MELEE PROTOTYPE

DES311 | Game Design Practice

Personal Development Portfolio

Liam Rickman
1902527@uad.ac.uk
Game Design and Production

CONTENTS

This portfolio documents my design, development, iteration and analysis of my DES311 project.

Throughout this portfolio I've included links to development videos that will appear like this. Please watch these as they help further showcase my development process.

IMPORTANT LINKS

- Final prototype: https://drive.google.com/file/d/1xOc5ofgTvxIOpMESS72J-NaGPWULWslk/view?usp=sharing
- Gameplay video: https://www.youtube.com/watch?v=RL1EuZFDUPY
- Source Files (GitHub): https://github.com/LiamRickman/DES311_Demo
- Asset List: https://docs.google.com/spreadsheets/d/19myeyRu95yhLf8xdklMwvbopH8eIGCE_mxLpOH9i13s/edit?usp=sharing
- Trello Board: https://trello.com/invite/b/dOK6KKSC/368907e0d153268844f3c1fca6af3d54/des311

CONTENTS

AIMS & OBJECTIVES

AIMS

My aim for this project is to create a third-person character controller within Unreal Engine 4. The controller should feature melee combat which will require basic enemy AI to fight against. It should also feature at least two advanced movement mechanics such as climbing and vaulting.

I also want to improve my productivity skills and time management by using appropriate techniques.

UE4 (Epic Games. 2012)

OBJECTIVES

- Analyse game design techniques that can help me prototype a player controller and analyse game feel and polish techniques to iterate and improve the prototype.
- Design the player controller and any supporting mechanics required.
- Prototype the player controller and supporting mechanics until I reach a minimum viable product.
- Conduct user testing to gather feedback on the prototype's mechanics.
- Iterate on the prototype to improve the game feel and polish.
- Repeat the last two steps as many times as possible.
- Reflect on the project to see if I achieved my goals and improved my skills.

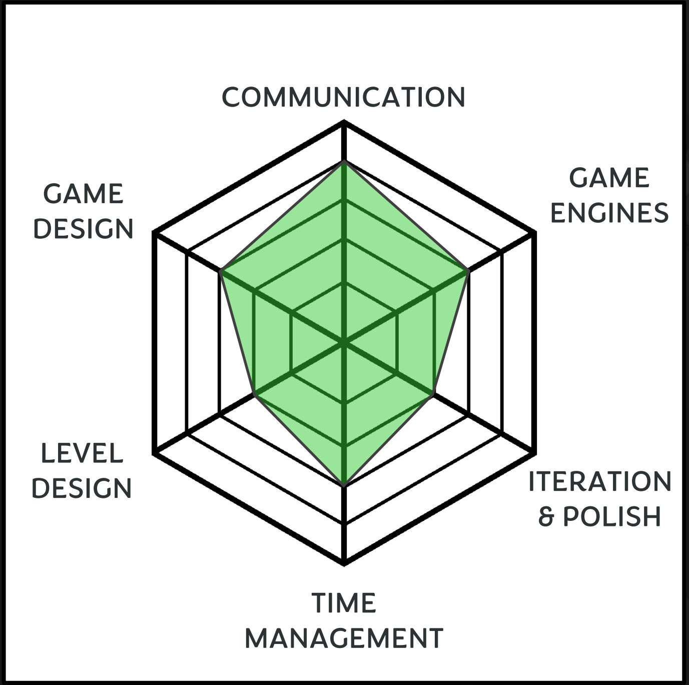
SKILLS DIAMOND

SKILL GAPS

- Game Design
- Iteration and Polish
- Level Design
- Time Management
- Game Engines

THE PROJECT

Once I had outlined my skill gaps, I brainstormed projects that could address as many of these as possible. I have a particular interest in prototyping and level design, however I didn't want too large of a scope that the quality of work would suffer. I decided to leave level design as I can revisit this area later, even using the character controller as part of the level design.



My project will cover Game Design with the initial design and prototyping of the character controller. It will then cover iteration and polish by iterating on the prototype to improve game feel. It will also cover game engines, specifically Unreal Engine 4, which I have not used much before and wanted to improve my skills in it.

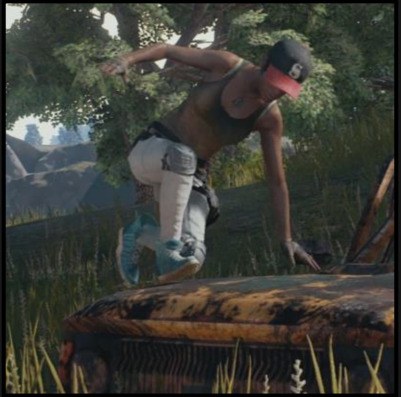
Finally I will also improve my time management skills by using productivity tools such as Trello and Notion.

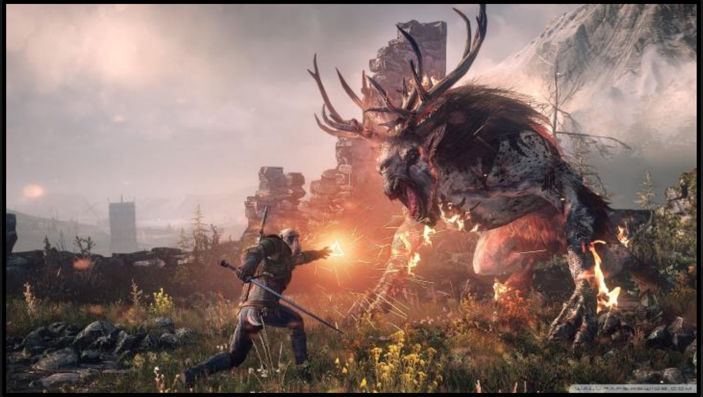
INSPIRATIONS

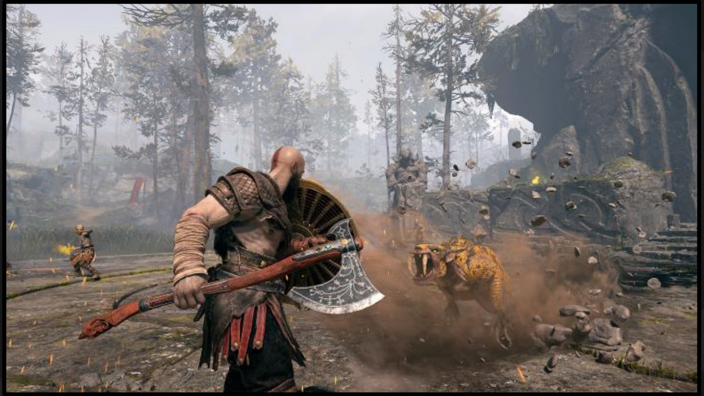
I was inspired by games such as God of War and Witcher 3 for their third-person combat controllers, especially when used in an open world environment.

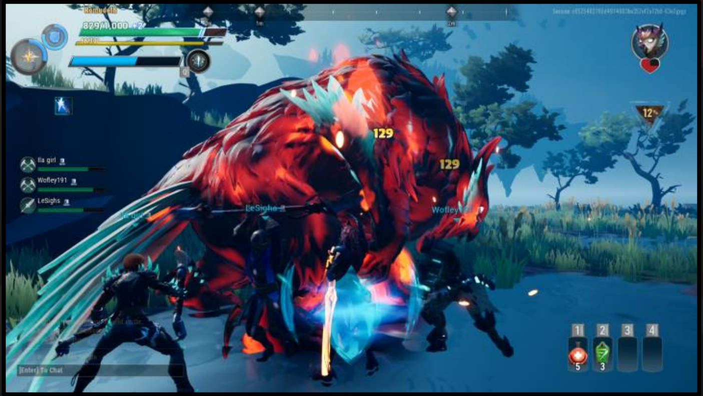
Dauntless' melee combos are particularly appealing as it makes each move feel impactful and skill based rather than just button mashing.

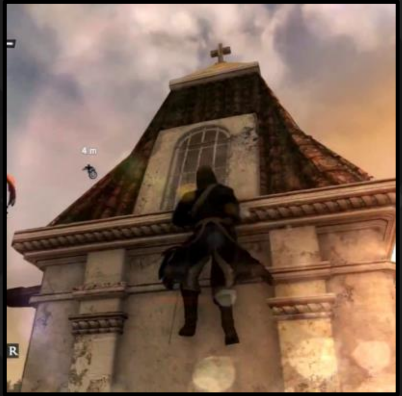
Finally, incorporating more advanced movement such as vaulting and climbing was inspired by PUBG and Assassin's Creed to give the player more options when traversing an environment.


PUBG (PUBG Studios. 2017)


Witcher 3: Wild Hunt (CD Projekt Red. 2015)


God of War (Santa Monica Studio. 2018)


Dauntless (Phoenix Labs. 2019)


Assassin's Creed IV: Black Flag (Ubisoft Montreal. 2013)

GAME ANALYSIS – GOD OF WAR

COMBAT

God of War was my primary inspiration for this game as I hadn't made any third person melee games before. The combat in this game is fluid, heavy and impactful. It is clear that the main character Kratos is very powerful and handles most enemies with ease.

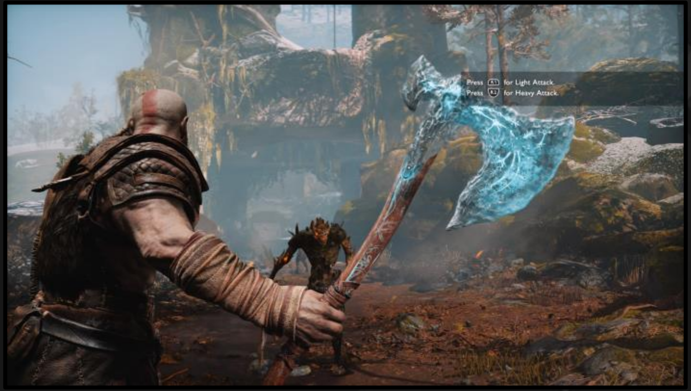
He uses a variety of weapons in combat, primarily an axe, however he uses fists, chain blades, and other weapons. This helps keep the player engaged and avoid combat becoming repetitive.

The combat is very snappy, with enemy lock and targeting being a key feature. This is done automatically as explained in a GDC talk by the developers (Sheth, M. 2019, 7:43)

AXE THROW & RECALL

God of War has an axe throw mechanic that helps break up combat and add more combos rather than spamming the same inputs. This uses lots of animation cancelling (McDonald, J. 2021. 21:27), for both throwing and recalling the axe.

This mechanic gives the player more options in combat that should keep the player more engaged and interested with the combat as a whole.



God of War (Santa Monica Studio. 2018)

GAME ANALYSIS – THE WITCHER 3

COMBAT MECHANICS

The Witcher 3: Wild Hunt uses primarily sword based combat but also includes some mage style abilities such as fire attacks to help break up combat and give the player more choices.

This is similar to God of War and its use of extra abilities to add more diversity to the players attacks, helping prevent combat getting stale.

