GOLDSMITHS, UNIVERSITY OF LONDON

IS53007C Project in Games Development

OnSight53

By

Tien Vu

BSc Games Programming

Student ID: 33401574

Supervisor: Jeremy Gow

15th May

Git repo: https://github.com/cakeydoodoo/Final-Project.git

Blog: https://www.tumblr.com/blog/tienfinalproject

Abstract

OnSight53 is a tech demo that showcases a combat system where the player is able to freely navigate in a closed world in which enemy NPCs follow the player and attacks them until eventually killing them. The objective of the tech demo is to test the four different weapon classes and attack combinations to combat enemies which will in turn give you points similar to retro arcade games. The purpose of the demo is to show a combat system for a future RPG game, which means it does not have a story resulting in no progression. There is no win condition which means the demo can last forever until the user either dies or quits the demo.

Contents

Chapter 1

Introduction

1.1 Summary

In order for a game to become a success, I believe that it requires a good balance between mechanics/gameplay and a good story, more so the mechanics, especially in the case of a game where fighting is one of the key components of the game. This is the very reason why I chose to create a tech demo as opposed to creating a full interactive game with a story. Games take years to release because they are constantly trying to improve their system, perfecting it until the creators are finally satisfied with what they have. Of course, the same cannot be said about games that are mostly driven by the story even with minimal mechanics. However, I felt that incorporating a story right now before having a polished game mechanic would make my game less enjoyable.

The core feature of this tech demo is the weapon system; it consists of 4 types of weapons, each sporting its own set of attack combinations which I will go into more detail in the chapter 3 'Design and Implementation'. You can find what the inspirations for the demo in the following chapter and how I implemented it along with how I plan to use it as the core for a future game in the chapter 5 'Conclusion and Future Works'. I will talk about the how it went from the initial design to what it became by the end of the project the in chapter 4 'testing and evaluation' along with the overall changes that I was forced to make in the sub heading 'changes' in chapter 3.

Chapter 2

Background

2.1 Literature Survey

As mentioned in the introduction, I believe that the success of a game depends on not just its story or even its end goal but its gameplay mechanic especially (obviously) in the case of a game that involves combat. Even a game such as Final Fantasy X, one of the most well-known of title of the Final Fantasy series, which is vastly popular for its powerful story, I feel that it would not have succeeded without its iconic turn-based mechanic and Sphere Grid levelling mechanic; which is the main reason I chose a combat tech demo for my project, to create a combat system that could be used for a future game. The combat system was inspired by a game that I played as a child, Kingdom Hearts; its combat style allows the player to change the attack combinations and its order allowing endless attack possibilities. In my demo, rather than allowing to change attacks, similar to the game Nier: Automata where the player has a choice between two types of weapons which can be changed mid combat, I wanted the player to have a choice of four weapon types: Greats Sword, Sword and Shield, Twin Daggers and Hand to Hand, each with its own set of attack combinations that is interchangeable mid combat.

The reason I chose to do this system was that I felt the original gameplay for Kingdom Hearts lacked the customisability that Nier: Automata had mid combat however, Nier: Automata lacked the combination customisability that Kingdom Hearts 2 had out of combat. I wanted to, in effect combine the two systems together allowing the player to change weapons mid combat which had its own set of attack combinations/commands however, said attack combinations could be changed via the main menu; this however posed to be a problem which I will describe in the implementation chapter.

After the combat style was decided, it was time to think about the design and how I was going to go about creating something like this. Before starting the demo, I wanted to see how other games implemented their weapon switches, so I looked at games that I could think of that used multiple weapons. However, I did not just stick to the genre that my demo was a part of; the weapon switch system was actually inspired shooters such as CS:GO and Fortnite where the player could switch between weapons using the number keys. I found that using the number keys was the easiest way to switch between weapons and also felt the most natural since the keys were so close to the movement keys WASD.

One of my modules in the third year of my degree was Physical Computing. After seeing what could be created using an Arduino, I felt really inspired to create my own controller for my tech demo. Similar to how the Kinect works, I wanted the Arduino to be able to detect the movement of the player in real life using a camera which then uses maths to calculate certain variables like the velocity and displacement of the players movements which then translates that movement into movement in Unity, however I quickly found out that it would have taken too long to implement something like that, forcing me to use the keyboard and mouse as my controller.

Chapter 3

Design and Implementation

3.1 Concept

The idea behind the combat system was to be able change weapons and attack combinations on the fly which allows the player to use the different weapons to their advantage. Each weapon would have special characteristics against different types of enemies for example, the sword and shield class may have an advantage over a ranged archer class, or a twin dagger class may have advantage over a great sword class due the difference in speed. I wanted the player to be able to pick a weapon to dispatch the enemies relevant to their classes.

My demo concept was to only display the games combat system so that I could perfect it before adding any story to it. The main requirements of my tech demo are as follows:

- Four changeable weapons
- Customisable attack modules
- Enemy AI using NPBehave
- PGC environment
- Arduino controller
- Fast paced action – similar to Nier: Automata

Although it was not as important as the requirements listed above, I wanted to create my own designs for the avatars and environment to fit a certain theme; I wanted the demo to look and feel like a game of the JRPG genre however, due to certain limitations I was forced to change the look and feel of the demo which did not turn out to be a negative thing by the end of the project.

