Motion Driven Computer Game Using Kinect
Final Report

Author: Simon Winton
Supervisor: Yukun Lai
Moderator: Michael Daley
Module Title: One Semester Individual Project
Module Number: CM3203
Credits: 40

Table of Contents

1 Introduction

1.1 Project Summary

The computer games industry is an ever-expanding market that in recent years has seen the release of several motion control devices that allow the user to control the in game actions with their movement, and in some cases without the need to touch a controller. An example of this is the Microsoft Kinect, which uses it cameras to sense the movement of the user and also the sounds that the user makes.

This project will focus on the development of a Windows PC game that incorporates the Microsoft Kinect as a controller, focusing both on the creation of a game as well as the challenges and execution of adding motion controls to it.

1.2 Project aims

There are three primary aims of the project. The first of which is to product a functional game. Creating a functional game will mean that the project will aim to have some form of level system in place for the user to progress. This game should be relatively unique to what's available and the final result should have at least one working level. The second aim is that controls for the game must use the motion detection in the Microsoft Kinect hardware in intuitive and simple movements. This means that the user should be able to navigate menus and play the game using simple gestures and movements. The movements should feel natural to the user and all the commands that the user wants to give should feel intuitive. Finally I will aim to make the game have clear user feedback. This will involve a clear representation of what the user is doing at all times including both navigating the menu and in game. The users should also have clear feedback on how the level is progressing via a scoring system and at the end of the level a summary of how well they did.

There are several secondary aims of this project; the first is to add extra Kinect functionality. I will be working at creating controls using body movements of the Kinect, however it would be beneficial to the user if they are able to choose to use spoken commands to perform certain actions or shortcuts within the game or menus. Although one of my primary aims is to provide a clear user interface I also believe that it would be beneficial to have a visually appealing user interface. Graphics in modern gaming is now considered almost as important as the gameplay itself meaning that if there is time it would be beneficial to make the game look pretty. Another secondary aim is to add sound effects and background music into the game. This will add atmosphere to the game and enrich the user experience, as another sense will be stimulated. To make the game more entertaining it would be nice to add in more levels with ranging difficulty. Therefore my third secondary aim is to add in a variety of levels to showcase what the game can do and provide significant challenges to the user.

This will provide more of a challenge to the user and encourage them to keep playing. Another aim will be to allow customisation within the game. This can be something as small as allowing users to add in their own objects, to more significant customisation like allowing the user to use left handed controls, or to set their own controls. Another form of customisation could be allowing the user to create their own levels, which will help the game expand as well as making the user feel more comfortable playing the game. The final secondary aim would be to implement social gaming as playing with friends has become more accessible and fun. This will give the game replay value, as players would be able to challenge their friends to beat their score, play directly against their friends or simply try to get to the top of a leader board.

The output of the project will be targeted towards the general public. To decide on who the game should be targeted at I first did some research. From this I was found that 52% of gamers in the United Kingdom are female, and surprisingly 27% of gamers are over forty-four and only 22% of gamers are children or teenagers [3]. As there is such a wide range of diversity with regards to gamers in terms of age, gender and interests the game should be not aim to target too heavily on one core customer type so it has mass appeal. One of the ways the game will need to do this is by insuring everyone can do the actions for the controls. This means that I should avoid actions that will avoid strenuous activities or difficult actions which players may find difficult.

1.3 Approach

To implement this project I will use a phased approach that I specifically modified to create this game. The first phase of the project will be a discovery phase. This stage will involve me doing research into what the best direction is for the task. I will be looking at multiple libraries, software, graphical styles, the best way to create a game and figure out what my game will be about. By the end of this stage I will have a clear idea of exactly what I want my outcome will look like and the tools that I will use to make it happen.

The second phase will be the design phase. In this phase I will lay out the plans for the project, laying out how I want the code to link together and designing the aspects of the game. I will define what the user requirements are, define how the user will view the game and decide how the user will control the game. When this stage is complete I will know how I am going to carry out the project.

The third phase is the implementation. This will involve me following the design to insure that I get the result I want. During the project there may be issues, at which time I may need to make adjustments that could differ from the original design. However the outcome of this stage will be an implementation of the design.

The forth and final phase is the reflection stage. At this time I will be reflecting back on the code that I have done in comparison to what I originally set out to do. I will then write out a list of what further improvements are needed in the game. I will then check if the requirements have been met, if I have completed

the primary user requirements and take the time to reflect on my personal growth and the experience I've gained. At the end of this stage the project will be completed.

1.4 Assumptions

Throughout this project I will make a couple of assumptions. I will first assume that the user is familiar with the concept of motion-controlled gaming in terms of that they understand how body movements are picked up and how the lighting should be. I will also assume that the user will be placing the Kinect in a position that is directly in front of them.

1.5 Summary of Outcomes

The project will have one outcome however there will be two main focal points to determining the success of the project. The first will be that a game is created. There will need to be some form of playable game that will involve a process of winning and losing, or a score driven game where the goal is to achieve the highest score. The second determining factor of the success of the project will be the use of the Kinect controls. The Kinect should be the sole source of controls in the game meaning the user should not have to use any other controllers other than when they are launching the game.

2 The Background

2.1 Microsoft Kinect

Gaming has become an ever-increasing pastime for people worldwide with 32.9 million gamers in the United Kingdom alone [1]. With a flourishing industry console developers are able to push the forefront of technology, produce new hardware and develop innovative software that increases the enjoyment and engagement that their customers receive from the gaming system. One such example of this is Microsoft's Xbox Kinect. The Kinect is a webcam based peripheral which was originally designed as an add-on the Xbox 360 console and released to the public in late 2010. The Kinect device uses a microphone, a RBG camera and an infrared projector to detect the movements and the sounds that the user makes. This allows for the games to incorporate gestures, spoken commands and body movements to enhance or at times fully control the game being played. In early 2012 we saw the release for the Kinect for windows which further improved the array of motion controlled games, however these generally consisted of smaller mini games rather than big title games that were appearing for consoles.

By releasing the Xbox for Windows it also made developing motion controlled games more accessible to indie developers and small game houses. Microsoft Had previously released the Kinect software development kit for Windows 7 in 2011 which allowed users to develop in Visual Basic.NET, C# and C++ however after releasing the Windows edition of Kinect developers started looking at further expanding the libraries, adding plug-ins and software development kits for a variety of different programming languages.

2.2 Constraints

There are several constraints in a project like this. The largest constraint on the project is time as the project only spans a twelve-week period from start to completion. This means that the game has to be adjusted so it is easily scalable. This will mean the end result will be able to provide the user with more of an experience and more to play for than they would be able to with just the one level. This should also mean that if the game were to be expanded after it would be easy to add in new levels. The constraint on time may also mean that I am unable to add in many extra features including sounds and multiplayer.

Another major constraint is that only one person will carry out the project. Although I will be able to get advice from my supervisor I will not get any other input into how the game should be made or other creative input. This could slow down my design stage and make the decision making process could be complicated as I will need to consider everything myself. The implementation stage will also suffer from the project only having one developer. In typical game development there will be a team of people who work for months to create a game and with larger titles the process of making a game could take years even with large teams. I will therefore have to make sure that I stick to my time plan

to insure that work gets completed and the final result is a fully functional game. I may find that I do not have the time to complete sections of the project; therefore it would be wise for me to prioritise what is best for the end result.

The project is also constrained by my current knowledge, or in this case lack of. This project will develop both my skills as a developer and my knowledge however learning will take up time that I do not have. It is important that I am aware of this constraint in the project and try not to do anything too extreme, as focusing on one part of it may result in the other parts of the game being under developed.

