



DESIGNING A GAME IN UNITY THAT UTILIZES LEAP
MOTION TECHNOLOGY

by

Joshua Newlands

Supervisor: Nicola Whitehead

Project submitted as part of the requirements
for the award of BSc (Hons) Applied Computing

April 2019

DECLARATION FORM AND FORM OF CONSENT

DECLARATION of Originality

I, Joshua Daniel John Newlands declare that I am the sole author of this Project; that all references cited have been consulted; that I have conducted all work of which this is a record, and that the finished work lies within the prescribed word limits.

This work has not previously been accepted as part of any other degree submission.

Signed: Josh Newlands

Date: 22/01/2019

FORM OF CONSENT

I Joshua Daniel John Newlands hereby consent that my Project, submitted in candidature for the *B.Sc. computer science* degree, if successful, may be made available for inter-library loan or photocopying (subject to the law of copyright), and that the title and abstract may be made available to outside organisations.

Signed:JoshNewlands

Date: 22/01/2019

ABSTRACT

This project explores the connection of leap motion gestures with human factors such as fatigue. The project also discusses the connections between leap motion HCI and traditional forms of HCI, viewing how the interactive actions change when dealing between 2D virtual interactions and 3D virtual interactions. The main focus is of the use of leap motion in relation to gameplay and game development, looking at leap motion in existing games and drawing theories based on the experiences involved in those to discuss possible improvements.

CONTENTS PAGE

Declaration FORM AND FORM OF CONSENT	2
Abstract	3
Contents Page	4
Chapter 1	6
1.1 Introduction	6
1.2 Aim	6
1.3 Objectives	6
1.4 Structure Of The Project	7
Chapter 2 Research and Review of Literature	7
2.1 Introduction to the literature review	7
2.2 Gestures	8
2.2.1 Gesture based communication	8
2.2.2 Gesture based HCI	9
2.3 Gesture based games	11
2.5 Gesture based Tech	13
Leap motion	13
Kinect	14
Data Glove	14
Power Glove	14
Wii	15
Touch screen	15
Keyboard & mouse	15
Leap motions limitations	15
2.6 Gesture based development	16
2.7 Conclusion	18

Chapter 3. Research & Development Methodologies	18
3.1 Introduction	18
3.2 Research Overview	19
3.3 Research Design	19
3.4 Conclusion	20
Chapter 4 - Design and Implementation	20
4.1 Introduction	20
4.2 Problem Formulation	21
4.3 Design	21
4.4 Implementation	22
Chapter 5 - Testing	22
Chapter 6 - Conclusions and Recommendations	23
6.1 Conclusion	23
6.2 Recommendations For Further Research	23
Chapter 7 - Reflections	23
Chapter 8 - References	24
Appendices	27
Gantt chart	27
Game Scripts	27
Credits	27
EndTrigger	28
followPlayer	28
GameManager	29
GestureManager	30
LeapBehaviour	31
LevelComplete	32
LoadMainMenu	33
MainMenuLoad	33
playerCollision	34
PlayerMovement	34

CHAPTER 1

1.1 Introduction

This project looks at designing a game in unity that implements leap motion technology.

Leap motion technology is a gesture capturing controller that utilises a small device to capture the movement of hands in real time and then utilising a complex driver the tool can recognise the hands and translate the information to the computer in real time.

Some problems that can arise in developing an application with this technology can range from player fatigue of holding their hands out too long, to motions that may flicker out of track, as well as designing a game that can fully utilise the potential of the controller.

The literature review will be looking reports and books that either relate or come across problems relating to the title of this report, in order to view conclusions and come up with methods of design that can overcome these common problems.

1.2 Aim

The Aim of this project is to design a functioning prototype game using unity that utilises a leap motion controller in a way to consider human factors.

1.3 Objectives

The objectives are to:

- Create a game to showcase the leap motion tech.
- Test the game against a list of requirements.
- Interview people testing the game to get data about whether there is too much fatigue in using leap motion.
- Interview people testing the game to get data on if the leap motion control was an improvement to the game experience.

1.4 Structure Of The Project

This report consists of three sections, the first section looks at already established theories. The theories relate to the current uses of leap motion technology as well as similar technologies that exist. It also describes underpinning development software that is made available with the technology. The second section of the report goes through the research methods that are carried out in order to develop the game that meets the aims and objectives of the project. Lastly, the third section looks at the implementation of the findings and compares them to the original requirements set out in this project, evaluating and giving recommendations for future studies relating to the work.

Chapter one presents an overview of the report and details the aims and objectives that will measure the success of this research.

Chapter two reviews current literature in the field concentrating on relevant section to the technology.

Chapter three establishes the research methodology that is used in this project to develop the software.

Chapter four outlines how the requirements will be met, the software used, the justification of the design, and showing the implementation of this with a section discussing the conclusion.

Chapter five shows the testing and evaluates the results against the aims and objectives.

Chapter six draws conclusions and recommendations for future studies based on the findings of this project.

CHAPTER 2 RESEARCH AND REVIEW OF LITERATURE

2.1 Introduction to the literature review

The general topics that this literature review focuses in are Gestures, Gesture based communication, Gesture based hci, gesture based games, gesture based technology, and gesture based development.

2.2 Gestures

With the leap motion controller capturing hand position and movement the ability to communicate meaning with hands becomes an important factor in design utilising leap motion controllers. For this reason this section will look at current reports on gesture based communication and HCI.

2.2.1 Gesture based communication

Gestures are defined as any intended human movements that convey meaningful information for interaction [1], in a survey done to understand what types of gestures are most commonly used the results showed a 21% usage of hands, a 10% usage of fingers and a 13% usage of 'multiple hands' which makes up a 44% usage of the hands and fingers for gestures[5]. This shows that a large amount of gesture based communication can be done through the use of the hands rather than needing the whole body.