The Witcher 3 has an execute or finisher mechanic that helps attacks feel more impactful. When killing an enemy the game slows down and plays the attack from a different angle to show the impact of that attack.

PARRYING AND DODGING

To avoid taking damage from enemies the player can parry attacks, which will block the attack and stun the enemy for a short period of time. This allows the player to counter attack and deal damage themselves. This continues giving player more choice in combat instead of button mashing.

There is also a dodge mechanic where the player can dodge to avoid damage. This makes the player think more about the enemies attacks to dodge them properly.

Witcher 3: Wild Hunt (CD Projekt Red. 2015)

GAME ANALYSIS – DAUNTLESS

WEAPON OPTIONS

Dauntless gives the player many weapon options to use in combat: Aether Strikers, Axe, Chain Blades, Hammer, Ostian Repeaters, Sword, and War Pike.

I have focused my analysis on Sword gameplay as this is the planned weapon for my prototype.

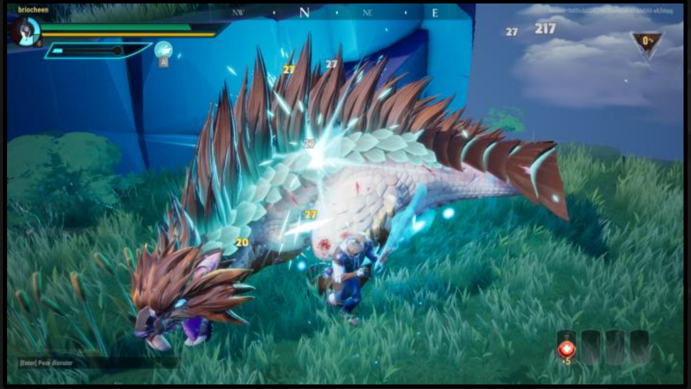
ATTACK COMBOS

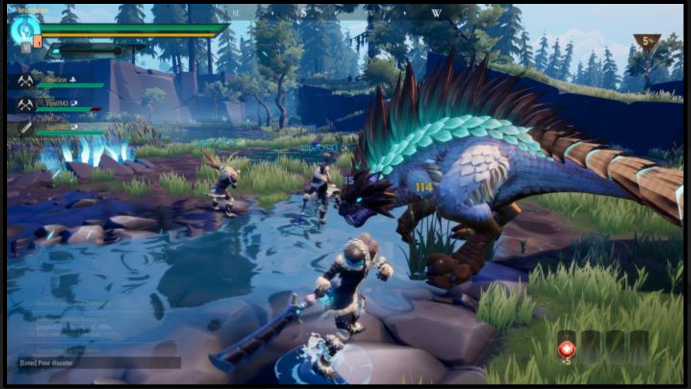
The attacks used in Dauntless feel heavy, every swing feels purposeful. When landing attacks they feel meaty like they are doing high damage.

The attacks do take a while to perform, with long windup times. This does make them feel more impactful but could be awkward in faster paced combat.

The enemies in Dauntless are large creatures that have attack patterns that the player can predict. They usually have wind ups to their attacks that the player can react to and get their attacks in when its safer.

The attack combos are particularly interesting as they add more skill to the gameplay instead of mashing buttons.





Dauntless (Phoenix Labs. 2019)

GAME ANALYSIS – PUBG

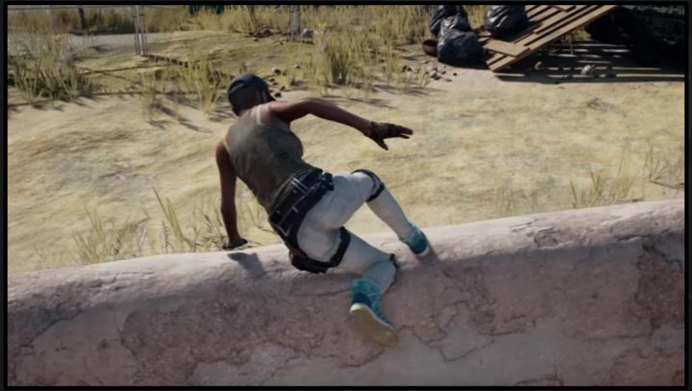
VAULTING AND CLIMBING

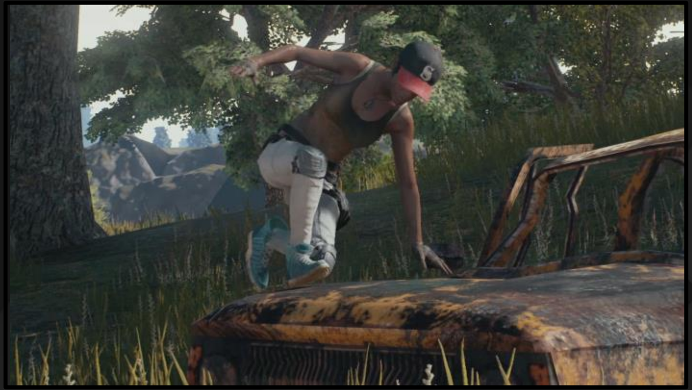
I've analysed PUBG for its vaulting and climbing mechanic as I wanted to implement some advanced movement mechanics to my prototype. Before vaulting was added to PUBG, there was methods of "vaulting" using an exploit which provided an unfair advantage to other players (MCV. 2017).

The developers then added vaulting and climbing which levelled the playing field for the player base as they did not need to learn an exploit to gain the movement advantages.

This mechanic in PUBG gives the player more choice in navigating their environment. Most objects and structures can be climbable if there is space for it.

This makes movement feel more dynamic than standard move, sprint, crouch, and jump mechanics, which helps improve game feel.





PUBG (PUBG Studios. 2017)

ACADEMIC READINGS

A PLAYFUL PRODUCTION PROCESS

Lemarchand talks about core loops and how game designers generally focus on this for a vertical slice (Lemarchand, 2021).

A core loop may just be run/jump/climb, however there may be more underlying mechanics that are required to showcase the core loop effectively such as objects to climb or jump over.

I have used this to plan supporting mechanics that should help improve the core mechanics, instead of only focusing on the core mechanics directly.

THE ART OF GAME DESIGN

In this book, (Schell, J. 2020), Schell talks about different workflows such as waterfall, spiral, and agile. I have decided to use an agile workflow for my project as this will give me the most flexibility in iterative design.

Schell has stated ten tips for productive prototyping where they discussed useful techniques for prototyping. They talk about forgetting quality, and quickly getting the mechanic loop implemented, either digitally or physically, so you can see how it will function and if you can take it further.

GAMES DESIGN WORKSHOP

This book (Fullerton, T. 2019), talks about the importance of playtesting and also ways to gain valuable feedback from them.

This section has helped inform me of what to look out for from my playtesters, how to read their actions and what it may mean for my game.

Fullerton also notes how many designers only use play testers near the end of production for bug tests. I've used this to plan playtests earlier in development to allow time for iteration.

THE DESIGN OF EVERYDAY THINGS

This book (Norman, D.A. 2013) talks about feedback and how I could use it to improve a players experience with my prototype. For my mechanics I will need to ensure I give players feedback on time, whilst balancing the amount to ensure I give them enough feedback without overloading them.

Feedback could be given to the player via sound effects, visual effects, even haptic feedback if I design for devices that support haptics.

ADDITIONAL RESEARCH

GAME FEEL

A video by Game Maker's Toolkit (Game Maker's Toolkit. 2015) discusses "game feel" and ways to apply it to games. I have taken some notes that I aim to use in my development later.

- Use exaggerated visual effects to make inputs more impactful. E.G. exaggerated sword trails, large blood trails, flashing enemies taking damage.
- Make sound effects bassy and meaningful. Avoid quieter subtle sound effects.
- Screen shake can help give more impact to mechanics. Combined with small pauses can be a good effect.

COMBAT SYSTEMS

Another video by Game Maker's Toolkit (Game Maker's Toolkit. 2018) talks about combat systems specifically. I noted down a few points that I could focus on in my designs to help improve how they function and feel.

- Each type of attack should have something unique about it to give the player a reason to choose that attack over another. E.G. fast attacks, long range attacks or stunning attacks
- Players shouldn't need to rely or play around defensive moves as this can get boring. E.G. Early assassin's creed games allowed you to block without consequence so you could repeatedly wait for the enemies to attack and instantly counter them.
- Animation cancelling should be balanced. The more animation cancelling allowed, the more enemies should be able to punish you.

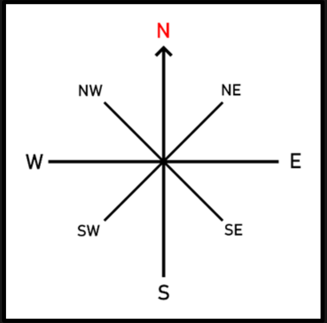
I will also be conducting research into how to develop specific mechanics in the development stage. This will likely be YouTube tutorials covering mechanic creation along with developer forums such as StackOverflow and the Unreal Engine forums.

BASIC MOVEMENT

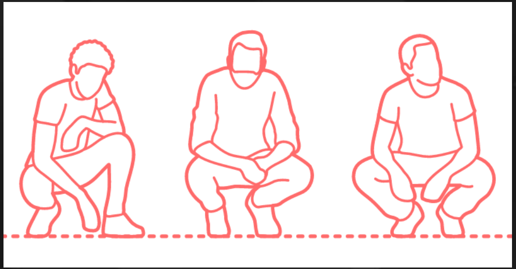
WALKING/RUNNING

The player should be able to walk and run in 8 directions depending on which movement keys are being pressed.

When sprinting the player should move at twice the speed they walk at. They will be able to sprint in all directions as walking.



Movement Directions



Crouching (Dimensions. 2021)

CROUCHING

The player should be able to crouch which will make the player model crouch down to roughly half the height of standing (Determined by sourced animations). The hitbox should also change to allow the player to crouch under objects. The player should move slower while crouching.

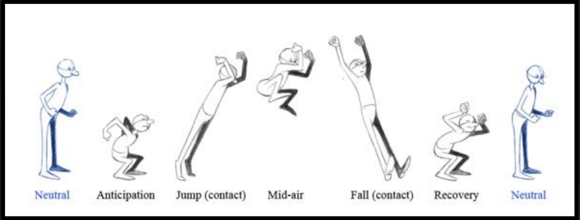
JUMPING

While grounded AND their weapon holstered the player should be allowed to jump at any time (unless there is something above their head). The jump should retain momentum from any movement beforehand.

While it is not realistic, the player should be given some air control to help make micro adjustments and improve game feel.

Ideally, the jump will have 5 different stages as shown in the image below.

This will be reliant on the specific animations I can source however and may need adjusting later.