I was not sure where to start when it came to the workflow which pushed me to create basic rules/checkpoints that I wanted to follow:

- Perform self-tests after every single asset created in Unity
- Design the characters
- Find animations
- Code movement and attack for the player
- Play test and make changes
- Code enemy movement and attack
- Play test and make changes
- Put in environments and UI
- Play test and make final changes

When it came to play testing, I wanted a group of testers with a variety of backgrounds so that I could get as many diverse opinions as possible as well as other game designers to harshly critique my demo. This was so that I could change the demo as much as I could to accommodate as many groups of people as possible but keeping in mind the original target audience, people who enjoy playing action games.

3.2 Sprints

I initially started off the project by designing a character which I would have then imported into blender to be 3D modelled and then animated, however after meeting with my supervisor I realised that the core mechanic of the game should be more of a priority as creating my own character would take up too much time due to my lack of design experience; because of this I decided to start on the

core mechanics of the game using animations that I could find on www.mixamo.com which is a website where you can download and use the animations for Unity. After finishing the movement for both the player and enemy I found myself stuck on what to do next which brought my project to a halt. After writing down the key components that I wanted to achieve by the end of the project (shown in Figure 1) I chose to keep myself motivated and keep up the pace of work by making personal checkpoints of where I wanted to be in the form of bi-weekly sprints. The tasks that are marked with high priority are what I believed to be the backbone of my demo and are the key features that allow the demo to function; anything with a lower priority is either for the general aesthetic or playability of the demo.

The reason I chose to do bi-weekly as opposed to weekly sprints is the nature of each sprint; I felt that that certain tasks such as: player animations, was too large a task to finish within one week; these can be found below.

Figure 1: Overall project goals

Task	Priority	Done?
Player attacking/movement animations	High	
Player movement	High	
Weapon switch mechanics	High	
Enemy hitbox/health/death	High	
Enemy tracking/spawn/attacking system	High	
UI – to make the demo more game like	Medium	
PGC environment	Medium	
Create a game controller using the Arduino	Low	
Concept art of the character	Low	
Game Controller	Low	
Create behaviour trees for the enemies for difficulty using NPBehave	Low	

After separating the overall tasks in order of priority I started the bi-weekly sprints which are the overall projections that are broken down into more specific tasks, again marked by priority. My very first sprint is shown in figure 2. The main thing that I wanted for my demo was, naturally, the player and its mechanics and since the players movement was done I could focus on the attacking portion of the demo so I made that the focus of my first sprint.

The reason why I put the NPBehave and PGC environment as a low priority, despite being one of my major requirements was because it did not directly affect the demo mechanics. These particular things would only affect the main games playability.

To also clarify, the game controller is not there twice. One states game controller using the Arduino and the other just states game controller. The latter actually refers to a console game controller for example, the Xbox controller or PS4 controller.

Despite originally starting on the art designs first, I changed the concept art to low priority because I realised that it was pointless with all the free art that can be found on the internet, however, I still would have liked to have my own art work involved in the demo.

Below in figure 2, is the first official sprint. The reason I started doing sprints so late was because I had spent a massive amount of time trying to design the art work of my character. It was probably my biggest mistake going into the project.

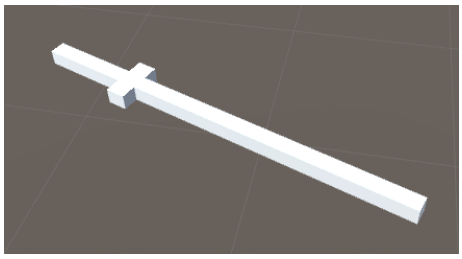
Figure 2: Sprint 1: 1st March – 15th March

Task	Priority	Done?
Find attack animation for shield and sword	High	yes
Combination transitions for the attack animations for shield and sword	High	no
Weapon Switching	High	Yes
Enemy tracking/movement	High	Yes
Enemy hitbox/health/death	medium	No

Despite starting my sprints late, I managed to bounce back and pick up the pace and not long after I got to a position where I was happy with my progress.

The main task of the sprint was the weapon switching mechanic since the demo is based around being able to switch up your combat style mid combat. This was a bit tricky considering I have never done anything like this however, after some pondering, I found that it could easily be implemented with a simple use of arrays; I made it so that it disables all other game objects(weapons) that were not equipped by forcing the object in the hierarchy to disable and enable where appropriate. Since I was not concentrating on the aesthetic of the game at this point, I did not care too much about how the weapons looked so as a placeholder I created extremely basic weapons from the cubes provided in unity, shown in figure 3.

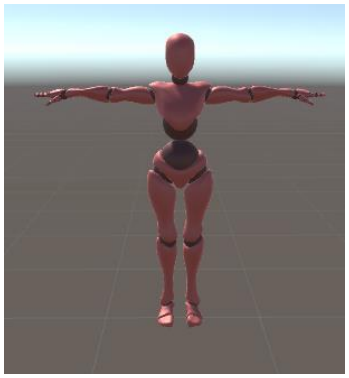
Figure 3: Basic sword made from cubes



After the weapon system created and implemented, it was time to start attacking, this is where it got tricky due to the fact that I couldn't do my own art and animations. As previously mentioned, the animations were found using www.mixamo.com where I also took the basic Xbot avatar, shown in figure 3, since I could not add my own avatar. To start with I wanted to find the animations for the 'sword and shield' class because I thought that I would that it would have the work that needed to be done on it with the blocking mechanic which I ended up not doing. Finding the right animations was

tricky because I wanted specific movements; after many hours of browsing I came to the conclusion that I was not going to get what I wanted so I had to compromise for a slower gameplay.