Gestures can be categorized into two types so far, static and dynamic gestures. Static gestures do not require any movement and relate to a single posture that conveys information. Dynamic gestures involve the movement of the gesturing body part[7]. Static gestures may be of great significance to HCI in context of the application in use, holding a posture that points upwards could have the application scroll up a page until the posture ends [4].

In looking at 3D visual based gestures, focusing on the hand gestures, there are several forms that are important to consider. Other than the recognition of hand poses, there is sign language which is a more complex way of communicating with our hands[4]. Though a lot of sign languages use the rest of the body in parts for communication some simple words and phrases from the languages could become a important focus point for gesture recognition. For instance, to close an application the simple gesture through sign language would be significant [5].

A study looking into a review of hand gesture recognition and application finds that the complexity and diversity of hand gestures poses a major problem in getting a standard into hand gesture recognition [11].

2.2.2 Gesture based HCI

When computers were first created they were used for very specific uses; most people had no use for them. Things have changed since the early days of computer and developments to hardware and software have made them a necessary component of everyday life. A majority of the world is using smartphones, smart tvs and other computing devices designed to enrich our lives [1]. Since computers have become this necessary to modern life, there is no one group of individuals using computers, computer users are a very diverse group with all sorts of different backgrounds and abilities. The human computer interaction aspects of computer design are very important to ensuring as many users as possible are catered for[1]. Studies have been done with the goal of reaching conclusions on how to create more suitable and natural connections between the user and a computers interface.

When we think of interacting with a computer we may think of a keyboard and mouse, but these hardware devices are not the only ways we interact. Smart TV's use a remote, mobile phones use a touch screen, we can use our voice to interact with computers. Using our voice creates a more natural interaction between us and a computer[2]. Another natural way to interact is with a gestures which may sound a simple way to interact, but a challenge arises when we consider not everyone's the same and create a standard way of doing things will need to take into account a very large amount of research into what people want. Creating a standardised subset of universally easily understood gestures that can be recognised by the computer to give the expected result is a task many software developers have been challenged with[2].

A gesture based control as done in the past may have the user simply point their hand over graphical sections of the screen to achieve results such as activating buttons on a screen, however as pointed out by Cao X "The interface is simple to use and easy to understand, but it does not breakthrough from the mouse/pointer metaphor, and does not make use of the rich information conveyed by human hands"[1].

A common-place understanding of gestures would be hand based such as waving the hand, or indicating a value with fingers, these would make the most sense to study for HCI, a study has been completed for implementing a vision-based gesture classification. [2]. The VBGC showed that the analyses of hand gestures can be interpreted in two ways, 2D and 3D. The

2D, though more reliable, could not show the more complex gestures that convey more sets of information, whilst the 3D form required a more robust system for being able to accurately detect the gestures[3].

A study into the hci recognition of static gestures showed that the static hand gestures aren't all easily recognised by the gesture recognition algorithm. In the below figure it concludes that the parts of static gestures that use folded fingers are not recognisable whilst the parts that are held out from the hand are [9]. Whilst the accuracy of detection of the outstretched fingers are high in 95.44%.

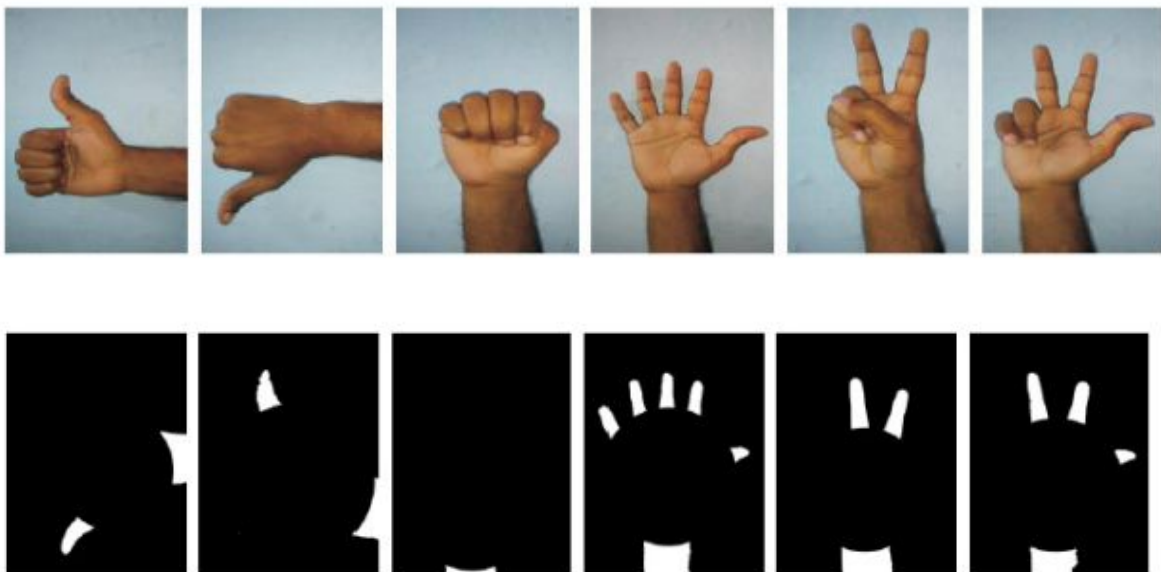


Fig 1. Static hand gesture numbered class 1-6, [9].

For dynamic gesture based hci leap motion tech have provided a few key gestures that can be utilised. The key gestures are the following;

- "Circle - A single finger tracing a circle.
- Swipe - A long, linear movement of a finger.
- Key tap - A tapping movement by a finger as if tapping a keyboard key.
- Screen tap - A tapping movement by the finger as if tapping a vertical computer screen." - [8]

These simple gestures that are recognised by the leap motion controller all have one thing in common: are single finger movements[8]. This coupled with the previous discussion on the static gestures in HCI show that the fingers are the key elements that are recognised by gesture capturing technology.