The Kinect sensor itself can also be considered a constraint. I will have to make sure that I fully understand the detail that the Kinect can pick up and how accurate movements are reflected on screen before creating my game. I will also need to make sure that my controls fit what the Kinect can pick up. For example the Kinect will not be able to see behind the user. This would mean that actions that involve the user facing sideways could potentially block out the use of one arm.

3 The Research

3.1 Selecting a Programming Language

The first determining factor of the project would be what language to use so I made it the first area of my research. The main limiting factor of the project in terms of deciding the language I should use was the languages that there are only a handful of languages that have established Kinect plugins. I then reviewed each language and narrowed down the possibilities to two languages, which were Visual Basic and Java. The reason these languages were specifically chosen was due to the personal knowledge that I have of each of these programs. By working in a medium that I am familiar with should allow me to speed up the programming time, allowing me to get more done in the process and allowing me to produce a more polished outcome.

I first looked into how to create Kinect games with Visual Basic. This is a language that is heavily supported by developers so there is a lot of information about how to program and it would be easy to find a solution if I were to get stuck. Visual Basic also has its own software development kit from Microsoft for developing with Kinect. This would make creating the game in terms of dealing with libraries easy and support for the creating relatively high. However there are some downfalls with using Visual Basic. Although I do have experience with it, it is not my preferred language, and although there are graphics libraries that I can use it would involve me going in without any prior knowledge.

I then looked into using Java. An advantage with using java is that I have more experience with it. I have learnt how to use OpenGL libraries and have previously explored other libraries such as the lightweight Java Gaming Library that on top of the use of knowledge built from studying for my degree. Although there are no Microsoft libraries or plugins that would allow for me to use the Kinect, I did find some that had been created by developers, the two most credible being OpenKinect and J4K libraries.

The J4K library was created by a small team consisting of students and faculty in the University of Florida Digital Worlds [2]. This java library works with the Kinect drivers to receive information from the Kinect sensor and allow it to be read in java applications. The information accessible includes a depth map, the coordinates of the player's joints and information from the RGB camera. The library has been refined making it easy to use, giving plenty of documentation and providing examples of its uses for developers to look at.

On the other hand the OpenKinect was a collaboration of over 2000 members who donated their time and code to the project [4]. The project has it's own wiki website which has a sufficient amount of documentation about the different classes and functions within the library as well as a selection of videos showing off what users have done with the library. The library has a section on the

popular website Reddit[5], which would potentially allow me to talk about any issues that I have.

On paper I feel that the OpenKinect library would be the most useful however the Reddit page is not particularly active with very little users posting in the last year. I am also not too happy that so many people have worked on the project and in comparison to a library created by academics I feel that the J4K library would be more reliable.

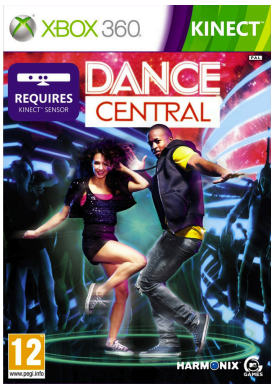
Before I made my final decision I downloaded the libraries and the Visual Basic SDK for Kinect. I wanted to test them out and to see how easy I found them to gather information from as in practice my opinions could be changed. After spending some time testing out each option I came to the conclusion that the best choice for the game is using Java with the J4K library. I found that this library was incredibly easy to implement and get controls from and found the large amount of documentation on setting up and about each class to be incredibly informative.

3.2 Gameplay

Before I was able to start programming I first needed an idea of what my game needs to do, what is the method of gaining points and how the player will win or beat the level. To do this I began doing research into what kind of game was popular and found that in the UK the most popular genres of gaming are Trivia, word or puzzle. With 33% of gamers stating that it is their favourite genre of game. The second most being action adventure and shooter games at 18% and the joint third being sport and strategy games at 10%[6]. Although these figures were important to keep in mind I had to keep in mind the scale of the project and the constraints involved. As the motion controls were a large part of my project I knew that I would be unable to create a game with too many complicated movements and controls or a large storyline. This ruled out the action adventure and strategy games. I then looked at what the Kinect was good at and what controls work by looking at a couple of previously released games for the Kinect.

The first game I looked at was Kinect Sports. This game allows you to simulate six different sports by each sport having its own mini games which to relate to that particular sport [7]. When looking at the controls for each game it quickly became apparent that the motion controls were mainly focused on one body part. For example when looking at the football mini game “Target Kick” the user only needs to focus on the movement of their legs to move the on screen avatar to aim and kick the ball and in the table tennis mini game “Rally Tally” the hands are used to simulate holding a racket to serve and hit the ball.

This simple technique is used through all of the mini games with the trigger being one section of the body and aiming with the rest of your body. Kinect Sports is similar to the game that I am making as like this game I am targeting mine towards all age and gender groups. I feel that Kinect Sports has done this successfully by using a less realistic graphical style and making the movements natural, as if you were playing the sport itself.



The next game I looked at was Dance Central. Dance Central is a dancing game that allows the user to dance along with a large number of their favorite pop songs. The controls for this game are full body and the user will be required to copy the same movements that are on screen in order to score points. Although in the time scale for the project and with my lack of technical ability to dance it would not be possible for me to create a game like this. However I think that I will be able to use a similar framework in terms of the user flow. The game will have a set format for each of the levels, which will be easily copied time and time again, meaning that if I implement a game with this structure I should be able to develop a game with a good amount of levels for the user to play. However I do feel that the graphics of this game would not appeal so much to users of all ages and genders. The graphics are quite busy, and feel energetic from its use of bold neon colours. I personally found that this made the interface difficult to read at first however after a while playing the game I was able to understand what was what. The colour scheme of the game also makes it appear to be more feminine, and although as a dancing game this is most likely to be their main demographic I sound make sure in my game that I stay further away from the colours that might put a specific gender label on it.

After looking at multiple games I have a clear idea in my head about what I want my end project to look like. I would like to make a shooting based game, which like some of the mini games in Kinect sports focuses mainly on one particular part of the body, which will be the users arms and hands. I will set the game out in the same format as Dance Central by having several levels, which are able to use the same classes, but have different positions loaded into them. This should mean that I am able to create a larger scale game.

3.3 Software used

During my project I will be forced to use a couple of pieces of software in order to make the creation of the game easier and less time consuming. I will use a 3D graphics program called Blender. Although I am not familiar with the software

taking the time to learn to use it will allow me to create my own custom 3D objects to use in the game. This will make the graphics appear to be more uniform and I will not need to worry about getting textures or solve problems with objects downloaded from a website. There is also plenty of support for blender, meaning that I should be able to troubleshoot any issues that occur in a reasonable timescale. As I will be making my own models I will also need to make my own textures. To do this I will be using Photoshop as I am relatively familiar with how to use the basic features and some of the advanced ones. This should allow me to create uniform texture files for my objects with minimal amounts of problems and if issues do occur there is also plenty of support for this software.

3.4 Summary

From the research I am able to state that the aim of this project is to develop an engaging target shooting game that relies on the movements of the users arms to aim, fire and navigate menus. I will be able to confidently state that the aim has been satisfied when the project loads at least one level of a game, allows the user to control the game through use of their body and the Kinect and the results of the control that the user has performed are shown on the screen. The game will also need some form of score keeping to insure that the user is sufficiently challenged.