Jump Stages (Cloy Toons. 2014)

CONTROLS

W: Forward
A: Left
S: Back
D: Right

Left Shift: Sprint
Left Control: Crouch
Space: Jump

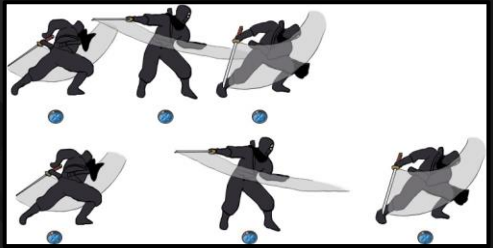
MELEE COMBAT

Thermal Katana (Kharitonov, V. 2022)

BASIC ATTACKS

The weapon of I'll be using in this prototype is a katana. The player will have two basic attacks controlled by the mouse buttons.

- Light Attack: This attack should be one handed (animation dependent) and deal light damage. This will have a quick windup and recovery to allow the player to quickly move away after attacking.
- Heavy Attack: This attack should be two handed (animation dependent) and deal heavy damage. This attack should have a longer windup and recovery time.


Sword Combos (Hungercalling. no date)

ATTACK COMBOS

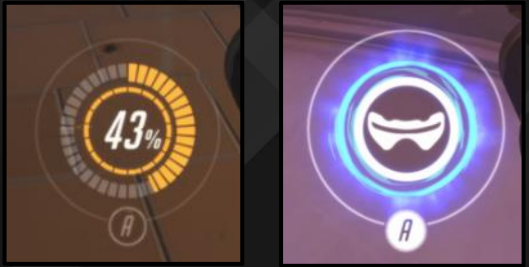
To give the player more options in combat, the player should be able to combine attacks together to perform combos. These combos will apply additional effects to the enemy if successfully completed. The players will have to input the correct attacks at the correct time, however there should be some leeway to allow players to time them correctly.

Combos should be three attacks long and the player should be able to alternate between light and heavy attacks as they please.

CHARGE METER & ULTIMATE

Whilst attacking and dealing damage with the katana, the player will build up a charge meter. To further encourage the use of combos, bonus charge meter will be given for successfully completing combos.

Once the charge meter is fully charged, the player can activate the katana for the next 10 seconds. This turns on the katana (changes texture to thermal variant) and any attacks made while the katana is activated will deal bonus fire damage.

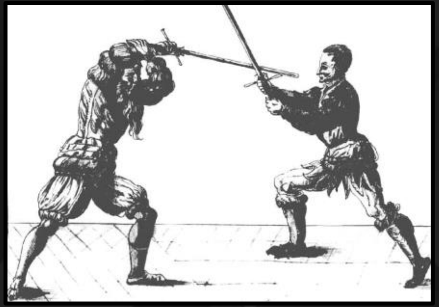


Charge Meter (Blizzard Entertainment. 2016)

MELEE COMBAT

PARRYING

To allow the player to protect themselves from enemies attacks they should be able to parry attacks to negate the damage taken. There will be a small window that the player can successfully parry an enemies attack that will stun the enemy temporarily allowing the player to get a free attack. This will cost stamina to ensure the player can't parry repeatedly.



Parry/Block Attack (Clements, J. 2002)

SNAPPING/TARGET LOCK

To make it easier to attack enemies without being positioned perfectly, when starting an attack if the player is attacking close to an enemy it should snap to the enemy and lock on, ensuring the attack lands. The tolerance will be small for this as I don't want to make it too easy for the player.

ANIMATION CANCELLING

Animation cancelling will only be allowed for light attacks. Heavier attacks cannot be cancelled to help them feel more dramatic and heavy hitting. This should make the player think more about their attacks before initiating them.

DODGING

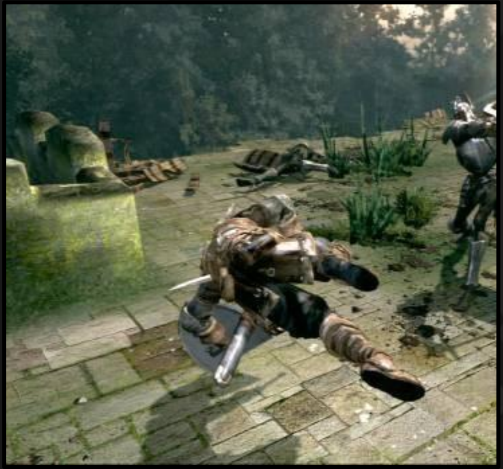
GENERAL

The prototype is focused around melee combat, so I wanted to give the player a way to quickly avoid attacks. This mechanic has been inspired by the dodge move from Dark Souls.

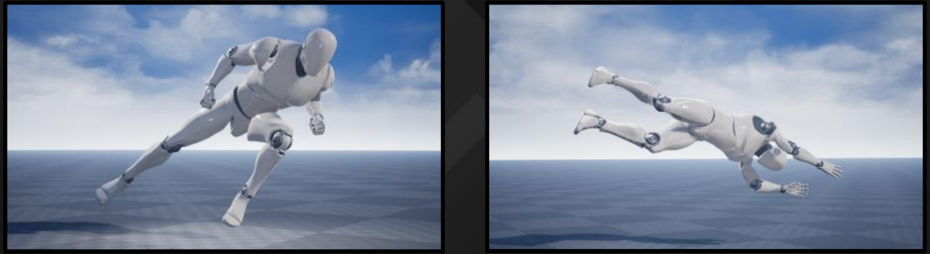
The dodge will be a dodge roll move where the player will jump forwards and land with a roll. Whilst in this animation the player should be invincible to all damage sources. It should cost stamina to dodge.

DIRECTIONAL DODGING

To decide which direction to dodge, the player would hold a movement key before starting the dodge. This will make the player dodge in that direction. If no movement key is inputted, the player should dodge forwards by default.



Dark Souls (FromSoftware. 2011)



Dodge Animations (Hammerhead – Animations. 2018)

CONTROLS

The player will use Space to initiate a dodge.

Whilst this is the same key as jumping, the player will be unable to dodge while holstered and unable to jump while a weapon is drawn so there won't be any overlap.

CLIMBING & VAULTING

GENERAL

Climbing and vaulting will be used to allow the player to traverse the environment with more creative movement.

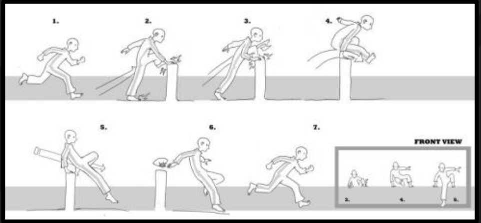
The player controller should automatically decide whether the terrain requires a climb or vault manoeuvre to progress.

To initiate a climb or vault manoeuvre, the player should move towards an obstacle and press the jump key.

VAULTING

The player can vault over waist high obstacles or below.

To move completely over the obstacle the player should continue holding forward otherwise they will stop halfway and stand on top of the obstacle if the environment allows for it. (Some objects may be too thin to stand on top of it)

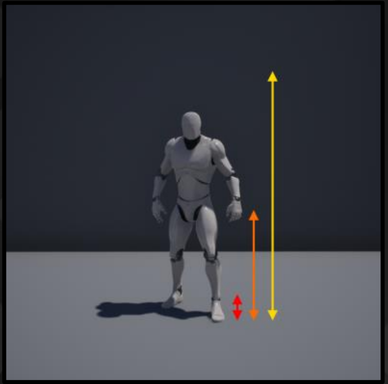


Vaulting Animations (Pau, A. 2010)

HEIGHTS

The image to the right shows the maximum height of obstacles the player can climb, vault or step over.

This is shown on the default UE4 skeleton, but the proportions will stay the same if the model changes.



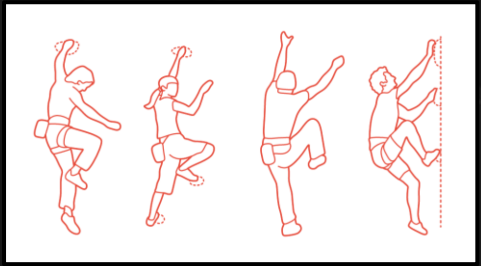
Different Heights

CLIMBING

Similar to vaulting, players can climb over obstacles a little over their character height by stretching their arms out and climbing up.

As with vaulting the player should be able to stop climbing halfway and stand on top of the obstacle if desired.

For a stretch goal, jumping into a climb would allow the player to climb even taller obstacles. However this may be more complicated and is not a necessary feature for the MVP.



Climbing Animations (Dimensions. 2021)

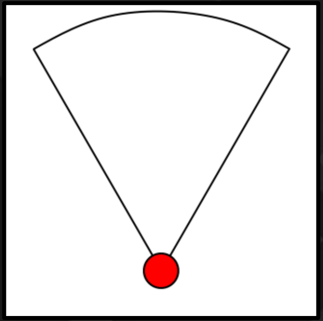
AI ENEMIES

MOVEMENT

Enemy movement should be kept fairly simple as they are not the focus for the prototype but are instead used to showcase the melee combat mechanics. There will be two movement states the enemies will be in: Guarding and Patrolling. Guarding enemies will stand still watching an area whereas patrolling enemies will move between set points.

Enemies will have an attack range and vision depending on their position and if the player enters this the enemies should move towards the player in an aggressive stance. If the player moves out of the enemies vision or range, they should return to their original position.

Enemies can't climb or vault like the player and should instead try to avoid the obstacle. This may depend on how the navmesh works with the level.


Enemy Vision Cone

ATTACKING

If the enemy is in range of the player, they will start attacking the player. The specific attack used should be decided randomly and deal differing damage depending on what was used.

Enemies should continuously attack the player until:

- Player is killed
- Enemy is killed
- Player moves too far away
- Enemy loses line of sight

SUPPORTING MECHANICS

DRAW/HOLSTER WEAPON

The player should be able to draw and holster the weapon to swap between combat and movement states. Each movement state will have different interactions.

In movement mode the player will be unable to attack or dodge roll, but will be able climb or vault.

The player will use 1 on the keyboard to draw and holster their weapon.

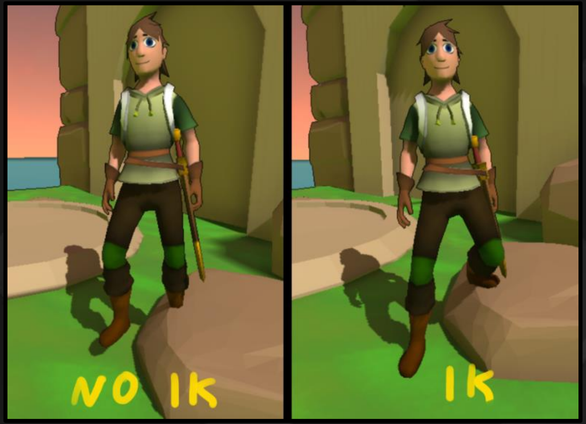
There should also be some limitations to when they can draw/holster the weapon such as jumping, mantling, dodging, or attacking.

INVERSE KINEMATICS (IK)

To help animations line up better, inverse kinematics should be used to more precisely place the feet and hands. This will be beneficial when using sourced animations as I won't be making animations specific to my scenarios.