The lead developer relations of leap motion Inc mentioned that the leap motion technology is not to replace the mouse and keyboard but to enhance its use, following this principle there have been developments of applications that front the progress on the enhancement of HCI both on a day to day level as well as on a high end specialist environment [15].

SpaceX company that creates advanced rocket technology are a great example of the application of gesture based HCI, they took the leap motion controller interface and developed interaction methods that interact with the 3D CAD model virtual prototypes and designs of their projects[14]. Through the 3D environment the leap motion tech proved to be far more intuitive than the mouse and keyboard allowing for fewer obstructions when it came to the design process increasing workflow and productivity.

Hand gesture has further applications than just enhancing computer control, the hand gesture technology can be utilized in robot control, in recording set gestures and hand position movements. Hand gesture based on vision can be more free flowing than with previous attempts with data gloves, the data gloves also being expensive when compared to the availability of the Kinect with its visual gesture recognition technology [19].

Another application type simply utilises the leap motion controller to work in accordance with the mouse and keyboard in order to navigate the desktop as well as basic desktop applications such as scrolling the pages through loose relaxed gestures. Michael Sutherland (lead developer relations at leap motion Inc) doesn't completely agree with this approach as he suggested that the 2D desktop environment itself is outdated and the leap motion provides the perfect gate-way for the desktop to transition into a more natural gesture controlled 3D environment [15].

2.3 Gesture based games

Games have been created and played for generations becoming a more and more popular pastime, as such there have been many platforms for these games from SNES, Playstation

and xbox to handheld consoles such as the touchscreen ds. One platform has stood the test of time and is still widely used today and that is the adaptable and robust desktop computer, initial games being keyboard command based games, since then however games have taken a turn to being more complex in nature.

The keyboard and mouse has been a popular tool for interacting with games on the desktop computer, however it is found that these standard tools whilst smooth in the 2D environment are found to be more cumbersome in some 3D games and can cause a rift in the immersive gaming experience. Utilizing gesture based controls would create a more immersive effect in a 3D modeled game.[7].

A problem with gesture based controls in a 3D environment is that they can't incorporate movement naturally. Due to the user being stationary usually within a small sized room there is not widely available tech to overcome this using feet movement [16]. Several games have tried to bridge this gap using a variety of tactics in gesture based controls.

One attempt to implement player movement naturally using gesture based controls was done by the paper plane game, it uses a self propelled plane to progress whilst the user controls the acceleration with their left hand and its direction with their right hand. Unfortunately prolonged playing of this style caused player fatigue in the arms and though leap motion tech was robust it loosed tracking of the acceleration in angles over 80 degrees[12]. A major problem found with gesture based games is player fatigue with a study being done into gesture-based interactions in video games with the leap motion controller finding that in a survey of people a large complaint in the experience was player fatigue after around 20 minutes of play time[10].

Self moving games like these are common place with the leap motion technology as can be seen on the leap motion game jam, other styles of player movement are usually seen on games that require a vr headset for direction, such as in the game "spaceship" that more fully explores the use of gestures and complex hand movements in order to perform relatively simple and complex tasks to "fix a spaceship"[13]. It looks at common place hand movements that we recognise with tools such as hammer, spanner and screwdriver whilst hints about other hand movements the player may need for the task at hand.

Games like Alien Invasion, a 2D platform city defence game incorporated leap motion controls into the unity project, the game included a visual representation of the hands for the users to always know where they were in relevance to the game field. The gestures that they chose to include in this game were simple[10].

“Left/right movement by rolling the left hand; the direction of the back of the hand determines the direction the character moves – Jumping by swiping the left hand upwards – Firing missiles by making a fist with the right hand – Laying bombs by swiping the right hand upwards or downwards, or any vertical movement with higher velocity – Emitting a shockwave by swiping the right hand left and right, or any horizontal movement with higher velocity “ - [10]

Previously discussed a fist is unable to be detected by the leap motion controller, within the description of how the controls were achieved it does go on to say that the fist was detected by calculating if the players finger angles were that of over the threshold. In order to overcome some sense of player fatigue the game implements a deadzone into the leap motion control and allows for the deadzone to be adjusted based on player preference [10].

For basic game ideas the games that have been generated by the community show that the gesture controls included in the leap motion sdk controller can be more than enough for players to interact effectively[10]. Some only adding very few gestures of their own, whilst complex and revolutionary ideas vote to create a whole new set of gestures especially for their own game mechanic [13].

2.5 Gesture based Tech

This section looks to present general information on the capabilities and comparative usability of leap motion and similar gesture based technologies.

Leap motion

A prominent and recent technology developed on the front of gesture input is leap motion tech, Leap motion technology was developed and released by leap motion, Inc in July 2013. The manufacturer states that the sensor's accuracy in fingertip position detection is to an approximate degree of 0.01 mm at 120 frames per second using infrared cameras, claiming to give a 135 degree field of view with roughly 8 cubic feet of interactive 3D space. Michael Sutherlands the developer relations leader has mentioned that the hardware is excessively simple consisting purely of infrared cameras that send a series of pictures to the computer

through the usb, all of the tracking algorithms and complex data analyzing are done purely by the leap drivers and not the leap motion controller[15].

In a talk that the leader of the developer relations team Michael Sutherland gave he demonstrates how to begin as a developer of leap based applications. He shows how the 3D information is taken and given to you in by the leap controller emphasising how unlike other gesture based controllers the information isn't given in a unfriendly 'blob' of data and instead shows real visual feedback on positions of the fingers and palms as well as orientation [15].