Figure 4: Player avatar - Xbot



When it came to the enemy movement, I, thankfully, actually already had something that I was working on prior to starting the project. For the movement I used a NavMeshAgent on a navMesh. However, this was only as a test for the movement as I wanted to implement a more complex AI using NPBehave from what I learned in the module: Games AI, but more detail on how I implemented it in the next section.

Anything that I could not implement in any one of the sprints I put as the top priority in the following sprint, as shown by the combinations transitions for the attack animations for sword and shield in both figure 4 and figure 5. The reason I could not finish the combination transitions was because it became very messy and confusing, not to mention it had been over a year since I had touched animations in Unity.

Figure 5: Sprint 2: 16th March – 29th March

Task	Priority	Done?
Combination transitions for the attack animations for sword and shield	High	Yes
Enemy hitbox/health/death	High	Kind of
Enemy Spawning system	High	Yes
Find animations for dagger and or great sword attacking and or hand to hand	High	Yes
Enemy attacking system(animation)	Medium	No

Once I found all of the animations that I needed for the sword and shield class, I needed to string them together to make them combo off of each other, however I ran into some complications. Some of my animations would make the avatar appear to move however, the avatar itself would stay in it's original position from when the animation started playing. I thought I could fix this by changing the rig of the animation to humanoid as opposed to generic but that broke the animations; in the end, I was forced to leave it the way it was. Despite this set back, I successfully strung together all the animations in the sword and shield class which I initially did using just states however I found a much easier way to contain all the states in one place without having to have multiple animator controllers – sub-state machines; this made the animator easier to manage as it was less clustered. Once all the

sword and shield animations were finished, I needed to find the animations for all the weapons which took up a lot of my time since there were very limited animations. In the end, what I thought to be the weapon class that had the smallest number of animations, had the most - the hand to hand combat class which I will go into more detail in the implementation section.

I initially did not want to find all of the attack animations however, once I started I thought it to be more efficient if I found them all at once as opposed to going through the animation list over and over again which I found to be tedious; this caused a minor set back since it made me spend more time on this task. However, once most the animations were found I decided not to string them together right away so that I could move onto the enemy since I now knew a quicker way to do it.

I first started off using a simple cube to represent the enemy. This was so that I could test out the player attack animations and how the attacks would interact with the enemy and its colliders. I added tracking script to the enemy so that it would follow the player which could then be destroyed on impact with the players weapon which I quickly fixed and made it so that the enemy would only be destroyed while an attack animation was playing.

With the enemy tracking and death out of the way, it was time to create a spawning system. I initially created one spawn position which would eventually turn into three that were placed in various locations, but I wanted to do that later because I did not want my scene to be too clustered while I continued to work on the demo. At this point in time, I needed to consider whether or not I wanted the enemy would have health or they would just die in a single attack, I eventually opted for them to die in a single hit which left me with the attacking system of the enemy now.

To recap, by the third sprint I now had the player, its movement, attacking, and the enemy movement. This was where I thought I could now test it. One of the main pieces of feedback that I received was that, with what I had, the testers wanted to feel some sense of accomplishment, especially since the enemy could not yet attack the player. Because of the feedback that I received, I decided to make that a top priority in the next sprint.

Figure 6: Sprint 3 - 1st April – 15th April

Task	Priority	Done?
Combination transitions for the attack animations for all weapons	High	Yes
Enemy Attacking system	High	No
Score system	High	Yes
Enemy animations	Medium	No
Player Health/Death/UI	Medium	Kind of
Find remaining animations for dagger and or sword attacking and or hand to hand	Low	Yes

Once testing was over, it was time to get back to business. Implementing the scoring system was very simple so I decided that It would be better to add it once I finished the player's attack animation transitions. After finishing the attack animations, I moved onto the enemy's AI system however, this

proved to be quite challenging and I spent many hours staring at a blank script and losing motivation; this is when I thought that it may be better to do something easier to keep up moral, so I added the scoring system which was done by adding in a canvas along with some text that printed a variable that increased with the death of an enemy.

With the scoring system finished, I did not have the heart to move back to the enemy attacking so I moved onto starting the player's UI, namely, the death. This was something that I thought would be difficult which proved to be wrong however due to the fact that I used NavMeshAgents I could not make the enemy and the player collide; once the enemy reached the player, they would stop shy of the player and never collide.

In order to test my players health and death, I created a separate entity which I called 'The Cube'. This cube would become the life force or my player, quite literally, so that I could test out the collision with the enemy which turned out to be a success which I thought would not happen right away with the previous collision problem with the player. I diagnosed the problem to be a NavMesh problem, so I left it was it was because once I fully implemented the enemy AI in NPBehave I could redo the enemy-player collision and delete the NavMesh.

I also picked up where I left the animations and found more since I felt that I did not have enough attack for all the weapons since I wanted each weapon to have unique attack combinations to give the player incentive to change from one weapon to another.

In the specifications I stated that different weapons would be more effective against different types of enemies however, I could not do this because it required me to find more than one type of enemy and then code many more behaviour trees which I did not have the time for.