For the feet, this will ensure they stay on the ground and have minimal clipping when walking up and down slopes.

For the hands, positioning them more accurately while vaulting or climbing will help improve immersion and game feel.

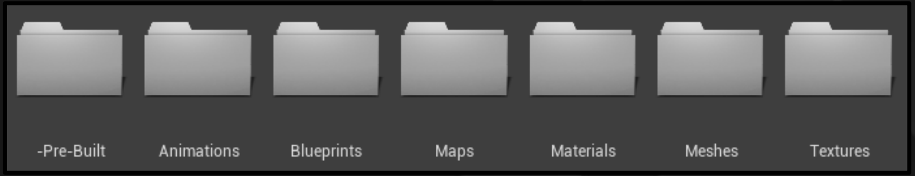


IK vs no IK (Lantz, P. no date)

DEVELOPMENT – PHASE #1

UNREAL ENGINE

I started out using Unreal Engine's third person controller template which came with a basic controller and test environment. I then setup a folder structure to try and keep everything organised as this would speed up my workflow later.



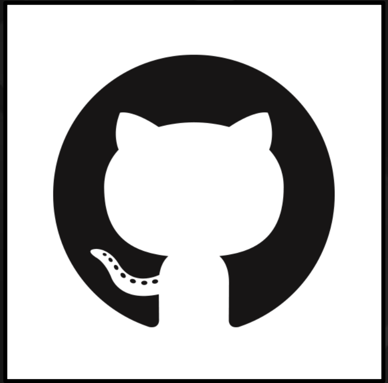
Folder structure in Unreal Engine

GITHUB & GITHUB DESKTOP

To mitigate loss of work, I also set up source control for the project via GitHub. I created the repository online and then used GitHub desktop to pull, push and update the project.

I setup different branches for each week I worked on the project and kept branches as backups to ensure I could revisit past work if needed.

The repository can be viewed here: https://github.com/LiamRickman/DES311_Demo



(GitHub. no date)

DEVELOPMENT – PHASE #1

Mixamo (Crunchbase. no date)

MIXAMO ANIMATIONS

Before starting to create my character controller I sourced animations for my character as I would be highly dependent on them as I will not be creating these myself. I found Mixamo online which has a free asset library of high quality animations and characters.

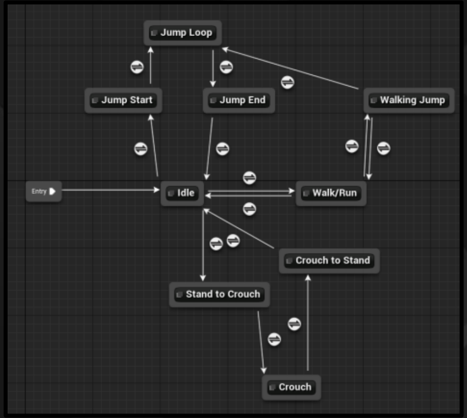
If first chose a character (Ely by K.Atienza) and downloaded numerous animations that I required. Most animations for this project were taken from Mixamo, with a used asset list available here.

BASIC MOVEMENT

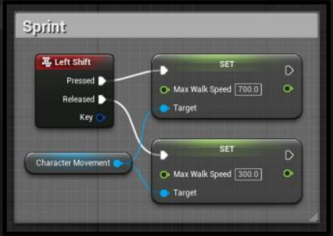
Unreal Engine's third person template provided the walking and jumping blueprints, however I added crouch and sprint functionality to complete the basic movement.

I also created an animation blueprint for the player to control which animations are played. This involved creating blend spaces for walking, sprinting, and crouching and updating variables to trigger animation changes.

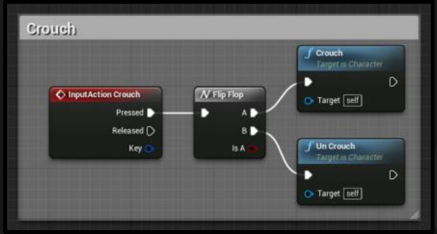
Gameplay of the basic movement can be viewed here.



Player animation blueprint



Sprint function



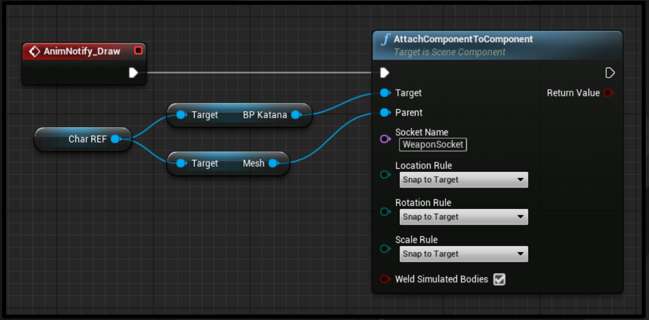
Crouch function

DEVELOPMENT – PHASE #2

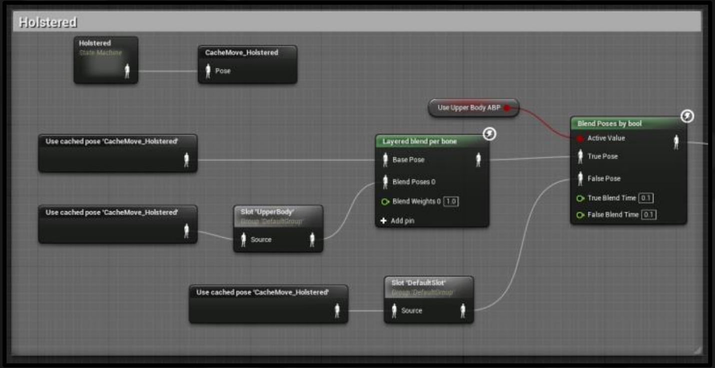
DRAW/HOLSTER WEAPON

This week I worked on implementing weapon drawing and holstering to allow the player to swap between standard and combat movement. This works by playing an animation and then either attaching the katana to the players hand or hips depending on the animation played.

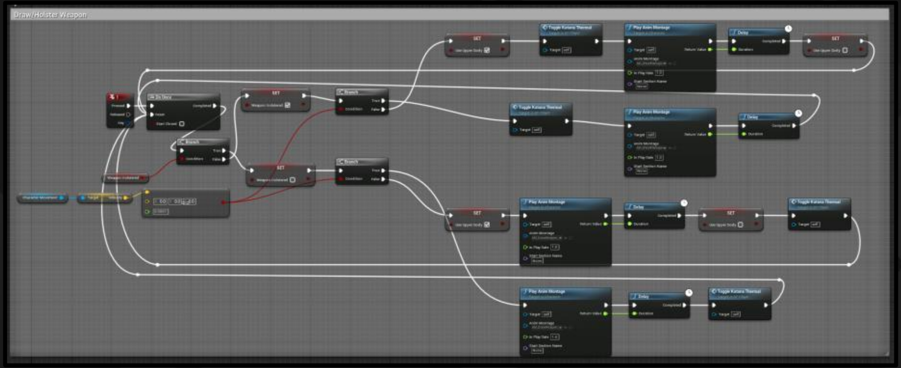
Along with this, I implemented upper body animation blending as this will allow animations to be played while moving to improve the visual feel of the game.



Attaching katana to hand



Upper body animation blending



Swapping weapons function in player controller

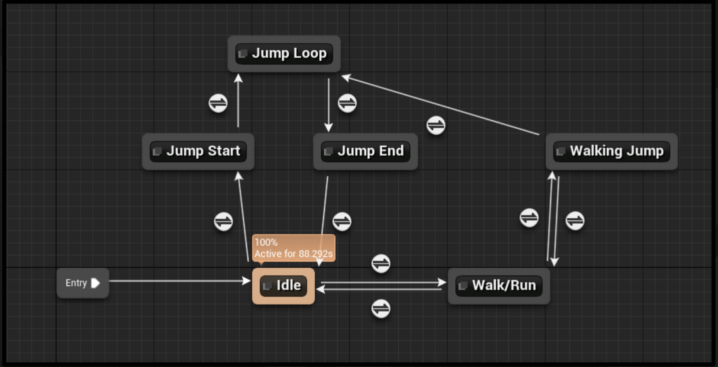
DEVELOPMENT – PHASE #2

COMBAT MOVEMENT

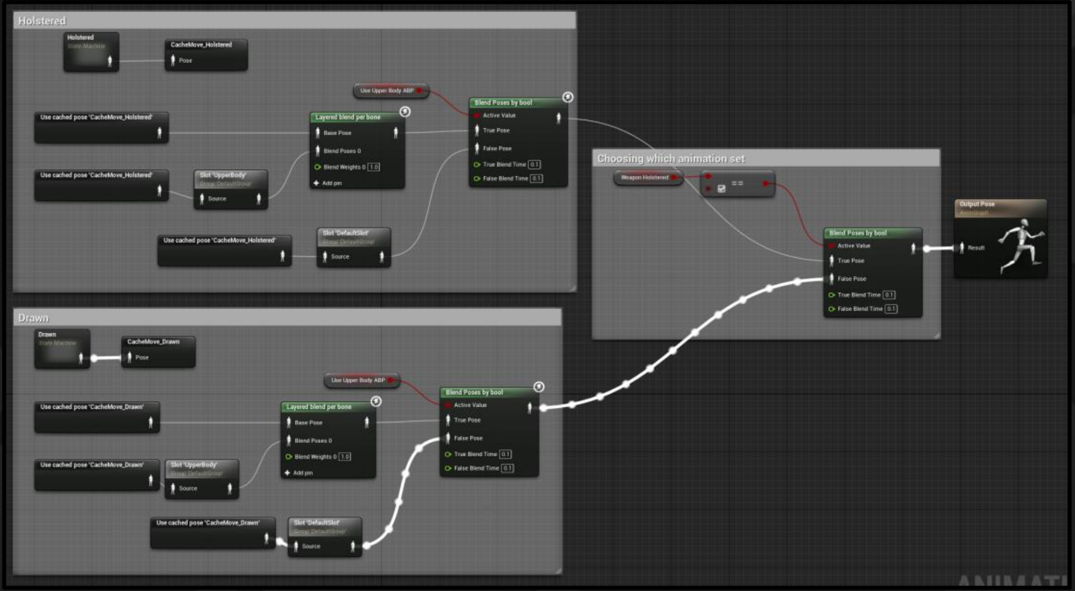
After drawing and holstering the weapon was implemented I worked on adding the combat movement state to the player. This ended up being similar to the standard movement state, however it did not feature crouching.

I implemented some code that would blend between the standard and combat move states depending on if the weapon was drawn or not.

Combat Movement demo here.



Combat state animation graph



Player animation graph with combat and standard movement blending

DEVELOPMENT – PHASE #3

NEW PLAYER MODEL

In this phase, I came across an issue while looking for animations. The Mixamo character I had been using previously had problems when retargeting animations. It had errors when targeting animations meant for the unreal mannequin. I knew this would limit my options for animations which may hinder progress and development so I decided to look for alternate character models, preferably one that matched the Unreal mannequin's skeleton.