Kinect

One of the gesture based technologies that have been available for some time is the kinect, the kinect was a cost efficient equipment developed to be used as a gesture recognising input system for the xbox 360 to be played in games. a problem with the first version of this technology was the lighting playing an effect on the reliable tracking of the player. The focus of Kinect as a technology was to recognise whole body gestures, rather than the hand gestures that leap motion focuses on. Though in a study done on the effectiveness of the leap motion controller it does state that it's able to reduce the noise interference of light intensity effectively enough that it's no longer a concern.

Data Glove

Another technology that utilizes gesture based controls is the CyberGrasp data glove, it can transmit gestures of the hand to the computer in real time. Due to the direct wired nature of the data glove it provides a distinct advantage to the accuracy of gesture recognition, being able to recognise the position of all of the fingers without any light interference or complex visual recognition algorithms in place [16]. A downside to this is its expense, size and the direct wire being cumbersome to use from distance.

Power Glove

Similar to the Data Glove a Power Glove was brought out by the american toy manufacturer Mattel Inc. as a nintendo game controller. This glove tracked a through ultrasound acoustic system that when paired with receivers placed near the display monitor detected hand

position. This technology negated the need for vision based gesture recognition algorithms, whilst the position was detected this way the hand gesture was detected through an ink based sensor this proved to be less precise compared to the data glove but made it far cheaper whilst the acoustic detection system negated the need for the restricted range of use [18].

Wii

The Wii was a console released in 2006 that placed a focus on motion controlled gaming. The console introduced many people to the idea of motion controls and was able to be enjoyed by people of all ages. The games were designed to take advantage of the Wii remote and gesture controls to simulate a characters movements were able to be reflected as an action in the game. Nintendo has continued with motion controls in its newest console, the 'Switch'. The Wii sold over 100 million units, proving that there is a large market for motion controlled gaming.

Touch screen

Touch screens have been used in many hardwares such as the nintendo ds, smartphones, and laptops. This allows for simple gestures that interact with software in a 2D sense, whilst the leap motion focuses on the 3D application of the gestures. The 2D gestures fail to move beyond the mouse paradigm that is the point and click, this is an aspect that leap motion tries to move away from.

Keyboard & mouse

The Keyboard & mouse are the one of the most common and conditioned uses of HCI, this shaped the desktop environment to be 2D and flat for information processing beginning in the command driven operating systems where there used to be only the keyboard for HCI to the flat graphical user interface operating systems that were developed along the introduction of mouses.

Leap motions limitations

Looking on the advice that this company has given the software developers on how to design a smooth game using this technology demonstrates some of its 'weaknesses'. They state that pinching is a more effective technique for interacting with virtual objects than

'grabbing or clenching even providing this diagram for reference on how to show players how to accurately pick up virtual items.

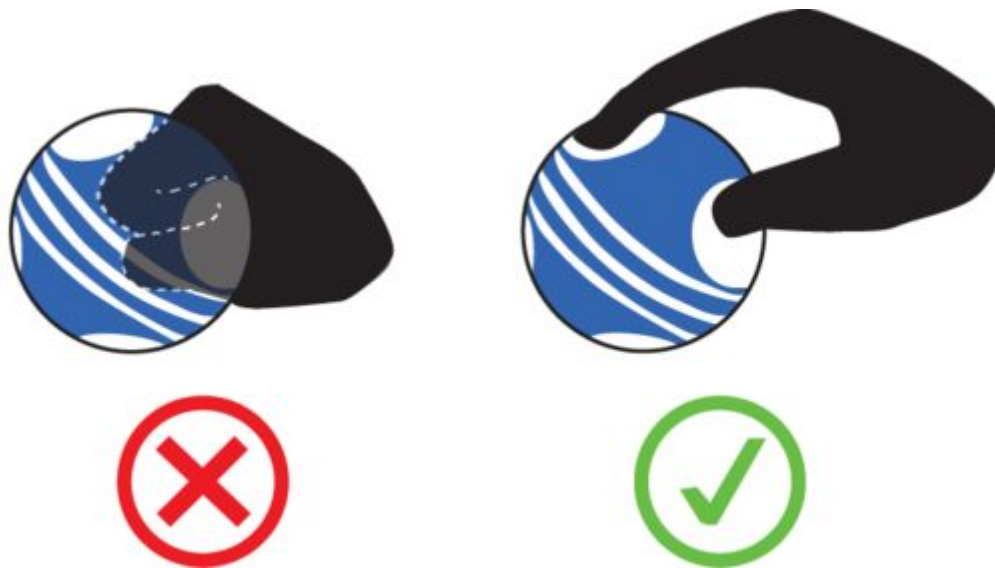


Figure 2: Diagram on how to correctly pick up virtual items taken from the leap motion instruction website.

When clenching the leap motion temporarily loses track of the position of the players fingers and starts to erratically predict a place for them which can cause difficult problems to deal with from both the players and software developers perspectives.

2.6 Gesture based development

Leap motion technology has provided developers with an SDK (Software Development Kit) and helpful tips for designing and developing games and applications that can synchronize well with the limitations of the device.

This is taken further when its demonstrated how the SDK has some precoded gestures recognised and can be used in development.

There are three simple gestures that are given by the sdk to be more easily implemented into development, circle, swipe and tap. The circle gesture shows the direction being made (clockwise, anticlockwise) as well the orientation the gesture is pointing at and the size of the gesture. utilising this can be as simple as 'is a gesture being made true or false, if true what direction, what orientation, what size'[15].

In the development of the games utilizing leap motion tech it can be easy to get too involved in the details of coding up the low level code to achieve a prototype, this can waste a lot of time when trying multiple ideas out, the developers of leap recommend using a prototyping framework in order to quickly pump out ideas before moving on to developing in a low level code for the final idea [15].

The developers of leap help ease new developers into using their tech by providing code examples of how the controller would work in applications as well as tutorials into the use of them, even going so far as to give exemplar menus so as to have the application developer focus on creating their innovative idea instead of getting bogged down by details, it further works to standardise menus making the users more familiar with how to navigate the menus across different applications using leap tech[17].