4 The Specification & Design

4.1 User Requirements

To begin I will first write out the user requirements for the game. It is important to define the user requirements, as it will allow me to grasp what the user will want from the game and what the functional requirements will need to be. By setting these requirements for the program now I will be able to refer back to them later on in the process to ensure that I carry out what is needed of the program. To distinguish between the importance of each requirement I will have a rating from one to five with each requirement that will symbolize how crucial to the user experience the requirement is, with one being of lowest importance and 5 being absolutely necessary to the project. The user requirements are as follows:

1	Clear Visuals of the Main Game Peripherals
Description	The user should be able to see a clear representation of the bow, arrow and targets on the screen.
Rationale	The user needs to be able to see what they are doing within the game. This consists of being able to see if they have an arrow loaded, where the bow is currently aiming, and where they need to be shooting.
Importance	5 – This is a requirement that has to be fulfilled for core functionality of the game.

2	Ability to Aim Bow and Arrow
Description	The user should be able to control the movements of the bow and arrow through moving their left arm.
Rationale	In order for the game to work the user will need to be able to move the bow around. As you hold a bow with your hands it makes sense that the bow is controlled with a hand. Without this requirement the user would not be able to hit the targets.
Importance	5 – This is a requirement that has to be fulfilled for core functionality of the game.

3	Use Motion Controls to Shoot Arrows
Description	The user should be able to fire the bow by moving their right arm forward and back once the bow has been loaded.
Rationale	The user will need to be able to fire an arrow at the targets. The action carried out will need to be as natural to the user as possible, however it will also need to be distinguishable as to not cause accidental firing.
Importance	5 – This is a requirement that has to be fulfilled for core functionality of the game.

4	Ability to Re-load Bow
Description	The user should be able to re-load the bow by moving their left arm into their body and moving the right arm forward and backwards.
Rationale	After the bow has been fired the user will need to be able to re-load the bow in order to fire another arrow at the target(s). As this is another action that would usually be performed with the hands it makes sense that the action to re-load is done by using the hands and arms.
Importance	5 – This is a requirement that has to be fulfilled for core functionality of the game.

5	Completing a Level
Description	Users will complete each level by shooting and hitting all of the targets in the level. Once hit the targets will disappear so they cannot be hit again.
Rationale	The user will need some way to win the game otherwise the novelty of playing would wear off. By having multiple levels for completion it will also be possible to add in scores for comparison and other add-ons into the game.
Importance	4 – Although not necessary for the outcome this requirement would result in a large loss of functionality or user experience.

6	Scoring Points
Description	The user will get 10 points each time they hit a target and a deduction of a point each time they miss the target.
Rationale	Adding points to the game will engage the player, encouraging them to beat their previous scores and allowing for the potential to compare scores with other players. The deduction of points also means that players will need to focus in order to hit the target using as few arrows as possible.
Importance	4 – Although not necessary for the outcome this requirement would result in a large loss of functionality or user experience.

7	Exiting the Game
Description	The user should be able to exit the current level at any time by clicking a menu button in the bottom right corner.
Rationale	This function is required for a good user experience. It will give the user the freedom of exiting from the current level without completing it or exiting the game entirely. This option will allow them to go straight to the main menu when they want to in order to continue playing without being inconvenienced.
Importance	3- A non necessary requirement that would cause some loss to the functionality or user experience

8	Summarize the Score After Each Level
Description	If the user clears all targets a summary page will appear showing the score of the current level.
Rationale	To show a level is completed a menu will appear displaying the users score. This will allow the user to review how they did before choosing to replay the level or go back to the main menu to choose another level to play.
Importance	2 – This requirement will cause minimal loss to the user experience if not included.

9	Navigating Menus
Description	The users should be able to navigate the menus using motion controls and be cable to click by hovering over buttons.
Rationale	To make navigating the menus intuitive the user should be able to use their hand to select and click on the buttons that they want. This will be more reliable than using motion controlled selection.
Importance	3- A non necessary requirement that would cause some loss to the functionality or user experience

10	Display All Playable Levels in a Convenient Manner
Description	The game must have a level select screen that allows the users to view a list of all of the playable levels.
Rationale	The user needs to be able to see a fill list of levels to allow them to select the one that they wish to play. These need to be displayed in a clear and suitable manner by having clear labels so levels are found easily.
Importance	3- A non necessary requirement that would cause some loss to the functionality or user experience

11	Background Music
Description	There should be a quiet sound loop that plays when the game is launched.
Rationale	Background music will add atmosphere and ambience to the game. With the right sound loop this will help to make the user more focused on the game.
Importance	1 – Excluding this requirement will have no effect on the outcome of the game.

12	Sound effects
Description	Each time the player completes an action such as firing a bow or pressing a button a sound effect should play.
Rationale	Adding sound effects would confirm a users action in another way that is not visual. It would also add an atmosphere to the game and help the user feel as if they are really performing the actions.
Importance	2 – This requirement will cause minimal loss to the user experience if not included.

13	Scenery
Description	For each level there should be some form of scenery and/or background consisting of sky, ground and potentially objects.
Rationale	Adding scenery will make the game more interesting for the user. It will make the game more visually appealing and this will make the users want to continue playing.
Importance	1 – Excluding this requirement will have no effect on the outcome of the game.

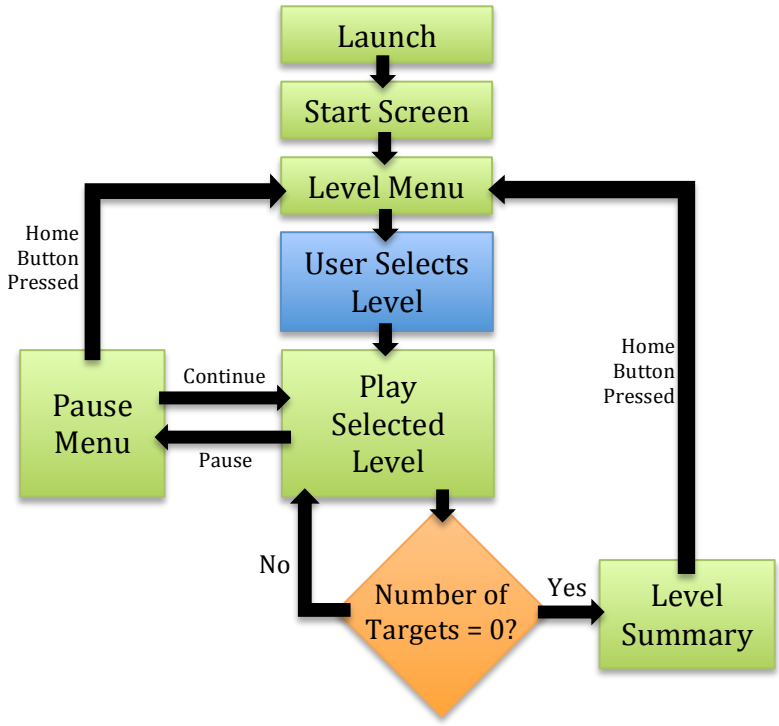
14	Multiplayer
Description	The user will be able to play with or against a friend in either a turn based shooting style, or a head to head shooting with the winner being the player with the most points.
Rationale	With the rise in popularity of multiplayer games it would be worth me adding in this functionality into my game. However this does not detract from the main user experience and would only act as bonus content.
Importance	2 – This requirement will cause minimal loss to the user experience if not included.

15	Social Networking
Description	Each time the player completes an action such as firing a bow or pressing a button a sound effect should play.
Rationale	Social networking is a good way to advertise and get people talking about the game. It would be beneficial in terms of the games marketing for the users to be able to share their scores and convenient for the player to show off their high score.
Importance	1 – Excluding this requirement will have no effect on the outcome of the game.