Figure 7: Sprint 4 – 15th April – 1st May

Task	Priority	Done?
Enemy animations	High	Some
Enemy Attacking system	High	Kind of
Enemy AI	High	Yes
Environment	Medium	Yes
Player Health/Death/UI	Medium	Kind of

With the enemy attacking system the only piece that was missing from the core mechanic of the demo, I finally went back to do it. First of all, I found an avatar for the enemy – a troll-like vampire, seen in figure 7 below, which became the start of an idea for the environment theme. With the avatar found, I needed some animations E.G. walking and attacking; I did not find the attack animations at this point in time because I wanted to make sure the avatar could properly interact with the player before added anything that could potentially cause any difficulties. Once I had the walking animation, I tested the collisions with the player and to my surprise, worked the first time without any trouble. Since I did not have any attack animations I used the enemy's body as a weapon, I made it so that the

enemy would ‘attack’ the player simply by walking into them, in its current state, the ‘player’ would be the cube which was stationary.

The reason why I left player death in the sprint was because I was not finished with it, I wanted to add a death animations for the player.

Figure 7: enemy avatar

After allowing the player to be hit and then die, which made the players movement script inactive, I thought I would see if my characters could move freely on a different environment as opposed to a flat surface, so I found a free terrain from the asset store and tested it. I first baked the terrain under the navigation bar which created a walkable environment and pressed play, what I found was very interesting; it worked, however at some points, such as on hills, both the player and enemy would sometimes fly off to nowhere, I did not expect this and I was not sure how to fix it so I opted for a flat ground, which posed similar problems anyway, the player would sometimes slide around a textured platform. however, they are not nearly as bad was what happened in an environment with hills. I used some of the environment from the environment package however I still needed to find more to incorporate into the demo.

The next part was the biggest challenge of the project, incorporating NPBehave into my demo. I needed to work out how to add it into the demo through the NPBehave reference, given examples and spending countless hours staring at and comparing the scripts from the module Games AI. Eventually I managed strip apart the scripts from the tank game and apply the same techniques such as using partial classes to finally implement NPBehave into my project which I tested by giving my enemy the simple instruction to move forward.

After finally adding in the NPBehave, I now needed to completely revamp the enemy tracking system along with its attacking system as well as finding the attacking animations for the enemy. Since finding the enemy avatar, I wanted to find environment textures which suited the enemy’s appearance; despite this, I did not want to change the appearance of the player because I liked the plain look of the Xbot avatar. However, I felt that the environment was to do be done last because I had nothing to do with the mechanics of the game and was purely for general aesthetic and the users enjoyment as it was one of the feedbacks that I received.

I finished coding the enemy AI in the final sprint of the project.

Figure 8: Sprint 5 – the final sprint - 1st May – 15th May

Task	Priority	Done?
Fix the Player UI	High	Yes
Enemy Attacking system/animation	High	Yes
Replace enemy tracking with NPBehave	High	Yes
Find environment textures	Low	Yes

I first started off with the easiest part of the sprint – the player UI, I needed to fix how the players health looked; where I left it, the players health was a very smaller health bar that did not scale with different sized screens. I quickly fixed that by making it so that the whole canvas would stretch out to fit the entire screen should there be a change in window size.

Then to the harder part. When looking for the enemy attack animation I realised that I had already used most of the animations on Mixamo and had run out of options, so I decided that my enemy would only have one attack that would repeat itself in intervals. With the attack animation found it was time to move onto the bulk of things – the AI; once I figured out how to create and call functions, I created two basic, but vital functions, MoveAI and TurnAI which, as its function name suggests, moves and turns the AI should certain conditions be met. I managed to manipulate the enemy so that it would navigate its environment until it was in range of the player, which would then make it follow the player, turning and moving toward the player indefinitely unless the player got out of range again, once it reached a certain distance within the players vicinity, it would proceed to attack, chipping away at the players health slowly but surely until death.

Finding the environment took longer than expected; there were few environment texture packages that were free that suited the environment that I wanted. In the end, I opted to create most of my environment using some game objects found in several packages, although I did use one of the examples as three of my borders, so that none of the characters, both player and enemy could not fall off the edge; the final border being a castle wall that I created using some prefabs. I also designed and implemented a small village to add to both the aesthetic and to help the AI navigate around the terrain.

Figure 9: The terrain

3.3 Implementation

Something that surprised me, despite the lack of animations, was the amount of animations available to users for hand to hand combat. Despite believing that this class would be the least versatile, I found many different types of attacks for it, ranging from punches to kicks and even fireballs; admittingly I had a little too much fun with it. After finding the fireball animation, I realised that my attacks may have been too boring and linear; that was when I realised my demo did not have to be dull, that I could add a lot more attacks that I did not even think of; it gave me motivation to try new attacks, from fly kicks to charging attacks and actually using a fireball.

When giving the attack animations its play conditions, I had to choose which variables to use, triggers or Booleans. The players movement was an easy decision, I chose Booleans because I wanted the walk, run, strafe etc animations to play indefinitely while a certain action was occurring such as the w key being held down. The attacking, however did not work the same; I initially tried using Booleans for the animations but found that it was more work than was required since I needed to add extra code to set the Booleans to true and false. That's when I found that triggers worked similar to Booleans except they automatically turn off once it is turned on, this meant that I needed to write less code the same action.

Before I could test my demo, I needed to make sure I reached the check mark from my specification which was the movement and attack for the player. Although this was finished I had nothing to test the players attacks on; this urged me to finally add my basic enemy AI which I already had as previously mentioned. The basic AI consisted of a cube that had a script that made it follow an object that had the navMeshAgent component attached to, that resided on a walkable navMesh. Once the enemy AI was implemented it was time to create its spawning position; this is where I ran into a problem: when I destroyed the enemy that was contain within the hierarchy, it would throw a nullPointerException error due to the fact that I was trying to reference an object that did not exist. I fixed this by creating a prefab instance of the enemy however, this created yet another problem: the enemies would no longer follow the person and I did not know why. That was when I consulted my supervisor for help. Too fix this problem I, instead of making the target of the enemy a public transform which was the player game object, I created a private variable that referenced the players position and used that as a target which was in turn, updated every frame. Now it was time to test the demo in its current state.