The leap motion / developer website as well as having an sdk available for developers to use it also pertains to a variety of languages that leap motion can be used in development with. Among the languages provided are Javascript, C#, C++, Java, and Objective-C. With each language the website gives a comprehensive list of the API references included with details perusing as to what each reference does in relevance to the leap motion[17].

An example of the references that leap motion dev documentation gives is the class 'gesture' in the C# language. They list the public functions available as well as how to use them, even stating that within the creation of a gesture object if that object is uninitialized then it renders the gesture invalid[17].

For the C# language in particular the leap motion developers have given a unity plugin for the use of leap motion with unity game development, they have also done the same for the unreal engine. The plugin description for the unity engine shows how the leap motion coordinate system is superimposed on the leap motion device in its desktop orientation as well as in its HMD orientation for the VR headset[17].

Included in the plugin is also Hand assets to use in unity, it gives a few basic assets that have physics and can be implemented with the gesture controls shown in the api references. This also gives the pre basis for creating custom hand assets from scratch.

2.7 Conclusion

There are many relevant conclusions that can be drawn from the literature review section. Gesture based communication are an essential stem point for the creation of controls for leap motion games, but some drawbacks from previously developed leap motion games can include player fatigue from the extended use of hands for gestures in high paced games. There can be many forms of gestures including static and dynamic, already built into the core asset api for unity is the recognition of simple gestures. Utilizing static gestures may result in the use of the leap motion controller being simplified to the point and click paradigm, using dynamic gestures may avoid this.

Through the study of various technologies similar to the leap motion controller it wouldn't be a stretch to conclude that the further away from the 'point and click' paradigm the technology tries to get the more the accuracy and tracking speed of the technology tends to suffer, whilst the leap motion is among the more accurate and faster tracking of these technologies, sticking to simple gestures may result in a higher reliability for tracking and accuracy during game runtime.

The next chapter discusses the research & development methodologies that are used for this project.

CHAPTER 3. RESEARCH & DEVELOPMENT METHODOLOGIES

3.1 Introduction

The methodology used for this project is one based on the agile principles. The methodology was created by another student for a project involving testing methodologies for solo development. Reviewing the methodology it is suitable for this project. The steps include doing a requirements analysis. This is essentially an analysis on what needs to be done for the software to be called 'finished'. The tasks of the project will be broken down into smaller tasks and a program called 'Pocket Kanban' on Android will be used to manage the tasks.

These tasks are part of a backlog of items that need to be completed. Kanban boards allow for tasks from the backlog to be organised into three categories: To Do, In Progress, and done. The work will be reviewed regularly and any changes that need to be made will be made to avoid any disruption in the development cycle. When all of the tasks are completed a complete system review will be done, which tests the system as a whole to see if all of the code works together. If this test is passed then the software will be completed, if errors exist they will be tested and fixed, before repeating the system test again to check it all works as it should.

3.2 Research Overview

This project uses the positivism philosophy with a deductive approach but is not truly deductive as a unstructured interview is used to gain feedback from the testing of the gameplay. The strategies used are document review, software development and surveys.

The positivistic paradigm is used in this for the concern with the generalisation from sample to population as well as the artificial location of the testing. Although the data produced from the unstructured interview would be quantitative, the context of the testing evens out the reliability. The deductive approach is concerned with testing theory, this is relevant to the testing of the game for usability/human factors, the project doesn't call for a full deductive approach due to the need for some generalisation and that data being gathered needing to be more qualitative than quantitative. The effect that this may have would be the reliability and validity of the findings averaging out neither being high nor low.

3.3 Research Design

In order to fulfill the aim and objectives of the research a plan of action is needed to focus the movement of the project to build an game application that meets the criteria. The research methodology that this project is using as previously stated is the agile methodology, this methodology is typically used in software development projects and would fit perfectly with the type of task that this project is focused on.

To begin the methodology the requirements are analysed to be broken down into smaller tasks, the smaller tasks then make up a backlog , this is essential to the creation of the outcome of this project. The methodology crosses over into the feature driven development

plan through the breaking down of requirements and the creation of the backlog, this means that each feature is established on a backlog and prioritised.

When a backlog is created it starts a cycle until the code is completed, the cycle progresses from prioritising an item on the backlog, to developing that item, reviewing that item in a daily review, and then updating the backlog before going back to prioritizing. At the update stage if all items are complete then a complete system test and a final review are done.

This type of methodology fits inline with the project's needs due to the focus on the breaking down of requirements and the focus on the development of features. This is a key point due to the type of software that this project developed, the game is a software that is full of features, without these the engrossment and enjoyment of the game would suffer.

The methodology allows for the steady progression of the software from start to finish whilst prioritising key features along the way.

3.4 Conclusion

To conclude this project loosely follows the positivism philosophy with the deductive approach due to the development of the project following these but the player testing phase slightly aligns with the Inductive approach due to the qualitative data format of the unstructured interviews. The research methodology for the development of this project follows agile as well as feature driven development, this is due to the objectives needing to be broken down into requirements that fulfill the criteria for the aim of the project. This methodology choice also helps with prioritising key features of the program

CHAPTER 4 - DESIGN AND IMPLEMENTATION

4.1 Introduction

This section outlines how the project will progress the solving of the problem statement to fulfill the aims and objectives declared earlier in the project. It will also discuss and justify the design decisions.

4.2 Problem Formulation

The aim of this project as stated earlier is to design a function prototype game using unity that utilises a leap motion controller in a way to consider human factors. In order to achieve this whilst using the development methodology stated in the previous chapter the initial objectives are broken down into features for a backlog. To keep track of this backlog and to move from 'ToDo', to 'InProgress', to 'Completed' an application is used that is available on the android called 'Pocket Kanban'. This creates a kanban board of the features that are being developed to be used in conjunction with the agile methodology.