16	User Feedback
Description	The user should have visual feedback of the actions that they are making.
Rationale	To ensure that users know what is going on within the game they will need visual feedback of the actions they make. This will reassure them that they are not making mistakes and reassures them that they are making the right choice.
Importance	4 – Although not necessary for the outcome this requirement would result in a large loss of functionality or user experience.

4.2 Users Perception of the Game

To gain further understanding of how I want to operate I put myself in the users position in an attempt to figure out what the flow of the game would be. I will then display the result in a flow chart, which I will be able to refer back to when programming to make sure that I am still on track with my original design. In this section I will also describe the flow in more detail in paragraph form. Below is the user flow diagram for the game:



When the user launches the game they should first see a start screen. This screen will show the name of the game and start button in a box with a plain background placed in the middle of the page with the title of the game featured at the top. In the background there will be a 3D rendered scene of a landscape with trees, rocks, grass and clouds. Once the start button has been pressed the user will go to the level menu. There will be the same 3D background with a box in the center however inside the box there will be a list of levels. There will also

be a next arrow to allow the user to the next page of levels and an instructions button that will open up a screen to show the instructions.

After the user selects a level the display will be cleared and a new display rendered. This will be of a different landscape to the menu and will have the same types objects displayed on it with a few more. These will include targets and a bow. The user will then be able to load and fire the bow with the goal to hit the targets. There will also be a display in the bottom right hand corner that displays the score. At any time while playing a level the user will be able to pause the game, which will give them the option to either continue playing, or go back to the main menu via an interface that appears. Once all of the targets have been cleared in a level an interface should appear which shows the users final score, targets hit and shots missed. This interface should also have a button that allows the user to go back to the level menu.

4.3 Kinect controls

The controls of this game will focus on the movements of the player's arms. I will now list through the movements and gestures that the player will do in order to control the game.

Navigate menus: The user will be able to use their right hand to move an on screen cursor. To click the buttons the user will hover over it for a short amount of time.

Load Arrow: To load an arrow the user must bring the left arm close to the body and bring the right arm forward and back.

Aim Bow and Arrow: To aim the bow and arrow the user will use their left arm.

Shoot Arrow: To shoot the arrow the user must move their right hand forward and back. The bow must be loaded in order to complete this action.

Pause Game: To pause the game the user will raise there right hand up and down.

5 The Implementation

In this section I will now look at the implementation of the game itself. This will cover each aspect of the game, from creating the display, adding in the separate game functions, 3D modelling and also controlling the Kinect through the game. As the game contains many different elements that were being made and brought together at the same time I have decided to split up the sections by the respective function to make it clear what I have done for each part of the game.

5.1 Game Foundations

5.1.1 Initial setup

To begin I first needed to set up the file structure for the game. Once I set up my project in eclipse I already knew that there would be several sections I would and keeping each function in separate packages allowed me to keep my work organised from the start and will make it easier for me to find any files as I go along. I then installed the Microsoft Kinect for Windows drivers version 1.8 which I was able to obtain from the Microsoft website. This version number was important, as it was the recommended version number for the Kinect library used within the project. I then made sure I got the newest version of OpenGL 3.2, which would allow me to have more freedom over the programming of exactly how my graphics are and will insure that I am up to scratch with modern standards as fixed function rendering pipelines have now been depreciated within OpenGL. I then installed the Light Weight Java Game Development Library version 2 which I would use for an array of functions throughout the process. Finally I imported the J4K library, which gave me access to the information sent from the Kinect such as body positions, depth and access to the RGB camera.

5.1.2 Display

To create the display I first set up a display manager class containing three methods. The first method I created needed to create the display. For this project I used the lightweight java gaming library's OpenGL display, so I first needed to set up the attributes of the display by using the built in function and then setting the width and the height of the display. From there I used the built in functions to create the display using a new pixel format and the attributes that were set earlier. I then added in some visual window settings, which consisted of the window title and the icons for the taskbar and window. To make the icons I used Photoshop to create one 32x32 pixel image for the icon and one 64x64 pixel image for the taskbar. As the game is a shooting game that used targets I thought the most appropriate image for this would be a simple image of a target. The final step for this method is to add the OpenGL viewport to it. This will be the section that will display all of the graphics and, as there is no reason for me to display anything outside of the OpenGL viewport I set this to the size of the window.

The second method that was created was the update method. This will update the viewport with the updated graphics. This happens by first syncing the

display, which will take in the maximum FPS, and then using the built in function to update. The final method used in the display class in the close method. This simply calls for the display to be destroyed. Each of these methods is public so they can be accessed from anywhere in the code.

5.1.3 Player Perspective

To make it easier to set and alter the view of the player I made a new class in the Entities package called camera. The only information this class will be used for will be to hold the position of the player and will allow the user to get this from a get method. To make the game easier to debug and to create levels I also put in some controls. Although commented out the move method, which requires a texture as an argument, will listen for keyboard controls and move in the corresponding direction. I based the controls of regular game controls with 'W' being forward, 'S' being backward, 'A' being left and 'D' being right with the Y position of the user being set to the terrain height which has been given to the method. To update the view on screen a view matrix is needed, which will be used in the shader files to determine what the player can see. To create this I first created a new method in the maths class with the arguments of a viewer position. The x, y and z co-ordinates are then inverted and a new matrix is created and translated by the negative viewer position. This means that in the shader class I will then be able retrieve the view matrix and apply it to all of the objects to move them to the left, right, forward or backwards which will give the illusion of movement. To make the players view more realistic I also applied a projection matrix to the shader, which will set how far back the user can see, what the field view is and near plane. These are set in the renderer classes and are put into the shaders to provide better looking depth perception to the game, which will be useful to help users judge their movements on screen.

5.1.4 Shaders and Lighting

The game needed to be lit and colour displayed in order for the game to have visual appeal to the user by giving the objects that are rendered shadows and texture. I first created a new package to keep the shaders in. I then created new text files, one for the vertex shader, which will deal with the light of each triangle, and the second for the fragment shader, which will be used to calculate the colour of each pixel. These shaders will be written in GLSL.

To create the vertex shader I first needed to define the GLSL version number at the top of the text file. For this game I will be using version 400 core. I then had to define what variables the shader takes in, what the shader will output to the fragment shader and then the variables which I will need to set in the shader program which are the transformation matrix, projection matrix, view matrix and light position. I then get the world position, the position of the object in relation to the camera and the glposition by doing some simple multiplication. Next I retrieved the normal of each vertex and checked if the object needs fake lighting, if fake lighting is required then the vertex normals are all set to point vertically. The normal are then translated the same way the objects are in order for them to face the correct direction and the vector from the light to the camera is found by taking the world position away from the light position. The variables

which are outputted from this shader are the texture co-ordinates, which have not been modified in this shader), the surface normal and the distance to the light.

The fragment shader starts off in the same way the vertex shader does, stating the version number, followed by the inputs from the vertex shader, the colour of the current pixel as the output and finally listing the variables which are to be set in the shader program. This shader will first normalise the vectors passed through from the vertex shader. Then the angle between the normal of the vertex and the light is calculated by using the dot product. This will return 1 if is identical, 0 if it is at 90 degrees and in-between it will return the value between those numbers. This will be used to calculate the brightness of the pixel as we can say that the closer the angle is to zero, the brighter the pixel will be. I then set a lower limit on this to 0.15 by using the GLSL function max. This would prevent shadows from being completely pitch black which would hide some of the detail and look unappealing to the user. I will then retrieve the colour from the texture file and multiply this by the lighting that I just calculated. This will return a colour value for the pixel.