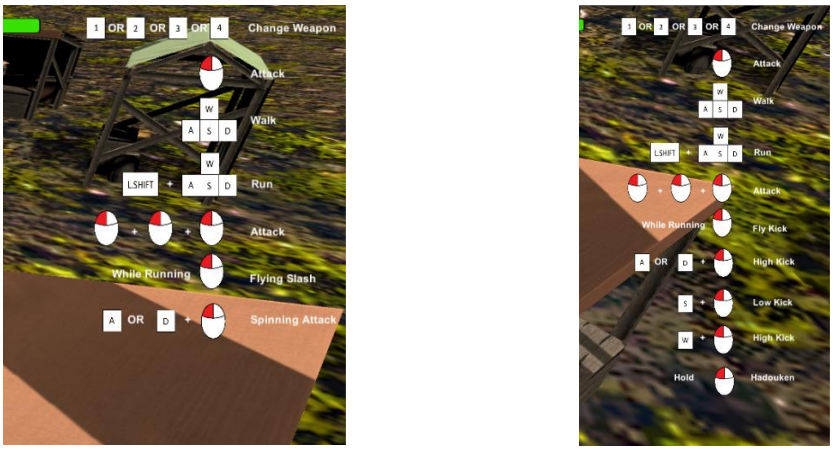
The main feedback that I got from the test was the lack of accomplishment. Users felt that there should be some sort of reward system for killing a hoard of enemies, especially since there was no endpoint in the demo just yet. This was something that I felt was fairly easily to create so I made it one of the top priorities in the following sprint – creating my first basic UI: the score system which just consisted of text on a canvas that would print a score variable that increases in value once an enemy was killed.

As stated in the previous section, I gave the player its health UI and death, however the enemy would never actually collide with the player which I assumed was because the enemy would stop once it reached its target, but this was not very important since I intended to change the AI system from using a navMesh to NPBehave. To give the demo a more game like feel, I gave the demo a hit indicator, making the scene flash red once the collision between the enemy and the players' life force (the cube) occurred. At this point, I play tested the demo again. Naturally, the user responses were related to the player dying when the enemy touched the cube instead of the player itself, with some suggesting that I add environments in.

Incorporating NPBehave into my demo was by far the trickiest part of the project. This was my previous experience only extended to editing a script with NPBehave already attached to it as opposed to incorporating it from scratch. After many hours of looking at the examples and the existing NPBehave incorporated script, I managed to include NPBehave into my system and thankfully, that was the hard part. When it came to code the AI, I had relevant experience in the field therefore it was a matter of applying what I had already known. After testing out basic movements, I was ready to go all in; I wanted to make the enemies swarm the player should they make the mistake of getting too close to the wandering enemies. To do this I needed to create several blackboard conditions, to name a few: targetDistance, targetInFront; these were to check the distance between the enemy and the player and whether or not the player was in front of the enemy respectively. I made the AI wander around the scene if the player was 'too far' from them which, if that was the case, the enemy would move forward until it reached a part of the environment then turn away from it and move forward again so that it did not collide with buildings and walls. Once I had the wandering part of the AI done it was time to make the enemy attack; I created a new function that played the attack animation for the enemy which would damage the player should the enemy's weapon collide with the player. Once the player was in range of the enemy (25 units), the new tracking system was set in motion; I made the tracking system check to see if the player was in front of the enemy by using the blackboard condition 'targetInFront' which uses the unit vector of the vector difference between the player and the enemy; if the player was *not* in front of the enemy, it will proceed to turn until it is where it will then check to see if the player is directly centred in front and if it was not, check to see if the player was on the left or right and then turn in the appropriate direction to face the player and move forward. Once the enemy was a certain distance away from the enemy (1 units), I made the enemy stop moving and attack, however, this made the enemy move to the player until it realises that the enemy is no longer centred where it proceeds to stop and turn again and restart its trajectory. I fixed this by adding another distance condition; if the enemy was not a certain distance from the player (1 unit), the enemy will again check to see if the player is centred and again turn appropriately. This made the AI update its direction while it moved forward so that it continually moves toward the player. With the AI finally implemented, I could finally delete the entity now known as 'the cube' and the navMesh.

After conducting another user test, I realised that I needed to create some kind of tutorial on how to play the demo, so I created a control UI which displayed all the buttons that were used to control the player as well as all the attacks E.G. how to perform certain actions such as the charged attacks or the fly kick. However, there was a flaw with this, depending on the size of the users' screen, the UI could cover most of the display and take up unnecessary space. The solution I came up with was to change the UI itself; making the main actions that were universal across all weapon classes stay on the screen but making the attacks come and go where appropriate. To elaborate, if the sword and shield class was equipped, the UI will only display the commands that were available to the sword and shield class. Similarly, if the hand to hand combat class was equipped, only the commands available to that class will be displayed, such as the Hadouken attack. These are shown in figure 11 below. This reason I did this rather than having a separate menu to show the controls was so that the player would not have to constantly switch between scenes or screens just to look at the controls. This would also force the player to multitask, testing the players ability learn the game as well as play it at the same time.