4.3 Design

Earlier in the study stated that the leap motion company encouraged developers to take a standardised approach to HCI for leap motion games, this can be considered a requirement. The parts of this that need to be considered would be simplistic menu with large interactable buttons for navigation. In order to keep as much accuracy and tracking with the leap motion controller whilst still conforming to a more 3D HCI as possible utilising the in built simple gestures is a must in this game, this way the game still shows how the leap motion controller does not have to conform to the 'point and click' paradigm. The use of this form of gesture controls simplifies things to keep a constant and understandable control scheme.

In order to compensate for player fatigue which is a main focus of the design of this game, the game is designed to have short levels that interval between the use of the right hand and the left hand between levels. This approach allows the player to regularly rest each arm rather than having the strain rest solely on one arm or the other.

In order to create this the design of the code is a focus for the project, as mentioned earlier in the project the leap motion website has documentation for the api of the simple gesture controls as well as providing a core asset kit for unity.

4.4 Implementation

Following the agile methodology and kanban board, the features added are each whole and complete. The features include:

- The Player that has a following camera.
- Keyboard controls for comparison.
- Obstacles to dodge.
- Set paths for the player to follow for simplicity, left, right, and middle.
- A game manager that handles level restarts and level changes.
- Three short level designs.
- Simple Main menu with two large buttons (Start, and Quit).
- Simple Credits scene with two large buttons (Main Menu, and Quit).
- A End Trigger to react when the player passes the end of the level
- UI to display the distance that the player has traveled from the start.

Unfortunately the leap motion controls for the gestures have not been implemented which is further discussed in the reflections section.

CHAPTER 5 - TESTING

Due to the problem with the leap motion that is discussed further in the reflections section, the testing was limited to basic functionality testing.

The Start button on the main menu tested and working, progressing the player to the first level. The quit button tested in the build to quit on both the main menu screen and on the credits screen. The levels tested positive to the player progressing, the obstacle collision restarting the menu, the player keyboard controls moving the player between left lane, middle lane, and right lane. The level also tested positive to triggering the level complete UI animation when passing the goal, and progressing the player to the next level. Each level tested positive to progressing to the next scene. The Score UI tested positive to tracking the users progress through the level.

CHAPTER 6 - CONCLUSIONS AND RECOMMENDATIONS

6.1 Conclusion

This project explored the usability and limitations of leap motion. The project produced a theoretical solution to the creation of leap motion games that both consider human factors such as fatigue and also demonstrate the 3D capabilities of the leap motion controller. The project concluded that the use of reduced level time and alternating between left and right hands for each level may help alleviate human fatigue. As well as this it also described the possibility for finding a balance between the 3D HCI capabilities of the leap motion controller with the performance and tracking by using simple gesture controls.

6.2 Recommendations For Further Research

For further research the implementation of the theory into a tangible application would provide data on the findings of this project. For consideration of further research the use of the leap motion controller in games that also utilise the Virtual Reality headset to find performance and human factor considerate solutions to a immersive gameplay would be a further step in the direction of leap motion technology in game development.

Other recommendations would be the exploration of the interaction with the leap motion controller and virtual objects such as grabbing or pushing. Factors to consider with this may be balancing the accuracy of the tracking with the complexity of the gesture design whilst maintaining intuitive controls.

CHAPTER 7 - REFLECTIONS

The aims and objectives stated in the beginning of the project were to create a leap motion game in unity in a way that considers human factors such as fatigue. The theories involved were discussed and partly based off of the leap motion documentation as well as evaluation of previously developed games that utilise leap motion. The principles brought up involved shortening level length, and switching required arms in each level whilst implementing simple gestures. These principles lay a strong foundation for further exploration into developing fun games that don't allow player fatigue to take away from the experience.

Progression through the project was tracked with a gantt chart and changes were dynamically made to accommodate unforeseen changes in circumstances that may interfere with the projects progression. Although the gantt chart was carried out as well as a priority system with the agile methodology for the development section of the project the aims and objectives were not met in the final outcome. This is due to a lack of the inclusion of external factors to the project such as the workload towards the end of the project year as well as down to the oversight in an aspect of the research that led to a sudden complication in the development stage. The development stage then couldn't produce tangible results and left little time for effective testing.

The oversight was of the leap motion api documents that were available on the leap motion website. The documents detailed the previously mentioned inclusion of simple gesture recognition in the inbuilt api for the leap motion unity core assets package. Those documents were not updated by leap motion for the latest version of the unity core assets package, which is the only package readily available that is compatible with the current unity build. This version of leap motion core assets package, version 4.4, dropped the support for simple gesture recognition api completely. Had this been more thoroughly researched during the research stage of this project then the gantt plan could have been adjusted to compensate for the additional development time required to code custom gestures into the project game.

This project in result of this only achieves a theoretical solution to the aim but does not draw any conclusive data to present.

CHAPTER 8 - REFERENCES

[1]X. Cao, "AN EXPLORATION OF GESTURE-BASED INTERACTION", 2004.