I created the shader program abstract class by first creating a constructor which will take in the name of the vertex and fragment file. These will then be loaded as light weight java gaming library OpenGL vertex and fragment shaders. To do this I created a method that creates a new string builder and loops through each line of the text files adding them to the string builder. Once all the lines have been read a new shader is created of the vertex or fragment shader and the corresponding file is attached to it. The shader is then compiled. Once this is done for both shaders I then created a new program and attached the shaders to the program, which is then linked to the program id and validated. To finish off the shader program I added in start, stop and clean up methods, which are public so the shaders are able to be started, or stopped throughout the program. The final addition to the class is a set of methods to set the variables that are used in the shader. For each of the bind attributes methods the location of the variable in the shader is needed and the value that it is set at. Once called the methods will simply set the uniform variable however as GLSL has no Boolean variable I will set the float 0 to false and 1 to true.

The final class I created, called the static shader, extends the shader program and is used to fill in all of the data for the shaders. I first hard coded the file locations as string variables and created a constructor to pass them into the shader program. I then added load methods for each type of input which take in the variable the shader needs to be set as. The other variable required in these methods is the location of the variable in the shader. To do this I created another method which used the shader programs getUniformLocation, which needs to be passed the exact same variable name as is in the vertex and fragment shader, and this will set a variable of the location in the class for the bind methods to refer to. Finally I wrote a bind attributes method to bind the attributes to the variables.

To create the lighting I first made a new class which would be used to define a light in terms of position and colour. I then made sure in the static shader the

light was being loaded into the shader classes. As stated above in the shader classes the normal will be used to work out how much the light effects the face and this will be used to lighten or darken the colour. The light position and colour would be set in the game loop.

5.2 3D Modelling

5.2.1 Setup

The first thing I did to create my 3D models was create a raw model class. This would be used to store the number of vertices and the ID of the models. This will be used later to identify the vertex array object(VAO). Next I created a new class called Loader. This will be used to deal with VAO related functions such as creating, storing data in them and unbinding them after use. The first method I created was the loadToVao method. This took in arguments of three float arrays, one for the positions, and another for the texture coordinates and also one for the normal of the vertexes, along with an integer array, which contains the indices. The first thing I did in this method was create a new VAO. To do this I made a new method called create VAO, which makes a new VAO, binds it so we can start editing it and returns an integer value, which is the id of the new VAO. The next task is to bind the indices buffer to the VAO. To do this I first created a vertex buffer array (VBO). To do this I called the built in function in OpenGL to make it for me, used the bind buffer to define it as an element array buffer and edit it. I then converted the buffer into an integer buffer and stored the indices array in it. To store the data for each vertex, texture coordinate and normal used I needed to create a new method. This method took in the attribute number in the VAO, the size of the co-ordinates and the data as a float array. For each piece attribute I need to create a new array buffer and convert it into a float buffer. I am then able to store them by a method, which I created which uses the built in function 'put' to add the data to the buffer. I then simply place the buffer into the VAO in the position that I assign. This method is called for the positions, texture coordinates and normals storing the positions in the first VAO, the texture coordinates in the second and the normals in the third. A raw model is then created and returned with the ID of the VAO and the length of the indices assigned to it. To finish I created a clean up method which loops through the VBOs and VAOs and deletes them. This method needs to be called when the game is closed to avoid issues.

To make rendering easier I created a class to hold all of the models and the details of their positions and rotations called entities. The constructor for this class will take in the ID of the object, the model as a Textured model, the position as a vector containing three floats, the rotations in the x, y and z directions and finally the scale of the object. The variables passed through are then set as local variables and I created getters and setters for each of them. Finally I added two methods to increase the position and rotation of the object.

5.2.2 Creating 3D objects

In order to add 3D objects into the game I first needed to get object files. These must include the positions of each vertex, the texture coordinates (UV coordinates) and the normal of each vertex. To do this I created my own objects in Blender. This involved me manipulating basic shapes and making textures to form a tree, a rock, a bow, an arrow, a target, some grass and a cloud. I only made one of each type due to the time constraints. I then made sure to export the files as a Wavefront (.obj) file selecting the data that I required from the file.

To add in the 3D object into the game I first created a new class called objectfileloader with the purpose of taking in the object file name and output a model to a VAO. The first method I created takes in a string which will be the file name of the object that is being loaded. This will then be used in the method to load the text file to a file reader, which is then made into a buffered reader. This is so I can then loop through each line of the buffered reader to look what the first letter is. This letter will tell me what information is on the line, allowing me to split the string up on the line and take the values needed and place them into a float array for me to access later. There will also be several lines for faces in the code and one the code has looped through enough to find the faces a different extraction method is needed. Each face consists of a triangle represented by 3 3D points with the points separated by a space and the coordinates by a forward slash. To retrieve the data I first split the string up by spaces and the 3D points in an array. Then for each item in the array I split it by the forward slash and stored each point in another array. Each vertex is then processed, getting the normal, and texture for each vertex and arranging them in the correct order for them to be read with the vertex. Once this is done the reader is not needed anymore so it is stopped. The vertex points are then loaded into the vertices array. The model is then put into an instance of the Model data class and returned.

5.2.3 Adding Textures

For the models to look realistic textures needed to be applied to them. To apply textures to objects I first created a new method in the loader class that takes in the filename of an image to be used as a texture as an argument. I then use the lightweight java gaming library's built-in texture loader to read in the image file as a texture. This texture is added to a list, which will contain all of the games textures, and the texture ID is then returned. I also added a method to delete all of the textures in the list in the "cleanUp" method.

I then needed to create a new class to contain information about the texture. This will only require two variables, which are the texture ID, and a Boolean that represents if the texture uses false lighting. The method then needs getters and setters for both of these variables. I then created a new method in the models class called Textured model that will store the raw model and the textured model. I then went back to the loader class and made sure that the code was implemented to take in texture VAOs, then checked the shader files to make sure that the texture coordinates were being passes straight through to the vertex shader then being used to then output the colour and that the colour coordinates are being set in the static shader.

5.2.4 Terrain

To create the terrain I first created a new class called terrain. In this class I first initialised some variables that would need to be used throughout the file. I set the size of the terrain, the maximum height or depth the terrain can be, the maximum colour a pixel in the depth map can be and the X and Z position of the terrain in the world. I also created blank variables to hold the raw model of the terrain, the terrain texture and a 2D array that holds the heights. In the constructor the loader is passed through, along with the texture and the filename of the height map. The texture is set to the texture made earlier. I then made a new method that would take in the loader and the height map and return a raw model. I called this method generateTerrain. In this method I first used a buffered reader to read in the height map image. A new 2D array is created the same size as the image's dimensions as the image height will be used to represent the number of vertices and new arrays are created to store the data for the creation of a raw model. In order to populate these arrays I needed to create a new method that would get the height of the terrain from the image. This method takes in the X and Y value and the image as a buffered image. Then the value of the pixel at that point is retrieved, converted to a figure between the maximum and minimum height and returned. This means that back in the generateTerrain method I could then loop through each vertex and set the X, Height and Z positions, retrieving the height from the previous method.

To get the normal for the terrain I created a new method which took the height of the surrounding pixels and took the right from the left for the X value, set the Y as two and took the top from the bottom for the Z value. This allowed the ground to have shadows. To finish the terrain I set the texture coordinated and the vertex points as the x and y. All of the arrays created were then sent through to the LoadToVAO method in the loader to create a raw model. The call to the generateTerrain method is in the Terrains constructor and sets the raw model produced to the model variable at the top of the class.