Figure 10: changes in UI in terms of the players controls, on the left displays the sword and shield, on the right displays hand to hand.



The images that can be seen in figure 10 are images that I created using paint which I imported into Unity.

One of the feedback that I was given in the second play test was to 'make it more game-like'. However, there were no specific instructions on how to do so, which left it down to my own interpretation. One of the things that I realised was that the player could still move around the surface while the attack animation was being played. I found that this to be very unappealing aesthetically, so I needed to fix it by setting the players movement parameter zero while the animation was playing, on that note, I also made it so that the enemy would only die while a player attack animation was playing. However, I did not make the value of movement zero for all attack animations, as previously mentioned, I ran into some trouble with certain animations that would make the players avatar appear to move but did not move the player themselves. I did not have a proper fix for this, so I allowed the player to move around while these animations played to mitigate the affects of the moving avatar.

As previously mentioned in the above chapter, I chose to kill off enemies in a single hit so that it does not overwhelm players. This made the demo slightly easier so, to raise the difficulty back up I increased the number of spawning positions thereby increasing the number of enemies that spawned. In the event of the player dying, I added a game over screen which allows the player to start again from the beginning however, after receiving feedback which stated that I should add a start menu scene I decided to add one to give a quick overview of what the demo showcased and the purpose of it.

One of the things that I did not do was make the enemy damage the player only when the attack animation played (that was the only way the player could damage the enemies), this was because the player had health and would get damaged twice anyway, once on the downward strike and once on the enemy pulled up its sword after striking. I decreased the effects of this by lowering the damage output of the enemy's attack.

3.4 Changes

When it came to design the combat system, I took inspirations from, as previously stated, Kingdom Hearts and Nier: Automata's gameplay however, for the art style, I wanted to create my own designs

and animations but once I finished my first initial character design, I realised that 3D modelling was way out of my scope and would have taken too much time, so I decided to find assets online.

In the initial vision of the game there were many features that I wanted to add but did not make the cut due to either time restrictions or just the lack of experience/skill to create it. In figure 9 below, you can see all the design ideas that did not make it into the tech demo and how it was changed accordingly.

Figure 11: A table showing the initial demo ideas and how it was changed

Initial Design	How it was implemented
Original character designs and animations	Character and animations used from www.mixamo.com
Player can change both the weapons the attack combinations via a main menu	Player can change the type of weapons which alters the attacks of each player
Custom made Arduino controller for the gameplay	Use of keyboard and mouse
Aerial combos	No Aerial combos
PCG environment	Closed space with obstacles
Blocking with a shield	nothing
Music	No sounds

One of the first things that I had to change was the character that I was designing; this is due to the fact that I had no experience in 3D modelling and it would have taken too much time to learn and as stated in the previous chapter, I resorted to finding free animations on www.mixamo.com that I could use since time was an issue. After speaking to my supervisor, Jeremy, I decided to focus on the game mechanics, which has also been altered, first before starting any type of original character design or animation.

As for the combat, I had to make several changes; I wanted the player to have full customisation over what combination of attack they wanted to use and even its execution order by opening the main menu and change around the attack modules. This was something that I could not do because of the limitations set on me by the lack of available animations; there were not enough attack animations for me to freely pick and choose; because of this, I chose not to include the attack modules customisation, but instead gave the player a larger variety of pre-set attacks which would change according to the weapon that was equipped at the time to make it feel like the player had more customisation than they did. On the topic of attacks, one of the things that I was not able to include was the use of aerial attacks which was literally due to the fact that there were no animations available.

One of the biggest changes that I was forced to make was removing the PCG environment. I initially wanted to create an open world demo that would never end, allowing the player to explore and endless world while being chased by enemies. This was quick to shut down because of the lack of experience and knowledge about procedurally generated content.

Time was a big restriction when it came to the project; this was one of the reasons that I was forced to keep the blocking mechanic out of the demo. One of the weapon classes, the sword and shield, was supposed to include a mechanic where the player can block an incoming enemy attack with their shield, however I felt that, with the deadline closing in, it was not a major part of the demo and would make the demo harder to win.

Similar to the environment, I wanted to add music for the overall enjoyment of the demo, so I contacted a sound engineer for help with music and sounds, however, due to some unforeseen circumstances she had to stop creating the music for my project. This happened quite some time into my project, so I did not have much time to find music and sound effects to add into the game.

Probably the most impactful change I had to make was the flow of the combat style. Due to limited available animations and my lack of experience, the pace of the combat in the demo was drastically decreased. It went from a fast-paced action RPG to a more realistic paced combat. Although this did not affect the main mechanic of the demo, it created a void in my heart, since it was not like the system I had envisioned. However, this did not stop me from creating what I wanted to create, which was the combat system with high customisability. This may have in fact been a blessing because if not for the lack of animations, I would not have thought of other things like charged attacks or running attacks.

Perhaps the most disappointing change was the fact that I could not add my own Arduino controller; this alone would have been a whole project in itself that I would like to one day create. The idea behind the controller was to make the demo more interactable for the user, giving it a very unique feature. I wanted the Arduino controller to work like the Kinect, detecting the users body movements and interpreting them in order to control the character within the game. This would have simply taken too long because it would be a completely new experience to me which would require me to do extensive research and testing which I did not have time to do. This is something that I would definitely like to create in the future for the continuation of this project.