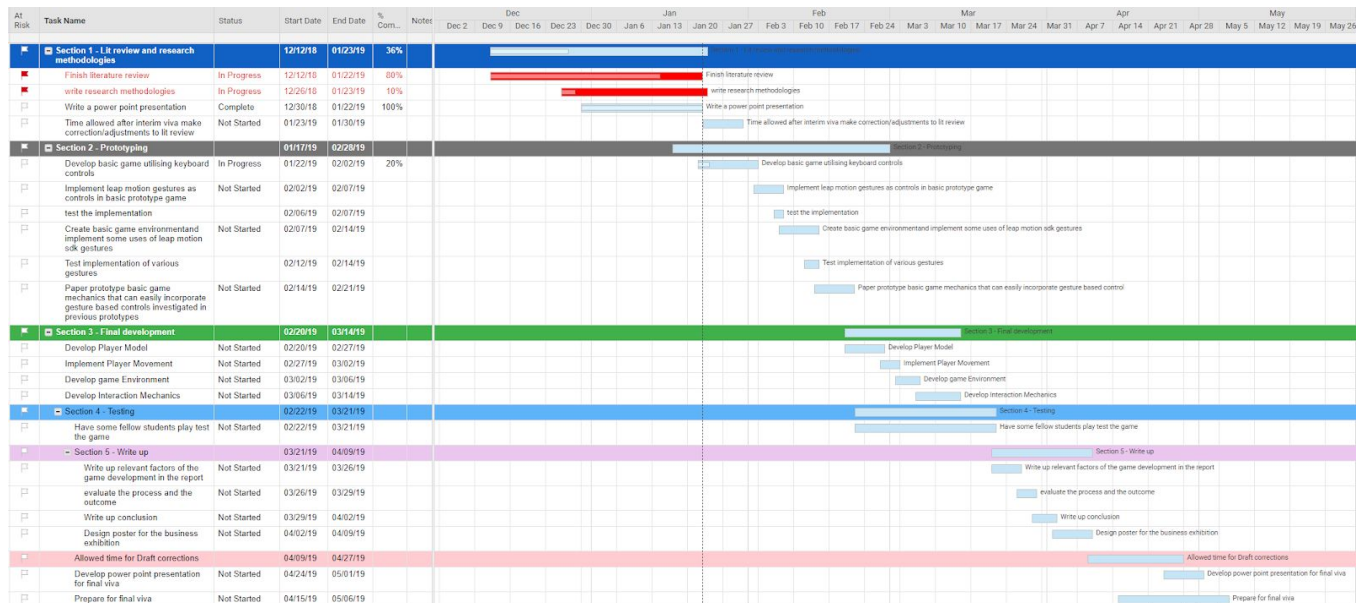
[2]L. Gupta and S. Ma, "Gesture-based interaction and communication: automated classification of hand gesture contours", *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, vol. 31, no. 1, pp. 114-120, 2001. Available: 10.1109/5326.923274 [Accessed 22 January 2019].

- [3]V. Pavlovic, R. Sharma and T. Huang, "Visual interpretation of hand gestures for human computer interaction: A review", *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 19, pp. 677-694, 1997. [Accessed 22 January 2019].
- [4]S. Mitra and T. Acharya, "Gesture Recognition: A Survey", *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, vol. 37, no. 3, pp. 311-324, 2007. Available: 10.1109/tsmcc.2007.893280 [Accessed 22 January 2019].
- [5]S. Rautaray and A. Agrawal, "Vision based hand gesture recognition for human computer interaction: a survey", *Artificial Intelligence Review*, vol. 43, no. 1, pp. 1-54, 2012. Available: 10.1007/s10462-012-9356-9.
- [6]G. Murthey and R. Jadon, "A Review Of Vision Based Hand Gesture Recognition", *International Journal of Information Technology and Knowledge Management*, vol. 2, no. 2, pp. 405-410, 2009. [Accessed 22 January 2019].
- [7]v. Gandhi, A. Kohnd, S. Raut, V. Thakur and S. Shaikh, "A Review of Various Gesture Recognition Techniques", *International Journal Of Engineering And Computer Science*, vol. 3, no. 9, pp. 8202-8206, 2014. [Accessed 23 January 2019].
- [8]LeapMotionDevs, "Gestures — Leap Motion C# SDK v2.3 documentation", *Developer-archive.leapmotion.com*, 2019. [Online]. Available: https://developer-archive.leapmotion.com/documentation/v2/csharp/devguide/Leap_Gestures.html?fbclid=IwAR2lc4k6bssxba7tJ0qQHEwbt9jFMBhnoN8FEMsLs2qTLZ8oPDUrBTtoYWHM. [Accessed: 23- Jan- 2019].
- [9]R. Sharma and G. Verma, "Human Computer Interaction using Hand Gesture", *Procedia Computer Science*, vol. 54, pp. 721-727, 2015. Available: 10.1016/j.procs.2015.06.085.
- [10]J. Pirker, M. Pojer, A. Holzinger and C. Gütl, "Gesture-Based Interactions in Video Games with the Leap Motion Controller", *Human-Computer Interaction. User Interface Design, Development and Multimodality*, pp. 620-633, 2017. Available: 10.1007/978-3-319-58071-5_47 [Accessed 23 January 2019].

- [11]Y. Xu and Y. Dai, "Review of hand gesture recognition study and application", *Contemporary Engineering Sciences*, vol. 10, no. 8, pp. 375-384, 2017. Available: 10.12988/ces.2017.7326.
- [12]P. Plane, "Paper Plane by VRARlab", *itch.io*. [Online]. Available: <https://vrrarlab.itch.io/paper-plane>. [Accessed: 23- Nov- 2018].
- [13]"Top games tagged Leap Motion", *itch.io*, 2012. [Online]. Available: <https://itch.io/games/tag-leap-motion>. [Accessed: 19- Nov- 2018].
- [14]"The Future of Design", *YouTube*, 2013. [Online]. Available: https://www.youtube.com/watch?v=xNqs_S-zEBY&feature=youtu.be. [Accessed: 28- Nov- 2018].
- [15]"Leap Motion SDK", *YouTube*, 2013. [Online]. Available: <https://www.youtube.com/watch?v=Fyk0F--43Cw&t=3071s>. [Accessed: 30- Nov- 2018].