Before I can call the texture I will need to make separate shaders for the Terrain. This is because instead of taking in an Entity the terrain shader class will take in a terrain. However as I have done this all before it means I only need to copy and edit a couple of files. I first made a copy of the staticShader, vertexShader and fragmentShader classes. The new staticShader was renamed it to TerrainShader and removed anything that related to using the fake light.

5.3 Rendering

The first class I created was the renderer class. This class contains the methods that will be used when rendering both terrains and objects in the game. The first methods I made were the constructor , 'prepare' and 'cleanUp' methods. The prepare method enables the depth testing to check what triangles are in front of others, then it clears the colour buffer, then it sets the background colour to blue which is the colour of the sky. The clean up method is called when then program is closing to clear the entity and terrain shaders. The constructor method

disabled rendering the side of the shape that can not be seen in order to speed up rendering and created the projection matrix.

Next in this class I needed to define a hash map, which would use the textured model as the id and a list of entities as the data. I then made a method called 'ProcessEntity' which takes in an entity, gets the textured model from it. I then attempt to retrieve the textured model from the hash map, if this returns null I know this is a new model and add it to the hash map as a new item, if it returns a list of entries I simply add the entity to that list. The next method I created took in the light and camera, and will be used to set the variables in the staticShader class. The first thing the class does is call the prepare method that I created earlier. The method calls the start method in the static shader, allowing me to set the variables in it. I then load the light and camera into it. This method will be called in the game loop after the ProcessEntity method has been called for all entities. This means that the hash map I created will contain all of the entities that I need to process. At this point I will feed the hash map into a new method in a new class that specifically renders entities. The shader is then stopped and the entities hash map is cleared. This is then repeated for the terrain shader, however instead of feeding a hash map to a new method in a new class for entities I will instead feed a single terrain file into a new method in a new class that renders terrains.

I then decided to create the new class that renders entities. The constructor of the new class takes in the shader and a projection matrix. The shader is set as a private class variable and the projection matrix is applied to that shader. The second method will take in the hashmap of entities and this is the method that will be called from the ProcessEntity method in the renderer class. The method will then loop through the keys and retrieve all of the data for that model and binding it to be edited. After retrieving the data it will getting the list and for each entity in the list apply the transformation matrix and draw the entity. Once it has finished it will unbind the TexturedModel.

Now the renderer for the entities was complete I needed to do the same for the terrain, so I first duplicated and renamed the entityRenderer. I then modified the renderer class to take in a terrain and removed the contents of the method. I then called the prepare method which is passed the terrain and binds it so we can use it. Then I applied the transformation matrix to the terrain and drew it. I finished by unbinding the terrain.

5.4 GUI's

To create the on screen labels, menus and cursor I needed a new set of renderers, textures and shaders. I first created a package to hold these classes and inside made a new class called the GUITexture. Each GUITexture will have a Texture, a position, a scale and a boolean that states if the GUI is visible. In the constructor the class the variables are set and the rest of the class consists of getters for all of the variables and a setter for the boolean.

As the guis will not require as much information as the entities I created a new method in the loader class that only requires the positions of the GUIs. This method will load the GUIs to the VAO. The method creates a new VAO, stores the positions in the VAO and then creates a raw model.

Next I copied the staticShader, fragmentShader and vertexShader, placing them in the new GUI package and renaming them. I then cleaned them up by removing all of the variables that I would not need, which was all but the transformation matrix and the binding of the position variable.

Next I created a renderer for the GUIs. In the constructor for this class I first set the positions for a quad that I then loaded into a VAO to return the raw model and I initialised the GUI shader. I then created a clean up method that would be called to clean up the shaders when closing. The last method I created was the render method that takes in a list of GUIs. The shader was started and I set the settings in OpenGl to start rendering objects and to allow for the GUIs to be transparent. I then looped through the GUI's to check if they were visible, if they are bind the texture to the model, scale and translate it and finally draw the GUI. The method is finished by stopping the shader and putting the settings back to the usual settings after all GUIs have been processed.

5.5 Kinect

The J4K library made it easy for me to gather the data I needed from the Kinect sensor. I first created a new class called "Kinect" which extends J4KSDK. This class would be used to get all of the data from the user, process it ad set variables in its class for the game methods to retrieve when needed. As the J4KSDK is an extended there are a few methods that are automatically included. However as I am only interested in tracking the skeleton of the user I started working on the onSkeletonFrameEvent. I first did some checks to make sure that a skeleton was in fact being received and tracked. I then got the positions of various body parts such as arms, shoulders and the height of the head.

Next I worked out the angles that the left arm was pointing in by getting the point of the shoulder and the shoulder closer to the Kinect sensor. I then used these two points and the location of the hand to work out the angle the arm is facing by using the built in java class atan2.

I then started to look at recording the movements of the user and turning them into commands to give to the game. The first thing I did to do this was to work out the difference between the current position of the right hand and the previous position. I separated the coordinates to do this so I would be able to tell which way the strongest movement was in. I did this by comparing the values of differences to each other and setting a string to the name. When I first placed this in the code and tested it I found that small movements were also being recorded, so on each movement I added in a value that the difference between the new and previous positions must be above. This meant that only bigger movement were counted. Once the direction is found it is added to a hash map, which gets the id to the movement and the value to a tally of the number of times the movement

has been recorded. For every ten pieces of data gathered the movements are then reviewed by looping through each entry of the hash map, and getting the movement with the highest tally. It is then added to a list containing all movements.

After adding the data to the list the next step is to check for actions. The first action it looks for is if the bow should be fired. The code first looks if the player's movement forward in the last ten bits of data has been forward. If it has it looks if the bow is currently aimed and if the last ten movements have been five or more forward motions suggesting the user has thrust his or her arm forward. If all conditions are met a boolean is set to true to fire the bow and one false to remove the arrow from aiming.

The next action that the code looks for is loading an arrow. First there is a check to make sure that the previous move had been recorded. Next the code checks what the move is not and the previous move. If the previous move is forward and the current one is back and the left hand is closer to the body than the right, then the bow should be loaded. While checking if there is a previous move I also added in the code to check if the user wants to pause. This is exactly the same principle as the aim, however this time instead of looking for a forward then back motion it looks for an up then down motion. When either of these are true a variable is set in the code to be read by the game function.

5.6 Game Structure

To create the game I first started with a launch class, which will be the first class loaded in the game. This file would create a new instance of the Kinect, loader, all of the levels, the display, main menu and game loop. It also starts a variable "keepPlaying" which while true keeps the user in the game. While keep playing is true the user will enter into different game loops depending on the level they are playing. After loading the main menu the first loop that the user will be forced to enter is the main menu loop. This will launch the menu for the user to select the level they want to play and return a value depending on the level they select. This value is then used to satisfy the 'If' statement of the level the user chooses, loading a level then launching a game loop. Once the user has finished the level the loop will start again and the player will enter the main menu.

In the method to load the main menu the program will gather all of the data stored in the level files and load the 3D objects using the methods and classes that have been created previously to load the model, then load it to a VAO, then create an new textured model, then apply the light settings. The data is then looped through and a list is created holding all of the entities. Once this is complete a new light, camera, renderers and loader is created and the guis are added to a list specifically for GUIs.

In the method to play a check is first done to make sure that there is a Kinect plugged in. Next the position of the hand is retrieved from the Kinect class. And a new cursor GUI is added to the list of GUIs with the position of the hand. Next the entities are passed to the renderer. However if the entities are clouds I apply

movement to them in the left direction. Once the reach a point off screen I then move them to the far right of the screen. Next I translate the cursor position into a pixel position and work out what button it is on. The button positions are hard coded into the main menu currently. To click the user needs to hover over a button and while in the space of the button a mouse timer will increase. Once it reaches a certain number the score of the button is returned indicating the level.

