OnSight53

# Contents

# Abstract

OnSight53 is a tech demo that showcases a combat system where the player is able to freely navigate in a procedurally generated open world in which enemy NPCs follow the player and attacks them until eventually killing them. The objective of the tech demo is to survive the game and pick new weapons to kill enemies which will in turn give you points similar to retro arcade games.

# Introduction

I chose to create a combat tech demo as opposed to a full game is because I believe that in order for a game to be a success, it requires adequate mechanics and gameplay, especially in the case of a game where fighting is one of the key components of the game. 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 ‘Design and Implementation’. You can find what the inspirations for the demo in the following chapter and how I implemented it and how I plan to use it as the core for a future game in the chapters ‘Design and Implementation’ and ‘Conclusion and Future Works’ respectively.

# Background/Literature Survey/How I chose my project

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 Bare Hands, 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.

## Design and Implementation

### Specification

I made a list of things that I wanted to include in the game which you could say is the design. The list is as followed:

* Four changeable weapons
* Customisable attack modules
* Aerial attacks
* PCG environment
* Arduino controller
* Fast paced action
* Own designs for the avatars and environment

I was not sure where to start when it came to design the demo 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 changes

### 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 (Jeremy Gow) 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](http://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.

|  |  |  |
| --- | --- | --- |
| 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 Controls | Low |  |
| Create behaviour trees for the enemies for difficulty using NPBehave | Low |  |

Figure 1: Overall project goals

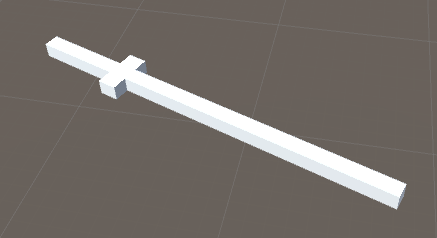
After separating the overall tasks in order of priority I started the bi-weekly sprints which are the overall projections that are broken down into more specific tasks, again marked by priority. My very first sprint is shown in figure 2. The main thing that I wanted for my demo was, naturally, the player 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.

|  |  |  |
| --- | --- | --- |
| 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 |

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

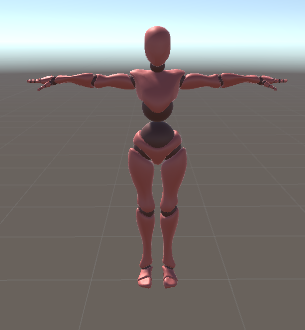
The main task of the sprint was the weapon switching mechanic since the demo is based around being able to switch up your combat style mid combat. This was a bit tricky considering I have never done anything like this however, after some searches and asking around, I found that it could easily be implemented with a simple use of arrays; I made it so that it turns off 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 Mixamo; I also took the basic Xbot, figure 4, avatar from Mixamo 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 compromise for a slower and duller 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 in figure 4 - Sprint 2.

|  |  |  |
| --- | --- | --- |
| 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 |

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

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 this 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 I will go into more detail in the implementation section; this 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.

|  |  |  |
| --- | --- | --- |
| 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 |

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

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 attack animation transitions. After finishing the attack animations, I moved onto the enemy’s attacking 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.

With the scoring system was 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 thought that once I fully implemented the enemy AI I could redo the enemy-player collision, so I left it was it was.

I also picked up where I left the animations and found more since I felt that I did not have enough attack combinations.

|  |  |  |
| --- | --- | --- |
| 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 |

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

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.

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, I expected this however I was not sure how to fix it so I opted for a flat ground, which posed some problems anyway 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.

|  |  |  |
| --- | --- | --- |
| 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 |

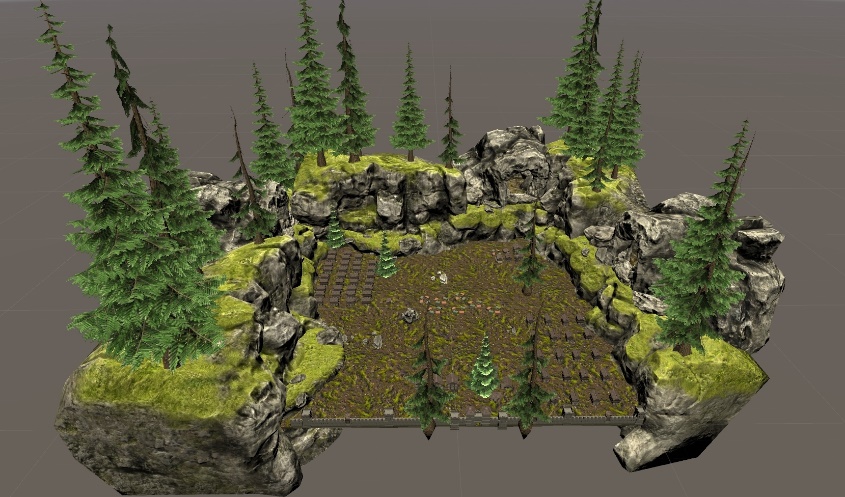
Figure 8: Sprint 5 – the final sprint - 1st May – 15th May

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



### 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.

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 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 testing I made score and UI. Where did I get the images, I made them

Chose not to give enemies health because it would mean the player would die more easily. Wanted to make the player feel overwhelmed due to lots of enemies, giving them health and less enemies is too easy and both is too hard.

Made it so that player cant move while attacking, only when I needed to so that they didn’t just slide back to the original position.

Adding the red flashing to indicate player being hit.

Even flat ground had problems with sliding

Talk about why I used triggers and Booleans as parameters for movement

Nice environment

Player weapon prefab

### Changes

When it came to design the combat system, I took inspirations from, as previously stated, Kingdom Hearts, Nier: Automata 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 10: 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](http://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 |
| Tutorial | Controls on screen |
| Music | No music because my musician sucks |

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](http://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.

# Testing and Evaluation

Once I have player attacking and enemy movement – feedback, add scoring.

I asked for feedback about the mechanics.

Test 2 – enemy movement and walking into the player to kill it.

Feedback: make the enemy attack the player. And add environments

Test 3 – enemy attacking and movement

Feedback: give the player better weapon art. tutorial

Add a better environment

Test 4 -

Player weapon art

Make it more game like – how do I interpret that

heath bar bigger, add control tutorial

add different screens, game over and start.

# Conclusion and Future Work

AOE attack for great sword

Faster paced game

Adding enemy health with death animation

Adding more enemy attacks.

# Bibliography

# Appendices

Assets from assets store:

Rocky Hills Environment - Light Pack

Rock & boulders

Yughues Free Ground Materials

MYFG-Weapon Pack Lite

Mixamo

ADD death animation????