributions: Deletions ▾

Contributions to master, excluding merge commits



Appendix B – Research Resources

Figure B.1 – Scanned evidence of consent forms being used.

**** PARTICIPANT NAME AND SIGNATURE COVERED FOR ANONYMITY PURPOSES ****

Is Finger Motion Tracking a Superior Control Method for Virtual Reality?

Supervisor: Dr John Murray
Email: jomurray@lincoln.ac.uk
Researcher: Liam Wilson
Email: 13458211@students.lincoln.ac.uk

In this study you will use a Virtual Reality system, the HTC Vive in combination with both a standard controller and a custom built Data Glove controller. The purpose of this study is to measure the immersion experienced when using different methods of interaction within a VR environment, and a series of tests will be conducted to achieve this.

When using VR, there is a very small chance that a user could experience mild dizziness and nausea, with a slightly higher probability in people who suffer from motion sickness. Because of this, the participant is requested to inform the researcher of any negative symptoms experienced, which will result in immediate termination of the study session.

During the study, the participant will experience three separate scenarios. The first will ask the participant to interact with virtual objects using the standard HTC Vive controller. The second will ask the participant to repeat the first scenario using the new Data Glove. Finally, the user will be asked to repeat the second scenario with some physical objects in the environment.

Once the simulation is complete, the participant will be asked to fill in a short questionnaire to assess the overall experience. The data gathered from this questionnaire will remain confidential and anonymous, and any personal details will not be recorded or used.

The participant is free to withdraw from the study at any time and their recorded responses will be discarded and destroyed in this instance.

Please tick box

- I confirm that I have read and understand the information above and have had the opportunity to ask questions. ☒
- I understand that my participation is voluntary and that I am free to withdraw at any time, without giving reason. ☒
- I agree to take part in the above study. ☒
- I agree to the use of anonymised quotes in publications. ☒

Name of Participant _____ Date 04/4/17 Signature _____

Name of Researcher Liam Wilson Date 4/4/17 Signature [Signature]

**** PARTICIPANT NAME AND SIGNATURE COVERED FOR ANONYMITY PURPOSES ****

Figure B.2 – Research questionnaire.

4/27/2017

dissertation_questionnaire - Google Forms



QUESTIONS

RESPONSES 20

Is Finger Motion Tracking a Superior Control Method for Virtual Reality?

All of the information within this questionnaire will remain confidential as described in the consent form presented before the testing took place.

Participant Information

This section asks that you provide some information about yourself which will later be used in quantifying the responses given.

Gender *

- ☐ Male
- ☐ Female

Age *

- ☐ 18-21
- ☐ 22-25
- ☐ 26-39
- ☐ 40+

Have you used Virtual Reality before? *

- ☐ Yes, I have used the HTC Vive
- ☐ Yes, but I haven't used the HTC Vive
- ☐ No

Interaction



<https://docs.google.com/forms/d/1xjBxauazYzf7U1UWv4NeumzC7AY6PRLp5pJlmsdxhLQ/edit>

1/3

4/27/2017

dissertation_questionnaire - Google Forms

Description (optional)

In which test did interaction feel most comfortable? *

- ☐ First, with the HTC Vive controller
- ☐ Second, with the Data Glove
- ☐ Third, with the Data Glove and physical objects

Were there any particularly challenging interactions when using the Data Glove? *

- ☐ Yes
- ☐ No

If yes, please explain your answer.

Long-answer text

"It was more difficult to interact with objects using the Data Glove rather than the HTC Vive controller." *

	1	2	3	4	5	
Strongly Disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly Agree

"The presence of real world objects improved interaction." *

	1	2	3	4	5	
Strongly Disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly Agree

Immersion

Description (optional)

"Using the Data Glove (without physical objects) made me feel more immersed when" *

<https://docs.google.com/forms/d/1xjBxauazYzf7U1UWv4NeumzC7AY6PRLp5pJlmsdxhLQ/edit>

2/3

4/27/2017

dissertation_questionnaire - Google Forms

	1	2	3	4	5	
Strongly Disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly Agree

"The presence of real world objects improved immersion." *

	1	2	3	4	5	
Strongly Disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly Agree

"The physical presence of the Data Glove device detracted from the experience."

	1	2	3	4	5	
Strongly Disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly Agree

Is there anything else you would like to mention?

Long-answer text

<https://docs.google.com/forms/d/1xjBxauazYzf7U1UWv4NeumzC7AY6PRLp5pJlmsdxhLQ/edit>

3/3