In the main menu the level will be loaded to the game loops load method. This will do exactly what the main menu load method does; however it requires more entities for the bow, arrow and targets. However the playGame method in the game loop is different. After entering a game loop and having various variables set the code will pass the entities into the render, and just like the main menu move the clouds. The targets are then sent to the entity renderer seperatly, the terrain is sent to the renderer and then the GUIs. The hand position is then retrieved and the pixel position found. The code then checks if the cursor is visible. There are only two occasions when the cursor is visible and that is when there are no targets left and the score is showing, or when the game is paused. If there are targets left the game is paused therefor we can check if the cursor is in the regions for the pause menu buttons and do the same code implemented in the main menu for hovering and clicking. If we know the game is not paused and there are targets left then the user is playing the game. To do this we first set the rotation of the bow equal to the angles retrieved in the Kinect function and send the bow to be rendered. The code then gathers the data about the controls from the Kinect, i.e. if it is being aimed/fired/shot. If the bow is being aimed we need to get the angle of the projectile in order to angle the bow. This is done in two parts, one for the right and left, and the other for the up and down angle. To work out the angle of the projectile I used simple trigonometry to work out the opposite and the adjacent values for each. I then generate a new instance of the bullet class, set the position and rotation in line to that of the bow. I then use the opposite/adjacent for both the up and down and left and right to work out the scale at which the arrow should move when it moves.

If the action is carried out to fire the bullet a method in bullet called fire is called. This takes in the ratio to move in the x and y directions, the targets and the terrain. If this method returns that a target is hit the score is updated with five points if it hits or ten if it hits a bulls eye. If the bullet misses a tally is counted of missed arrows, which get deducted once the level is complete. A GUI is then set in the bottom left to display the updated score.

If there are no targets left then the cursor appears along with a score review GUI that shows the player how they did. The cursor uses the same code as before to check if its hovering over buttons and clicking however with updated locations of the buttons for the score sheet. The guis are rendered by separating the number that make up the score and reading in the separate number images.

When the loop finishes the lighting is added to the renderer with the GUI's.

The bullet class holds the variables that describe the different states of the bullet. The fire method inside the class will move the arrow, check if it hits a target by

checking for every z value a target has if it is in a circle radius of the target by using the square distance. To work out the bulls eye it does the same however with a smaller circle. If a target is hit the target is removed from a list in the game loop and the arrow is destroyed. The arrow is also destroyed if it goes below the ground or too far back causing the player to loose points.

The level files are simply classes with variables that state the positions, rotations and scale of each Entity, the camera position and the GUIs needed in the level.

5.7 User Testing

As I approached the end of the project I carried out some user testing in an attempt to further improve my project and to find bugs that exist in the system. To do this I first wrote up a small survey, which started by asking some basic questions about their age and gender. I then asked the user what they thought of the game concept and if they felt that it was suited to the motion controls, then I asked them if they felt the controls responded well to their movements. I then asked the user what they like most about the game, what they like least about the game and the final question asked what they would do to improve the game.

When choosing the selection of people I tried to get a wide range of ages and genders, however on short notice I did find that I had a limited selection to choose from. Before testing I told the participants about the game, went though the controls and discussed why they were here and what I needed out of their opinion which was insight into how to improve the game.

From carrying out this user feedback session I was able to determine a few things about the project and what needs to be done to enhance it. From the first question, which asked the user if they felt that the concept of the game suited the controls, I received 100% positive feedback from people saying how they felt that the idea of the game was well suited to motion controls, with some saying that the controls “suit the game well” and that the game was different.

The next question asked the users if they felt that the game reflected their movements accurately enough, and which some simply said yes, it was apparent that more work needed to be done on the controls as some players felt the controls “could be better” as they had to slow down their movements in order for them to be recognised and others needed more practice to get the hang of the controls.

When asking users what they liked about the game I had a good response about the graphics, people seemed to respond well to both the style of them and the moving clouds in the background. I also had positive comments about the design of the controls, being told that they resemble using a real bow. The interface was also praised for being clear and easy to read.

When asking users about what they didn’t like I received some insightful comments. Some users found it difficult to follow the arrow, others complained that there was no tutorial and there were complaints about the cursor not

providing enough feedback when hovering over buttons. I also received some comments on things that the software didn't include such as that there was no multiplayer and no left handed controls. These criticisms reflected in the what could be done to improve the game with most users opting to comment that adding in the features they feel are missing or could be done better should be improved.

After reviewing the feedback I decided to address some of the issues that came up. I first altered the Kinect class so that the angles created on the left arm are 1.2 times bigger. This means that it is easier to aim, as the left arm does not need to be so far over. Another issue I was able to address was the user feedback when hovering over buttons. I was able to implement a quick bit of code into the play and menu classes that put a ring around the cursor to show that the selection is being picked.

5.8 Problems

During the project I encountered many problems and difficulties that forced me into making alterations and made the outcome of the project different to my original plans.

The first of these problems was with the 3D objects. When initially looking at object I found an abundance of them, which I thought I would just be able to use and reference after making small adjustments. However after implementing the files to import the objects and textures I soon realised that it would not be that easy. I found that while getting the vertex points and importing them was not a problem, the textures did not appear to download in a format that was easy for me to convert. After attempting to extract the textures and set the textures differently in the game I decided that it would probably be quicker if I made my own models and textures. This restricted me to less realistic looking models, however I would not have the issue of attempting to find objects that were of the same style. Unfortunately this took time out of my project that could have been spent elsewhere, however it has allowed me to develop new skills.
Another problem that arose was with the Kinect. Although there was no issue installing the Kinect I did have some issues with the movement. From the user testing I found that several people were struggling to find pick up the controls, either moving too quickly that the movements where not being picked up by the code I implemented or moving too slowly that the code doesn't recognise the action. Although some users eventually got the hang of the code there were also some that didn't. Unfortunately when looking into how actions in Kinect are programmed I found very little information, so I looked at my code again. I found that the issue was where I was reading in the values. Currently the game takes in ten values and finds the most common movement over a short period of time. This mean that when the users were moving too quickly the most common movement would contain two of the movements that are required for the action. It also means that when users are doing the motion too slowly that the action is not recognised as other movements are placed in between the movements, which will cancel out the action that the program is looking for. To try to fix this issue I lowered the amount of time before summarising the action. However this

was not successful as I found that the user was required to be incredibly quick in order for the actions to work, which would mean that when the users who were slow at picking up the actions played they would not be able to use the controls. I then reset the controls back to the original, as even though players weren't able to instantly get the controls the majority were able to get the knack of them eventually making the game playable. Although through my research I was not able to pick up anything that could help me I do feel that more research is needed to make sure that the controls are better picked up. This would ideally be in the form of more testing and fine-tuning how the software handles the data received from the Kinect.

A problem that I found with the Kinect was sometimes when the player was idol controls were still being picked up. The solution for this the same as the previous problem. The controls need more time to be fine tuned to make sure that problems only occur in extremely rare cases. I also found when looking into the problems with the Kinect that the skeleton retrieved from the Kinect was not accurate. At times for a split second the skeletons body positions appeared to be distorted. I looked to an example that had been provided with the J4K library and found the same issue occurring. Although I would assume this is the result of me using a third party Kinect library the cause of the problem could also be the lighting in the room or the position of the sensor. This could certainly be a contributing factor to the bow sometimes shooting when the user is idol.