APPENDICES

Gantt chart



Game Scripts

Credits

using UnityEngine;

```
public class Credits : MonoBehaviour
```

```
{
```

```
    public void Quit()
```

```
{
```

```
        Debug.Log("Quit");
```

```
        Application.Quit();
```

```
}  
}
```

EndTrigger

```
using UnityEngine;  
  
public class EndTrigger : MonoBehaviour  
{  
    public GameManager GM;  
    private void OnTriggerEnter()  
    {  
        GM.CompleteLevel();  
    }  
}
```

followPlayer

```
using UnityEngine;  
  
public class followPlayer : MonoBehaviour  
{  
    public Transform player;  
    public Vector3 offset;
```

```
void Update()

{

    transform.position = new Vector3(transform.position.x, transform.position.y,
(player.position.z - 5f));

}

}
```

GameManager

```
using UnityEngine;

using UnityEngine.SceneManagement;

public class GameManager : MonoBehaviour

{

    public bool gameEnded = false;

    public GameObject completeLevelUI;

    public void CompleteLevel()

    {

        completeLevelUI.SetActive(true);

    }

    public void EndGame()

    {
```

```

        if (gameEnded == false)
        {
            gameEnded = true;

            Restart();
        }
    }

    void Restart()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().name);
    }
}

```

GestureManager

```

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class GestureManager : MonoBehaviour
{
    //create enum list of gesture types

    //create ways to get current gesture type and store current gesture type

```

```
//a variable for a time block between two gestures
```

```
//create a dictionary of the active gesture types objects
```

```
}
```

LeapBehaviour

```
using System.Collections;
```

```
using System.Collections.Generic;
```

```
using UnityEngine;
```

```
using Leap;
```

```
using Leap.Unity;
```

```
public class LeapBehaviour : MonoBehaviour
```

```
{
```

```
    //create a gesture manager object
```

```
    //create a bool for if gestures are blocked
```

```
    //create a method to unblock gestures
```

```
    //create a way to get gesture types current type
```

```
    //create a list of active hands
```

```
//method to get current hand
```

```
//repeat to get two hands
```

```
//logic variables go here ( shit like smallest velocity to be considered still.
```

```
//initialise
```

```
//update hands and debug in fixed update
```

```
}
```

LevelComplete

```
using UnityEngine;
```

```
using UnityEngine.SceneManagement;
```

```
public class LevelComplete : MonoBehaviour
```

```
{public void LoadNextLevel()
```

```
{
```

```
    SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
```

```
}
```

```
}
```

LoadMainMenu

```
using UnityEngine;

using UnityEngine.SceneManagement;

public class LoadMainMenu : MonoBehaviour

{

    public void LoadMainScene()

    {

        SceneManager.LoadScene(0);

    }

}
```

MainMenuLoad

```
using UnityEngine;

using UnityEngine.SceneManagement;

public class MainMenuLoad : MonoBehaviour

{

    public void LoadMain()

    {

        SceneManager.LoadScene(1);

    }

}
```

playerCollision

```
using System.Collections;
```

```
using System.Collections.Generic;
```

```
using UnityEngine;
```

```
public class playerCollision : MonoBehaviour
```

```
{
```

```
    public PlayerMovement movement;
```

```
    void OnCollisionEnter(Collision collisionInfo)
```

```
    {
```

```
        if(collisionInfo.collider.tag == "Obstacle")
```

```
        {
```

```
            movement.enabled = false;
```

```
            FindObjectOfType<GameManager>().EndGame();
```

```
        }
```

```
    }
```

```
}
```

PlayerMovement

```
using UnityEngine;
```

```
using System.Collections;
```

```
public class PlayerMovement : MonoBehaviour
{

    public Rigidbody rb;

    public float forwardForce = 2000f;

    public KeyCode Left;

    public KeyCode Right;

    public bool Locked = false;

    public int currentLane = 2;

    public float horizontalForce = 0;

    public float horForceChange = 10f;

    void Update()
    {
        if ((Input.GetKeyDown(Right)) && (currentLane < 3) && (!Locked))
        {
            Locked = true;

            horizontalForce = horForceChange;

            currentLane += 1;

            StartCoroutine(StopForce());
        }
    }
}
```

```

if ((Input.GetKeyDown(Left)) && (currentLane > 1) && (!Locked))
{
    Locked = true;

    horizontalForce = -horForceChange;

    currentLane -= 1;

    StartCoroutine(StopForce());
}

if (rb.position.y < -1F)
{
    FindObjectOfType<GameManager>().EndGame();
}
}

void FixedUpdate()
{
    rb.velocity = new Vector3(horizontalForce, 0, forwardForce * Time.deltaTime);
}

IEnumerator StopForce()
{
    yield return new WaitForSeconds(.5f);

    horizontalForce = 0;
}

```

```
        Locked = false;

    }

}
```

ScoreCount

```
using UnityEngine;
```

```
using System.Collections;
```

```
public class PlayerMovement : MonoBehaviour
```

```
{
```

```
    public Rigidbody rb;
```

```
    public float forwardForce = 2000f;
```

```
    public KeyCode Left;
```

```
    public KeyCode Right;
```

```
    public bool Locked = false;
```

```
    public int currentLane = 2;
```

```
    public float horizontalForce = 0;
```

```
    public float horForceChange = 10f;
```

```
    void Update()
```

```
{
```

```
    if ((Input.GetKeyDown(Right)) && (currentLane < 3) && (!Locked))
```

```

{
    Locked = true;

    horizontalForce = horForceChange;

    currentLane += 1;

    StartCoroutine(StopForce());
}

if ((Input.GetKeyDown(Left)) && (currentLane > 1) && (!Locked))
{
    Locked = true;

    horizontalForce = -horForceChange;

    currentLane -= 1;

    StartCoroutine(StopForce());
}

if (rb.position.y < -1F)
{
    FindObjectOfType<GameManager>().EndGame();
}
}

void FixedUpdate()
{
    rb.velocity = new Vector3(horizontalForce, 0, forwardForce * Time.deltaTime);
}

```

```
}
```

```
IEnumerator StopForce()
```

```
{
```

```
    yield return new WaitForSeconds(.5f);
```

```
    horizontalForce = 0;
```

```
    Locked = false;
```

```
}
```

```
}
```