Paragon: Twinblast – Shadow Ops (Epic Games. 2018)

I came across the Paragon assets that Epic Games had put up on the Unreal marketplace for free. These are high quality AAA character models that matched the UE4 mannequin's skeleton. I imported this to my project and began retargeting the previous character's animations to the new character. This did take a while but would be worth it later as I had more choices for animations.

ALTERNATE CAMERA ROTATION

To add more clarity when the player is in combat or movement mode, I changed how the camera movement works in each state. When in standard movement mode, the camera does not impact the character's rotation so the player can look around the environment. While in combat mode, the camera will rotate the player so they always face forward. This will help with directing attacks once implemented.

A demo of the new character model's movement and camera rotation can be viewed here:

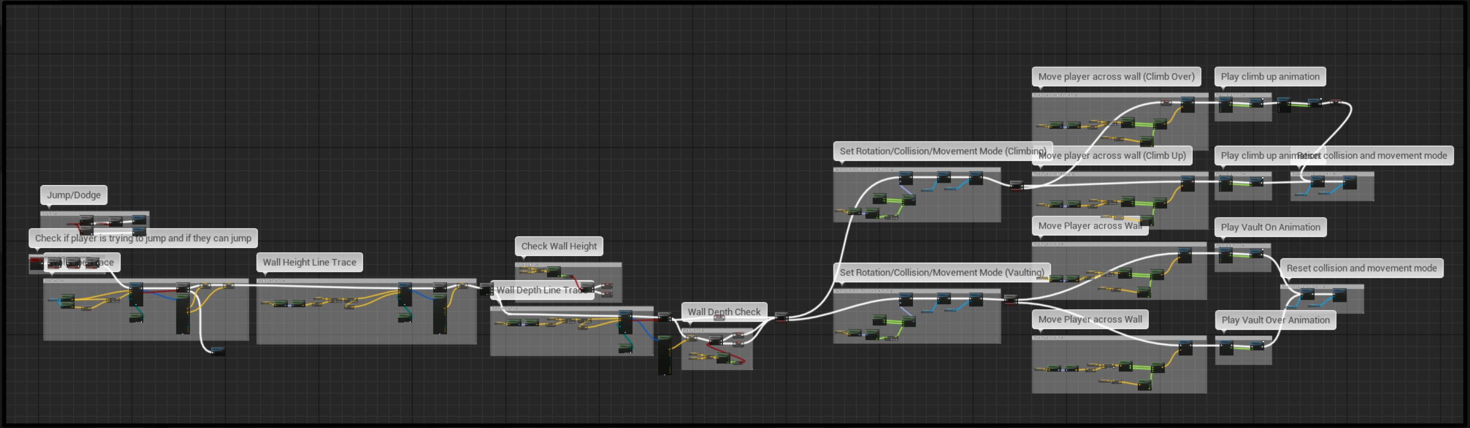
DEVELOPMENT – PHASE #3

CLIMBING & VAULTING

I also decided to work on the vaulting and climbing mechanic. I watched a few tutorials to learn how to approach creating such a mechanic (Apsland, M. 2020), (Visco. 2020). I then made my own variation which uses a line trace forward to see if the player is nearing a wall. If it finds a wall in front of the player it will then check how thick the wall is to decide whether to stand on top of the wall or go straight over. Finally it will play the appropriate animations, adjust the collision, and move the player.

As a first pass this worked alright, however in play testing it was obvious there were some bugs. First, most animations didn't line up properly and would position the player so they were floating. There was also no height check as the player could climb up and into objects that were too high, sometimes resulting in the player clipping through a wall.

A video showcasing the climbing and vaulting mechanic, along with the bugs can be seen here:

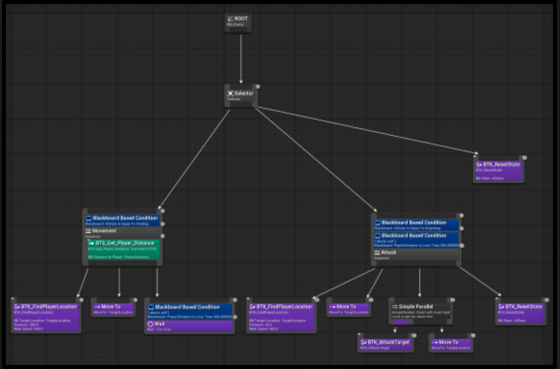

Jump/Dodge
Check if player is trying to jump and if they can jump
Wall Height Line Trace
Check Wall Height
Wall Depth Line Trace
Wall Depth Check
Set Rotation/Collision/Movement Mode (Climbing)
Set Rotation/Collision/Movement Mode (Vaulting)
Move player across wall (Climb Over)
Play climb up animation
Move player across wall (Climb Up)
Play climb up animation
Reset collision and movement mode
Move Player across Wall
Play Vault On Animation
Reset collision and movement mode
Move Player across Wall
Play Vault Over Animation

Climbing and vaulting mechanic

DEVELOPMENT – PHASE #3

ENEMY AI

I decided to learn about AI this week to see how challenging it would be to implement the basic AI I had designed. I used a behaviour tree AI system after watching a tutorial (Laley, R. 2020) to create a follow AI that would follow the player around until they reached a certain range. This can be seen here.



Behaviour tree

I then expanded this AI to include attacking randomly when they were in range. This felt very uniform as the enemies wandered around not feeling very natural with their attacks. I left this for now as I wanted to test how the AI would perform when in a group.

The tutorial included an algorithm to calculate which enemy should attack next, giving priority to those in front of the player. I tried implementing this, however it seemed to get stuck with one enemy attacking at a time. Considering the amount of enemies I was testing, they also did not attack very often which seemed unrealistic.

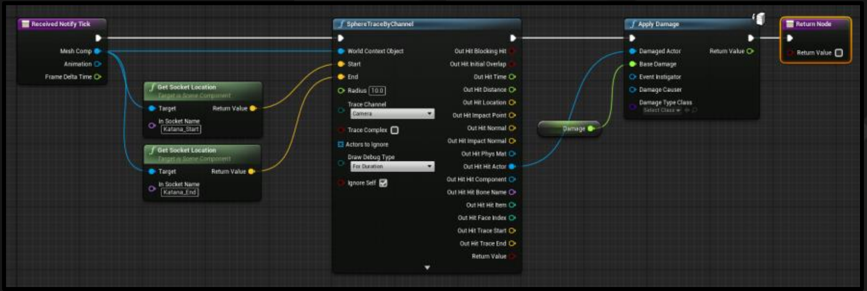
Group attacking AI demo.

BASIC ATTACKS

I decided to leave the enemy AI for now as I was unsure which direction to take them. I started implementing some basic attacks using two different animations for light and heavy attacks.

These could be used to damage the enemies, but could not kill them just yet as I have not implemented health yet. I used red debug lines to see when damage would be dealt to enemies.

Basic attacks demo here.



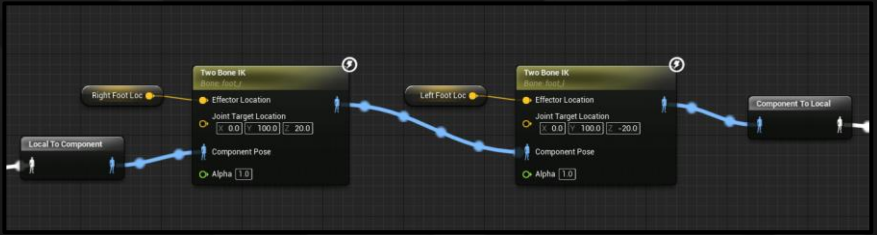
Behaviour tree

DEVELOPMENT – PHASE #4

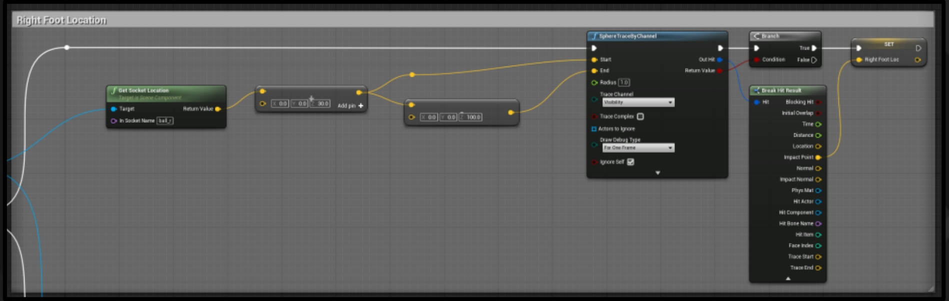
INVERSE KINEMATICS (IK)

In this phase, I worked on implementing the IK foot movement as I wanted to learn how complicated it would be for future development. I watched some tutorials and worked through the process. I started with editing the feet location, then moved onto the knee IK. This had some obvious issues with the knees often bending the wrong way. I fixed this later to end up with a mostly fully working Leg IK system.

There were still some issues with this system, particularly the transitions which felt too snappy and unrealistic. A major bug also appeared where the legs would sometimes get stuck on the player shoulders which resulted in some strange effects.



Moving feet in animation blueprint



Line trace used to calculate foot position

DEVELOPMENT – PHASE #4

Plugins used

CONTROL RIG & FULLBODY IK PLUGINS

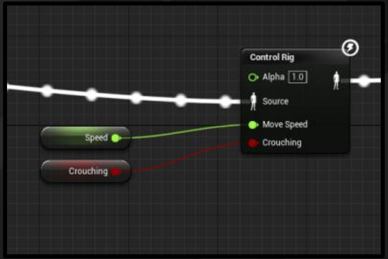
Whilst researching potential fixes for my original IK system, I came across two plugins for Unreal Engine that are in beta stage. I found a tutorial that explained how these plugins worked and decided to try implementing this to see if it fixed any of the issues I was having previously. This system uses a blueprint style interface to calculate feet and bone positions.

I first created feet line traces similar to the previous system I used. The pelvis was then moved down to match the lowest foot position – something I hadn't done with the original version.