I also struggled a bit when creating the shaders for my game. This was a new skill that I was learning and I was not familiar with creating a shader program. The issue with the shaders were that they needed to be created in text files. Although this is not a problem in itself it did mean that debugging them became a challenge. This did mean that on occasion when an issue occurred in the shaders a large amount of time was spent attempting to correct it.

6 The Results and Evaluation

The output of the project was planned to be a Kinect game, which uses the movement of the user to control in game actions. I believe that I have completed this goal as I have created a level based game in which users get points for shooting arrows at targets from a bow which is aimed by moving the users left arm and fired by thrusting their right arm forward. I have also added in extra controls to reload the bow and for menu navigation.

To test and evaluate the game I did a user testing session, which proved to be valuable to the game. I was able to gather data directly from the people who would be the target market for the game and from that was able to make several slight adjustments in the remaining timeframe. I feel that more user testing would benefit the game, as it would allow for an even greater insight into what can be improved within the game. It would be also useful to observe how people use their body in the game to aim and fire the bow. This could allow me to make the controls more reliable as currently the game is best programmed for me.

Although I am happy with the outcome I do understand that it could be better. I feel that the things that I have done well in the design of the program. It utilizes the OpenGL and Light Weight Java Game Library well to efficiently run the game. This is show by its remarkable quick load time and limited amount of loading and creating graphics. However I do feel that the game would have been easier to create had I done it in Visual Basic or another language that the official Microsoft Kinect SDK supports. The only issue with this is that I would have had to have learnt a lot more, meaning that in this situation I would not have been able to do as much due to the time constraints.

The final product is similar to what I designed with only a few small differences. The first is that I only have one level page. Although I did plan on having larger buttons and multiple pages time constraints meant that I had to adapt the design to one page. The other difference from the design is that the user does not go back to the level page after the selecting to go back to the home page. Instead this goes back to the start page.

7 The Future Work

The project will still need some more development in the future in order become a better and more efficient game. One of the aspects of the game that will need to be developed further are the features in the game. I think that the game needs more options to keep the gameplay from getting stale which include adding different target types, adding extras such as power ups, and allowing the users to select from a range of weapons.

The game can also be improved by improving the graphics. Although the users responded well to the current graphics there is currently only a small range of objects in the selection. It would be more visually appealing for the user to have different objects, and even animated objects that move as the game is being played. By adding more graphics it will also be a good opportunity to increase the amount of levels to showcase the range of objects. This could also mean that different settings and landscapes can be created from the project and it would be possible to implement with very little effort. The current game also does not currently feature any in game instructions. A tutorial level or an interface showing the controls would be beneficial to the game.

Another feature that would make the game better is adding in a multiplayer function. Based on UK statistics in 2011 approximately 72% of gamers played online games with strangers and 81% with family or friends [1]. There are several ways the game would be able to feature this including adding in league tables, allowing users to publish their score to social networks and incorporation the 2 player functionality that the Kinect supports. The result of this would be a more marketable game and would encourage the player to replay levels.

The code for the game could also be improved. The game currently only looks for collisions with targets, however it would be more realistic if the game looked for the arrow colliding with all objects. This would allow for layering object in front of targets, leaving partially exposed targets. This could lead to a more accurate game. As the game is currently in development and it has only gone through one stage of user testing it would still need a lot more to be satisfactory for release. It would be important that all bugs were found and removed prior to release to not deter players.

Kinect functionality could also be improved. At the current time the Kinect does not count for it to be placed at extreme angles. This means that if the Kinect is in any other place other than being at chest level and in front of the user it will not work. The Kinect functionality can also be improved by making the actions more reliable. This will mean that the user is less likely to become frustrated when actions are not being performed, or are being performed too soon. It would also be good to have more actions in the game. This would make it more interesting and engage the user more. An example of another action could be dodging enemy attacks by moving the whole body left or right or ducking. The Kinect that I have used for this project has been the original Kinect for windows, however there has been a launch of the Kinect 2, which would have improved functionality and

allows for more points of the body to be read such as the hands. This would be useful to make better and more accurate controls, such as releasing a clenched fist to release the bowstring.

8 Conclusions

In conclusion, when reviewing my final result I am able to see that the game meets the primary aims that I had set at the start of the project. I was successfully able to implement a game with eight different levels, which I designed personally. The game also supports motion control for all actions once launched, with the controls being simple and intuitive. The game also has a clear interface that allows the user to see precisely what they are doing and how they are progressing in the game.

Although my primary aims were achieved I did not manage to complete all of my secondary aims, however I do feel that I was able to make good headway on several of them. Based on the user feedback I received I seem to have created a visually appealing interface. Users of all age ranges and genders seemed to respond well to the consistent graphic style and the low polygon look I created. I was also able to implement a variety of different levels with different difficulties. However I was not able to add in audio commands, sound effects, customization or multiplayer features into the game.

I believe that with more work this game could be further improved and made to be better and more exciting for the user, however I feel that I have laid a good foundation for the game to be built upon and to be expanded. I also feel that the game would require a significant amount of testing before it could be ready to be sold. With more time and effort I have no doubt that the game could eventually become something that is marketable to a wide audience of gamers.

9 Reflection

When looking back on this task I have learnt a vast amount in terms of both programming skills and soft skills which I will be able to take on in the future to help with projects and other assignments I get.

In terms of programming I have learnt a fair amount about using uncommon technology within a piece of software. I have learnt the best was to approach a task dealing in new technology and I have made mistakes this time around that I am able to learn from. I have also learnt that research is key to implementing a system with new technology and it is important to look at examples of code and try out several different solutions before settling on one solution, which maybe regretted later.

I have also learnt a lot about game development and what is required to make 3D environments that look good and load quickly. I have spent time improving my OpenGL skills by learning how to make my own shaders and also developed new skills such as creating 3D models. I have also learnt the importance of user testing when working on a product like this and how user feedback can shape a product to become a polished piece of software.

In terms of my non-technical skills I feel that I have learnt just as much. I feel that my time management skills are better as I have learnt to better break down and prioritize tasks in a project. In the future this should lead to me planning and executing better projects in a more efficient timescale. My adaptability and problem solving skills have also improved as I was faced with several new challenges with new technologies, which I was able to look at logically and creatively to work around.

On the whole I feel that the project has helped me become a more rounded developer by challenging both my technical and non-technical skills. Although the project still needs some development I am happy with what I have achieved in such a short time frame.

10 References

10.1 Websites Referenced

[1]http://www.iabuk.net/blog/10-uk-video-game-audience-stats

[2]A. Barmpoutis. 'Tensor Body: Real-time Reconstruction of the Human Body and Avatar Synthesis from RGB-D', IEEE Transactions on Cybernetics, Special issue on Computer Vision for RGB-D Sensors: Kinect and Its Applications, October 2013, Vol. 43(5), Pages: 1347-1356.

[3] http://www.theguardian.com/technology/2014/sep/17/women-video-games-iab

[4] http://openkinect.org/wiki/Main_Page

[5] http://www.reddit.com/r/openkinect

[6] http://www.statista.com/statistics/240757/games-played-in-the-uk-by-type/

[7] http://www.rare.co.uk/games/kinect-sports

10.2 Image References

Kinect Sports:

http://www.gameplanent.com/Websites/gameplan/Images/Kinect%20Sports%20Game.jpg

http://xbox360media.ign.com/xbox360/image/article/113/1132319/kinect-sports-20101103060400488-000.jpg

Dance Central:

http://firsthour.net/screenshots/dance-central/dance-central-cover.jpg

http://i.kinja-img.com/gawker-media/image/upload/s--uIe3GkFs--/18j2w3zj50bh8jpg.jpg