Chapter 4

Testing and Evaluation

4.1 Public Testing

As stated in the chapter: Specification, I wanted to perform unit tests which means: after any new component of the demo was added/created, I would play test that component extensively to make sure that it works. I chose to perform these tests independently because I felt that having play testers after every task would take too long and I would lose very valuable time and as a gamer myself I thought that I could trust my own opinion. Unit tests became tedious extremely fast however, after some time, they became routine; my first unit test was on the players movement (both the actual movement and animation). Since this was the first unit test, it was a gruelling task, especially with it being the longest test, I had to test the animation repeatedly to make sure everything was perfect and transitioning well. The main reason the difficulty of this particular unit test was so high was because there were so many animations to test, from the idle standing position to walking, strafing, and walking backward to the running versions of each of them. Making sure each of them transitioned between each other took a really long time which is one of the reasons why I chose to do my springs biweekly as opposed to the more common weekly sprints.

Aside from unit testing I had users, both gamers and non-gamers, test my demo. The purpose of this was to get feedback from users so that I had a more diverse pool of opinions to ensure that I my project would accommodate to a broader audience. I initially wanted to perform play tests at specific points of production:

- Once the player movement and attacks were finished
- When the enemy movement and attack was implemented
- When the full enemy AI was added
- After the game was completed

But naturally, it was sometimes necessary to perform extra play tests in the event of me doubting the quality of my work. The play testing was done in small groups; this made it easier to collect feedback from each individual person since I did not have to ask them closed questions which reduced the chances of me swaying their opinions about the demo in a positive light.

For my first play test I decided to use a control group of gamers; this was because I wanted a group of people that had plenty of experience with basic movement and attack in games. I wanted the testers to critique my demo quite harshly to hash out the flaws from the beginning. The purpose of the first play test was to help with my own unit testing (on the players movement) and also check how viable the attack animation transitions were. Despite my intentions for the play test, most users did not have many negative comments about the movement, but they felt that a reward system should be included, which is what the score system was born from. When I asked my testers about the actual game mechanics, some comments included the fact that certain attacks that made the avatar appear to move but applied no physics to the actual object itself; this however, was something that I could not properly fix so I had to mitigate the affects my allowing the player to move during certain animations. When it came to implement the feedback into my project, I made a list of all the changes that I felt were relevant and then ordered them in level of importance. For this particular sprint, the main feedback that I received was some kind of reward system. I did not feel the need to include the feedback about the movement due to the fact that I was still working on it at the time of the play test.

While I continued to add more attack animations, I had to perform unity tests on them all to see if there were any potential bugs, to see how well they transitioned into each other, especially while trying to change weapon mid combat. Once I had felt that I did enough unit testing, I could move onto the next task, the enemy attacking mechanics which was explained in the implementation section of this report. When it was finished I held another public play test however, this time it was open to everything, from non-gamers to casual gamers to gamers with countless hours under their belt. At this point in time, I now had the player moving, attacking and dying, along with the enemy movement and the player hitbox (the cube); which, I felt meant that I had enough components were enough for a functioning game demo. Of course, certain things, like the cube, were only place holders until the actual things that they represented were fully implemented.

In this portion of game testing, testers felt that the demo was too plain and that I should add an environment to make the demo look more like a game. After a brief explanation, testers understood that I could not make the enemy attack the player directly so they were very lenient on critiquing that portion of the demo but that didn't stop them from making any comments on it. Testers suggested that I needed to make the enemy attack the player directly, add in an environment and one even suggested to 'make it more game-like' however this particular person did not go into detail on how to do so (this was a written review) so it was left to my own interpretation. Reading that comment forced me think about my demo and made me realise that although I wanted to focus on the combat system, I had forgotten about the enjoyability of it; at the end of the day, it was to go into a future game. This urged me to fix a few minor things like making the player stop moving and got me thinking about future developments (within the project that was yet implemented) such as not allowing the player to move through solid objects like a building. With the player attacking system nearly fully implemented I got more feedback on that; most of the tester issues related to the Hadouken attack spawning two balls at the same time and not having a straight trajectory was easily fixed simply by changing the direction of the firing points and the direction on the initial force added to the ball. I also added a pre-set environment that I found in one of the texture packages that I found on the asset store which I quickly had to change since it started throwing my player character around wildly.

The next playtest was perhaps the most important one; this is where the enemy AI was completely implemented along with attacking improvements. Since I had only just finished the AI, the testers had their fair share of opinions with one describing it as 'janky'. There were many complaints about the AI 'stuttering', where the enemy would move forward and then backward repeatedly, though it did have a net movement of going forward, it still looked buggy. This however, could not be said for the tracking system for the enemy. Once the enemy was in range of the player, it executed its actions near perfectly. Another thing that the testers mentioned was again, the environment. Although I had some basic textures, the feedback I got suggested I add a more detailed environment which I managed to do after spending hours on finding free asset packages in the asset store. Fixing the AI was simply making the AI move only in one direction. I initially wanted the AI to be able to move backwards but that did not work and by this time, I was running out of time to figure out how to do it.