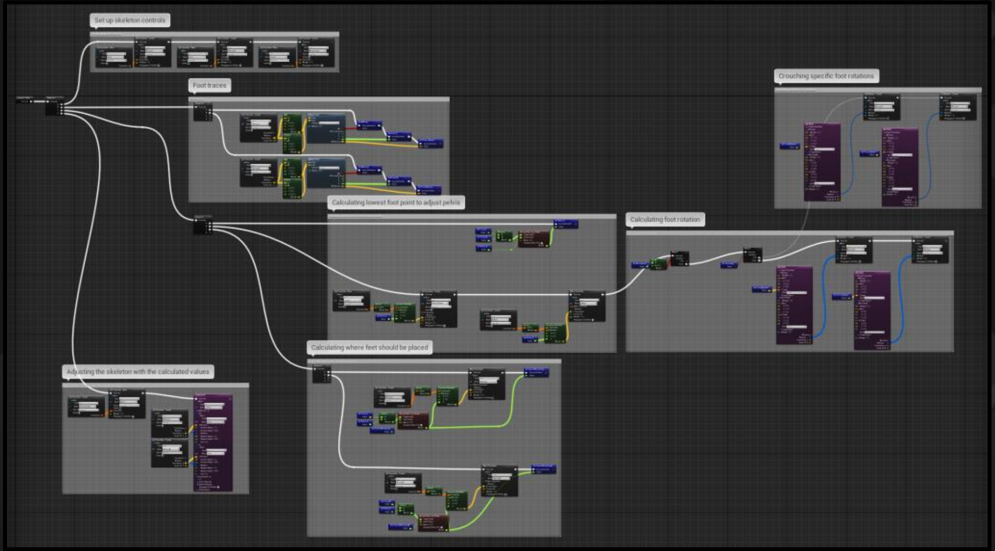
Then I added feet positioning to move the other foot up to match the ground height. Finally I added feet rotation to help match them with sloped ground. I only used this when standing still as it looked awkward whilst moving.

After testing, I also adjusted the crouched rotations as they did not function as expected.

Demo of control rig and fullbody IK here.



Player animation blueprint


Set up skeleton controls
Foot traces
Crouching specific foot rotations
Calculating lowest foot point to adjust pelvis
Calculating foot rotation
Calculating where feet should be placed
Adjusting the skeleton with the calculated values

Control rig blueprint

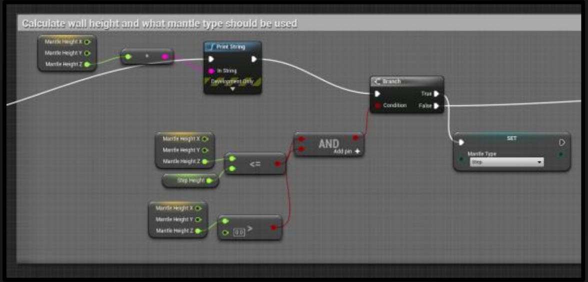
DEVELOPMENT – PHASE #5

REWORKED VAULTING AND CLIMBING

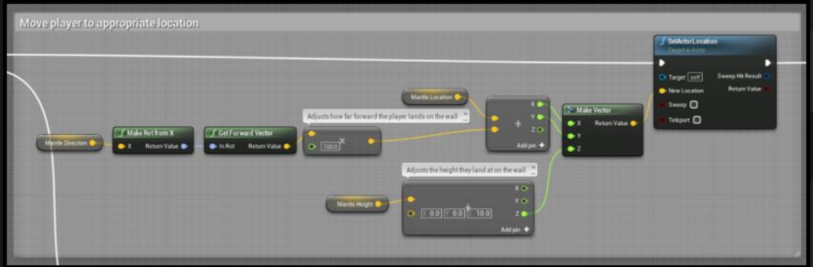
This week I wanted to fix the vaulting and mantling system as it had a few bugs that impacted the overall feel and polish of the mechanic. First I calculated the height of the wall to stop the player climbing up walls that were too tall.

To improve how the animations lined up, I created individual movements for each animation that could be adjusted separately to more accurately line up the animations.

The improved climbing demo can be seen here



Calculating wall height



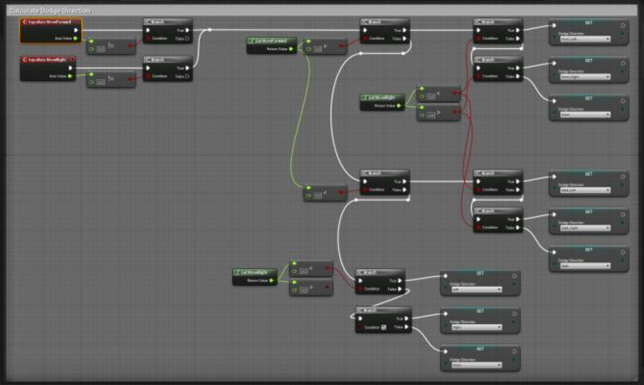
Move player to appropriate location
SetActorLocation
Mantle Location
Adjusts how far forward the player lands on the wall
Make Vector
Mantle Direction
Make Rot from X
Get Forward Vector
Adjusts the height they land at on the wall
Mantle Height

Moving the player per climb type

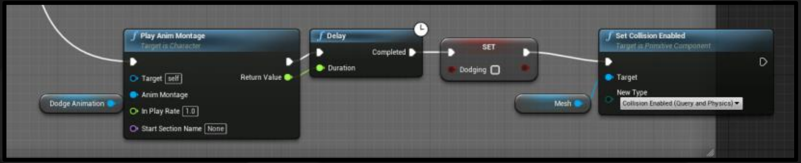
DIRECTIONAL DODGE

The dodge mechanic has been implemented and checks which way the player is trying to dodge by updating a enumerator based on their inputs. When the player tries to dodge, the animation is updated depending on what inputs they have used and a root motion animation is played to move the player. Collision is also disabled to ensure the player doesn't take damage.

Dodge demo here



Calculating dodge direction



Playing dodge animation

DEVELOPMENT – PHASE #5

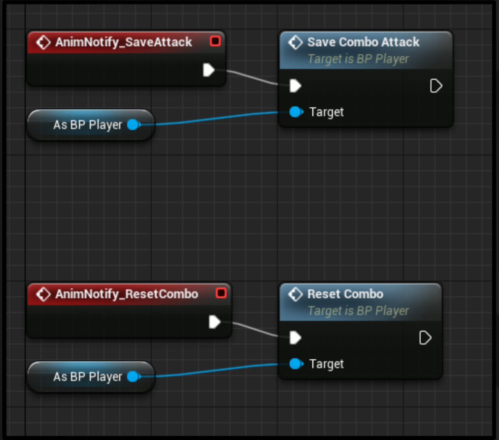
ATTACK COMBOS

I worked on the attack combos this week as I had only implemented basic attacks at this stage. Performing combos checks whether the player has tried to perform one via inputs and if they have in time the animation will trigger another attack to be performed.

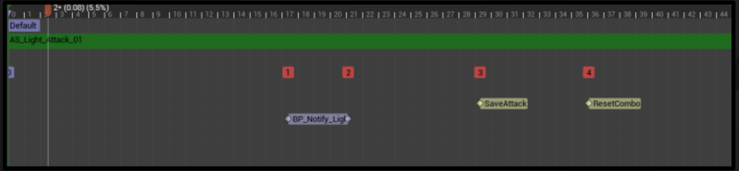
I had some issues (here) with root motion animations within Unreal. The mixamo animations I was using did not fully support root motion due to having no root bone by default. I downloaded a blender plugin (enziop. 2021), and was able to convert the animations to work properly.

I also used an enumerator to store which combo had been performed. This could be used for bonus effects later depending on the combo used.

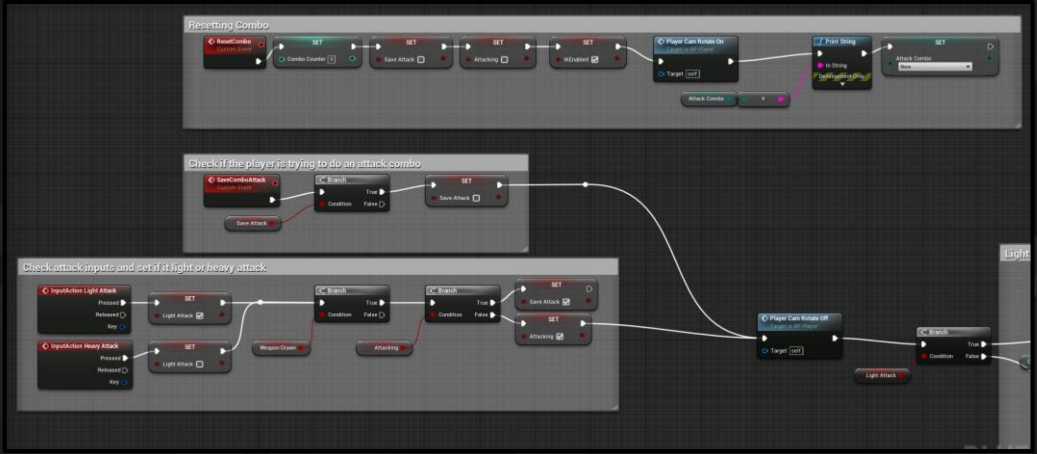
The full attack combos can be seen here



Playing attacks via animation blueprint



Animation montage setup



Performing attacks via inputs or animations

DEVELOPMENT – PHASE #5

REWORKING ENEMY AI

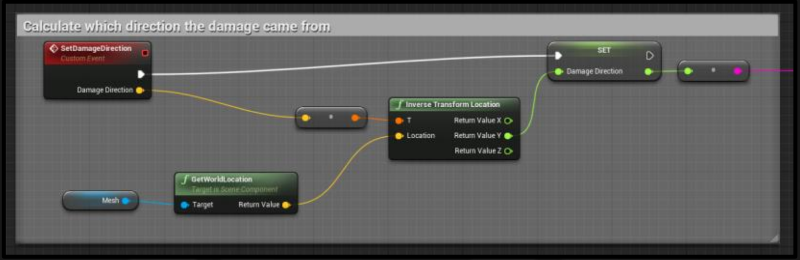
Previously I wasn't happy with how my enemy AI were functioning. They felt too artificial and lifeless, so I decided to remake them from scratch now that I knew more about behaviour trees and creating AI.

I started with a test dummy that the player could practice their attacks on. This dummy reacts to the player's attacks with an animation depending on where the attack came from.

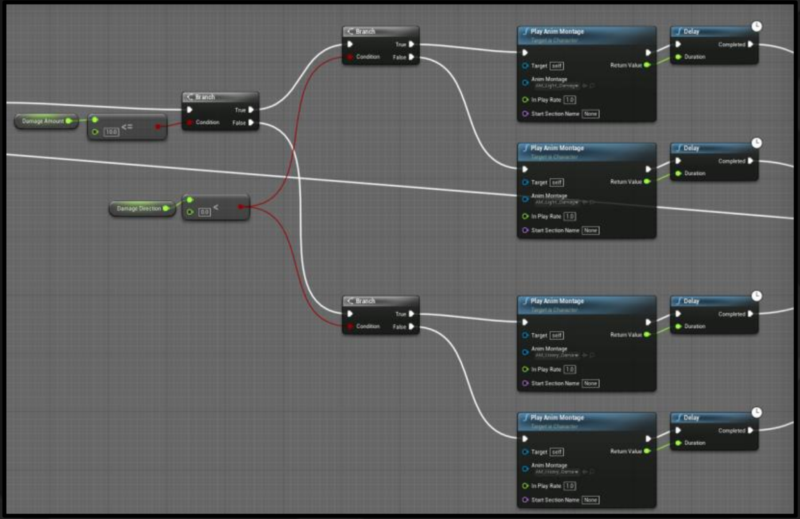
Unfortunately, I am limited by the animations I could source for the reactions, and the ones I could source were very similar so it is quite difficult to tell the difference between them.

If I have extra time later I will try to source more unique animations for each, however this may be impossible due to not making the animations myself.

A demo of the test dummy can be seen here



Calculating damage direction



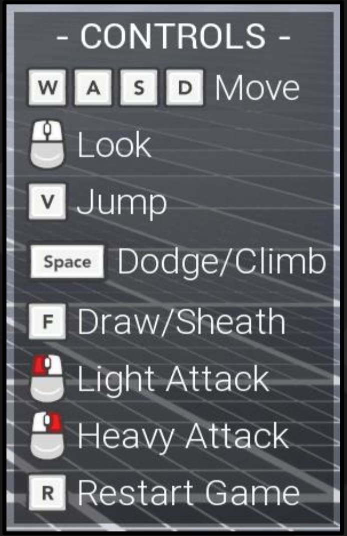
Playing damage animation

DEVELOPMENT – PHASE #6

CONTROLS INTERFACE

This week there was a planned playtest on campus for us to get feedback from classmates about our current progress.

Before the playtest I added a controls interface to tell the player how to play the demo. This was placed on a widget that would display all the time. I used an asset pack (Berbece, N. and Paun, P. 2021) to show button presses.



- CONTROLS -

W A S D	Move
	Look
V	Jump
Space	Dodge/Climb
F	Draw/Sheath
	Light Attack
	Heavy Attack
R	Restart Game

Controls interface

FIRST PLAYTEST AND FEEDBACK SESSION

At the playtest, I was able to get new players to test my demo and give me feedback. Unfortunately we were unable to do any data collection or feedback forms due to data privacy, however we could note down verbal and visual feedback.

I created bullet points online using Notion to keep track of feedback and areas to improve. This week I was focusing on initial feel, how the mechanics feel – whether they feel impactful and fun to use. I also kept an eye out for any bugs that appeared.

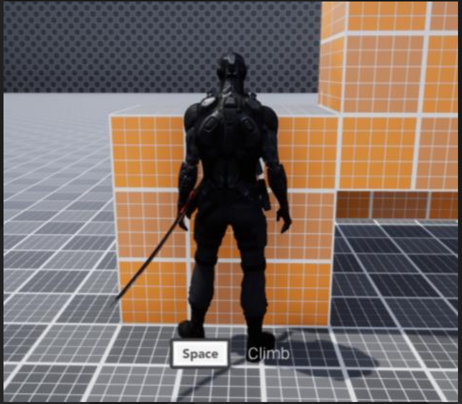
The full feedback list for this playtest can be seen here

DEVELOPMENT – PHASE #6

CLIMB/VAULT PROMPT

One reoccurring theme I saw in the playtest was that players did not always know what they could climb or vault. I have added a prompt that shows the keypress and what type of action they will perform when they get close to a wall. This UI only appears when near a climbable/vaultable wall.

I did had to move the wall check to use the Event tick to update constantly which could lead to performance issues, however with a small scale project it is not an issue just yet.



Climb/Vault prompt

DYNAMIC STAMINA SYSTEM

I decided to add a stamina system briefly mentioned in the melee combat designs. This will add more thought and consequences to the players actions.

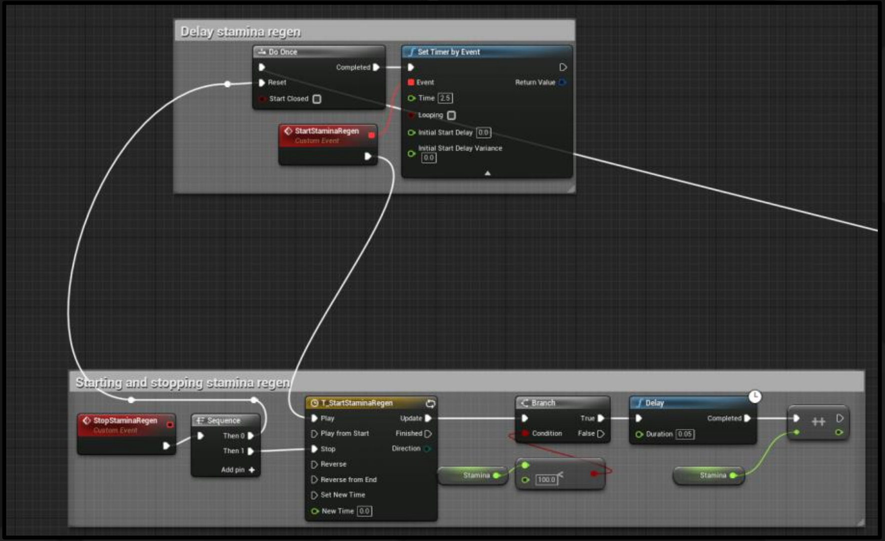
The player will use stamina for jumping, vaulting/climbing, attacking, or dodging. Stamina recovers after a short time but regeneration can be interrupted by performing actions.

A demo of stamina regeneration can be seen here

STATUS BARS

To display stamina to players I created a progress bar on the HUD. This will update as the stamina value changes on the player blueprint.

I also created a healthbar and overcharge meter, however the overcharge meter has no functionality yet.

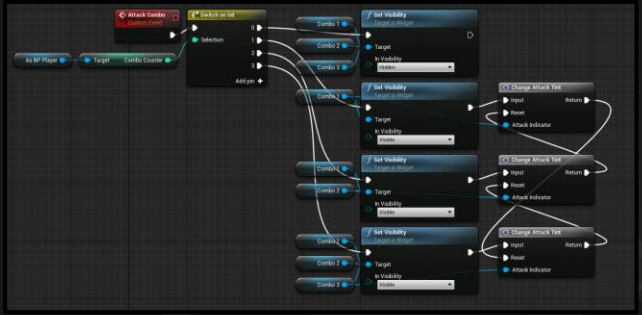


Starting and stopping stamina regen

DEVELOPMENT – PHASE #6

ATTACK COMBO INDICATOR

While I was working on the UI I also added an indicator to show the player what combo they are currently performing. This uses three bars to indicate how far through a combo they are and two colours to indicate whether they have used light or heavy attacks. Demo of combo indicator here



Attack combo interface blueprint

PATROLLING ENEMY AI

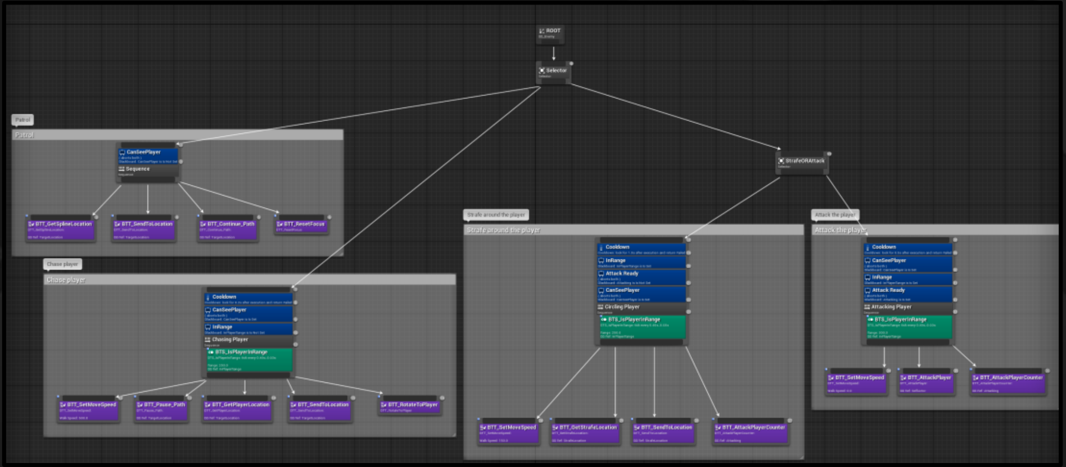
I continued working on the enemy AI to add more functionality. I decided to continue using behaviour trees rather than blueprints as I wanted to learn more about how to use them.

I started by creating a patrol movement using a spline where the enemy would follow the spline path moving from point to point.

I then added some sight perception which allowed the enemy to detect the player. This was then used to let the enemy move and chase the player if they detect the player. If they lose sight the enemy currently runs back to their original patrol.

Finally, if the enemy gets in range of the player. They will begin strafing around the players position. They have a 33% chance to attack the player with another 33% chance to perform a heavy attack

Patrolling enemy demo here



Attack combo interface blueprint

DEVELOPMENT – PHASE #7

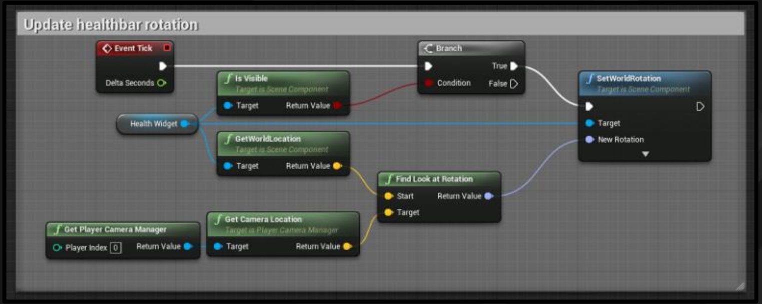
TAKING DAMAGE & ENEMY HEALTHBAR

Until now, enemies didn't actually damage the player and the player could not damage the enemies. I added this functionality using animation notifies and sphere traces to detect when either the player or enemy deals damage.

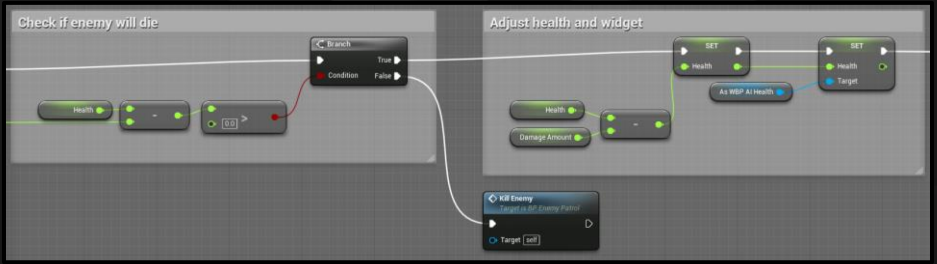
While adding this, I created an enemy healthbar so the player can see the enemies current health. I placed this above their head in the world space and it only appears when the player is in a certain range.

Finally, I added a few checks to the behaviour tree to stop the enemy attacking the player when the player dies and to stop all movement when the enemy dies.