When I finally had an environment and AI that I was satisfied with I decided to host one last play test, this time I specifically included fellow game designers to critique my demo. To my surprise, I did not get too many comments about the AI (though I did have some); this set of testers mostly made comments about the UI and how there was no game over screen and that the player health bar was too small. Several people however pointed out that if I was not in the room, they would not know how to play the demo since there were no instructions or tutorials. I realised that this was very important, so I prioritised that over the other feedback. Editing the health bar was the easiest so I left that until the end once I made note of it. The command input instructions were tricky at first; I was

not sure how I was going to fit everything in one place, especially when there was a high chance that anyone that were to play this demo, they would have a display with a different resolution to my own. I fixed this by separating all the commands and grouping them into their weapon classes. By copying and pasting the code for my weapon switcher, I was able to do the same with the UI; when a specific weapon was equipped, the UI would change accordingly. Once that was finished, I added a main menu scene that gave a basic introduction to the demo with a play button that takes the user into the demo. I also decided that I should add a game over screen with the option to go back to the main menu to make the demo more 'game-like',

4.2 Evaluation

Once I finished my project, I left it for a full day so that I could play test it myself, so that I could compare it to the initial projection of my demo; I found that, although the pace of the game was not what I initially wanted, I created a demo that still fulfilled the requirements of the specification that I initially drafted. Of course, excluding the things that I was forced to miss out like the PGC environment and the ability to change attack modules however these were compensated with the addition of more attack combinations and a detailed closed environment.

After playing the demo through, I found some flaws; even on a flat surface, the player would slide around and sometimes get caught in between two rigidbodys. After playing around with it, I could not find a solution, so I was forced to just leave it, however it still did not affect the combat system in any way. I also found something very misleading; I placed in stone steps to traverse the wall, except these stairs were not traversable. I had completely forgotten to change it so that there were walkable however, once again, it did not affect the demo in a very negative way, so I chose not to look into it. I also found another error that was not there before 'SendMessage("die") has no receiver'; even after searches online I could not seem to find the reason for this error nor could I find a solution, however, this error did not affect the demo at all, unless 'pause on error' is turned on in the Unity console. To 'fix' this solution, I added a little message in the main menu to turn off 'pause on error' in the console to avoid any problems when running the demo.

Playing through my demo allowed me to remember something that I intended to do toward the end of my project – adding in weapon textures. It was not hard to hard the weapon textures for my demo since there were not many free assets that I could use; thankfully the few weapon assets fit the feel of my demo. All texture packages and resources are found in the bibliography below. I also added the weapon textures for the enemy which had to be scaled about 20 times due to them being so small.

I believe that the demo does capture the essence of the game that I envisioned with the excepting of the customisable attack modules which I compensated for by giving each class an extensive number of attack combinations. One of the biggest problems was the AI; although the AI works, I feel like it could have been implemented in a better way, especially the 'wandering' aspect of it. The movement of the AI is not as smooth as it could have been, it also sometimes walks around in a circle which I could not fix but usually works fine.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

Although the demo did not completely fulfil the requirements set out by the specification, I truly believe that I created a demo that I can be proud of; I consider it a success since I was able to create the combat system which was the main aim of the demo. I feel that although I left out some key aspects of the demo, I believe that the core mechanic was achieved, the fact that the player is able to change fighting styles mid combat, changing the weapon classes thus in effect changing the attack combinations which was the main goal initially. If I compare my demo to the games that inspired it, I would say that the demo is definitely a slower version of what I wanted it to be; I would even compare it to the original Kingdom Hearts battle pace, minus the fact that my demo does not contain any magic or the ability to heal myself which is definitely something that I could add and will add in the future.

5.2 Future Work

There were many features of the demo that I wanted to add but I couldn't, usually because of time but sometimes because of the difficulty level. Some of the extra features that I decided to leave out was because they did not majorly affect the core mechanic of the demo. One of these was the completed version of the slam attack for the Great Sword weapon class; the attack was meant to generate a wave surging out from the player causing damage to any enemies with the area of effect radius. This is definitely something that I intend on adding to the game version of the tech demo. Another form of attack that I wanted to add but could not was the aerial attacks due to the lack of animations. Once I am able to create my own animations I plan on adding aerial combat along with enemies than can fly for a whole new combat experience. When I am able to create my own animations, the pace of the game should naturally speed up adding to the overall experience, that's not to say that the current state of the demo is not enjoyable.

Another thing that I will definitely add into the future game is enemy types for example, giving the enemies their own attack combinations or even weapons and using the behaviour trees to create a more complex AI to make certain enemies harder to kill by making them react to the players attack and dodging, blocking or even countering it. On that note, I will also make modifications to the player's current mechanic by adding blocking with the weapon/shield.

Something, unspecific to this project, that I will definitely create for the project in the future is the Arduino controller. Even if not for this project, it is something that I would like to create for any of my future projects. However, for this particular game I would like to create a controller than has both the depth camera to detect the users real like movement as well as having a joy stick as the input for the players movement. The way I envisioned the Arduino controller to work was the user would do specific movements to perform specific actions for example, moving the users hand from left to right would change the weapon class.

As I progressed through the project there were many ideas that popped into my head however, I could not implement everything that I wanted since there was a lot to add. An example of what came to mind was the lore of the world, why the world was overrun by these creatures and what happened. Or even adding a range class for both the enemy and the player and outposts that would shoot at the player.

Chapter 6

6.1 Bibliography

Assets from assets store:

- Rocky Hills Environment - Light Pack
- Yughues Free Ground Materials
- MYFG-Weapon Pack Lite

Animations and avatars were found on www.mixamo.com

To help getting started with the demo I used this video tutorial playlist as a reference. The tutorial does not use code since it uses Playmaker. I used the first five parts of the playlist as help.

The videos were posted 28th October 2015, 3rd November 2015, 4th November 2015, 12th November 2015, 12th November 2015 for parts 1,2,3,4 and 5 respectively.

https://www.youtube.com/playlist?list=PL8xcKsKsXEfvWsZIZuhV36I3f7DxTIStT