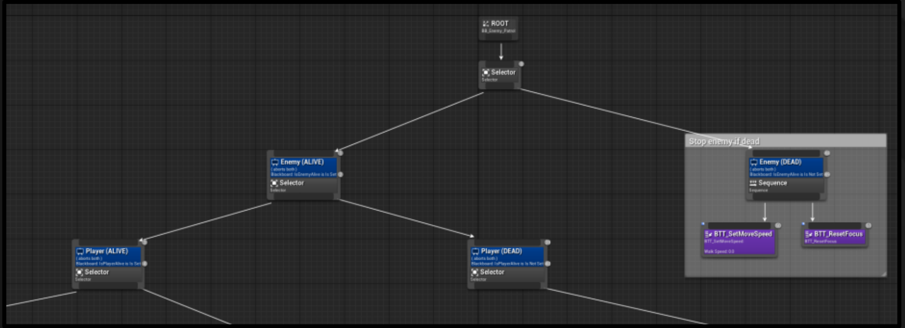
Taking damage and healthbar demo can be seen here



Healthbar faces player



Adjusting enemy health via damage trigger



Behaviour tree to check if enemy or player is alive

DEVELOPMENT – PHASE #7

SHOWCASE LEVEL

While watching play testers in my first play test it was clear there was no direction and they just wandered around level unsure what they could interact with. This was because the play test took place in my mechanics testing level.

To improve the test environment for my play testers, I decided to block out a test level that would better showcase my mechanics. I didn't bother doing any level design as I wasn't too worried about how it functions as it is more about the mechanics than the level itself. This saved time as I could just go into Unreal and block out the level a section at a time.

I created sections for: Movement & IK, Additional basic movements, Dodging, Climbing & Vaulting, Attacks, and Enemies.

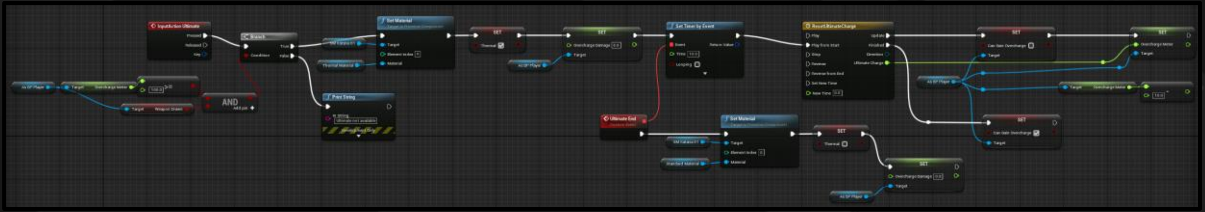
A walkthrough of the showcase level can be seen here

OVERCHARGE & SWORD TRAILS

I wanted to add more impact to the attacks so I researched sword trails. I created a material and particle effect for the trail and used a animation notify to play the particle effect for a set duration. I tried to implement more advanced sword trails, even looking into Unreal Engine's Niagara system, however I was unsuccessful.

Whilst creating the sword trails I added the overcharge mechanic as previously I had just been storing the overcharge meter from attacks with no way to use it.

The overcharge attacks now do an extra 10 damage and change the sword trails to orange and the katana colour to orange to simulate fire attacks.


Activating overcharge and changing katana material

Sword trails & overcharge demo here

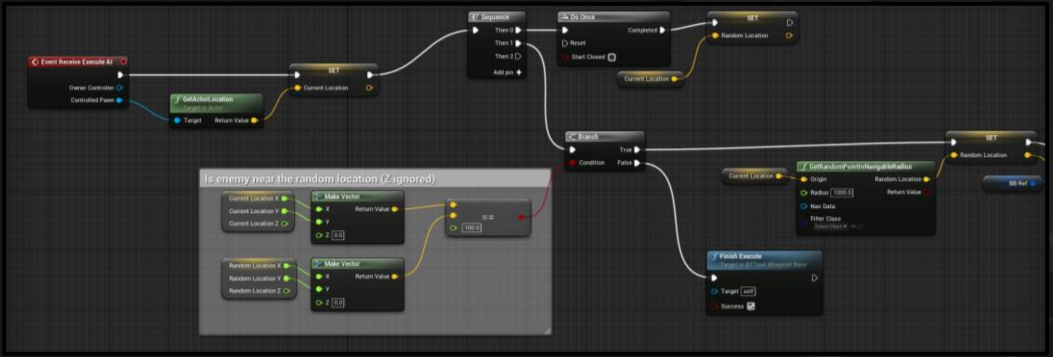
DEVELOPMENT – PHASE #7

RANDOM MOVE AI

While creating the test level I decided to make another variation of my enemy AI that would be better suited for the combat area as the patrolling enemies were a bit limited.

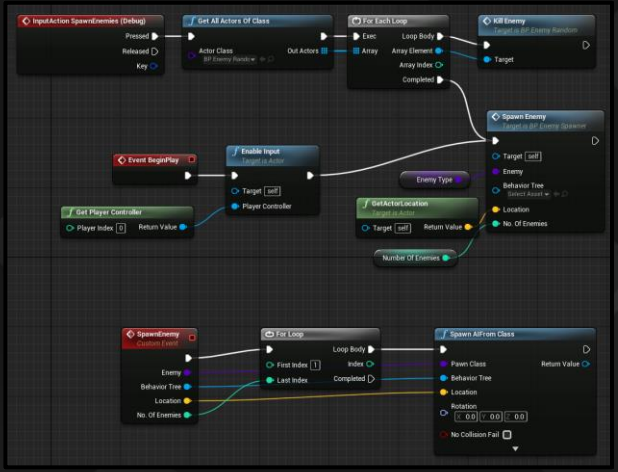
I created a copy of the patrol AI and then created a random location task that would find a random location around the enemy and would be used to try move the enemy towards it.

Random location check



ENEMY SPAWNER

I also created an enemy spawner to allow the player to practice fighting enemies as long as they want. This kills any enemies already present in the level to ensure the player can't spawn too many enemies at once.



Enemy spawner

Demo of random move AI and enemy spawner can be seen here

DEVELOPMENT – PHASE #7

SECOND PLAYTEST AND FEEDBACK SESSION

I planned another playtest this week, this would be an online session as the on-campus classes had finished for this semester. I gathered friends and classmates to playtest and provide feedback this week. This was the first test with the new showcase level which received positive feedback overall

The full feedback list for this playtest can be seen here

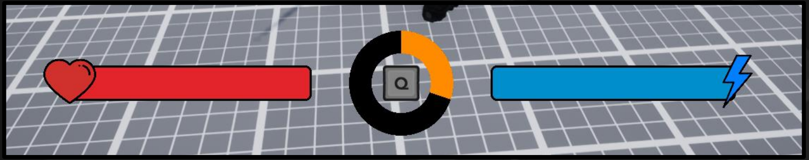
MINOR CHANGES

- I adjusted some of the controls for the game as some players complained about the layout here.
- Dodge is now on ALT instead of space
- Jump and Climb/Vault are on Space as it felt awkward jumping with V and climbing with space after
- F is used to draw/holster the weapon instead of 1 as this feels more natural. Attacking with no weapon out also automatically draws the weapon.

HUD CHANGES

I updated the HUD as I felt it was very generic and lacked some personality. I created a circular progress bar, similar to that used in Overwatch (Blizzard Entertainment. 2016) for the overcharge meter and then positioned the health and stamina bars around it

I removed the controls interface from the in-game HUD as I later added this to a pause menu.



Improved HUD

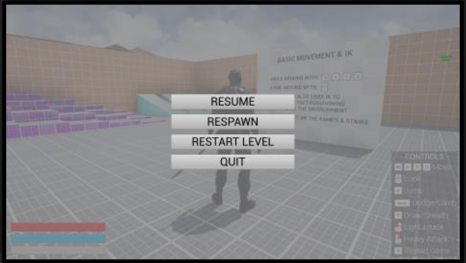
DEVELOPMENT – PHASE #7

PAUSE MENU

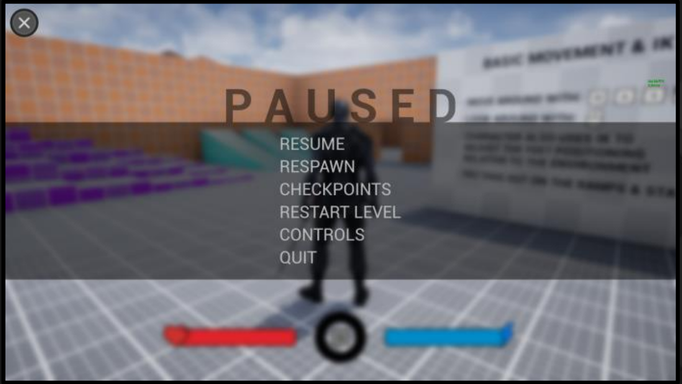
I decided to create a pause menu to allow the player to exit the game as well as respawn/restart if they come across any bugs. I originally created a very basic menu, however decided to make it a bit more polished later.

I also created a controls menu to allow the player to see the controls without having to check the mechanic board and also checkpoints which would allow the player to teleport to each area of the test level.

A video demo of the pause menu can be seen here



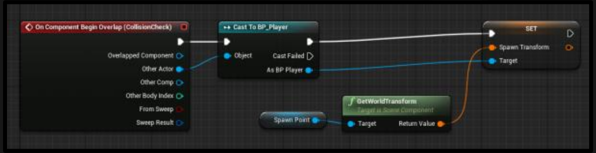
Original pause menu



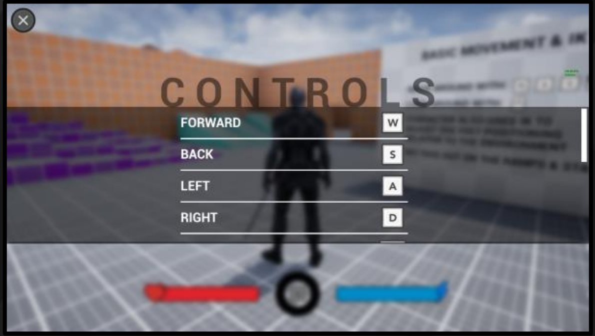
Updated pause menu

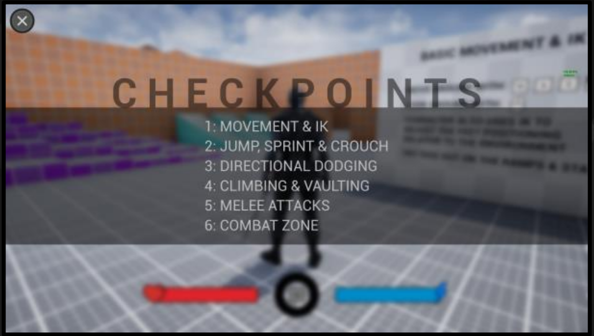
CHECKPOINTS

The checkpoints system works by setting the spawn point of the player when they move through a collision box. When they respawn after dying or selecting it in the pause menu they will spawn at the last checkpoint they passed.



Checkpoint updating spawn point





DEVELOPMENT – PHASE #8

THIRD PLAYTEST AND FEEDBACK SESSION

While I hadn't added many new features since my last play test, I had some new players who had not played the demo yet so I created an updated build and gathered some additional feedback.

This feedback can be seen here

FINAL CHANGES

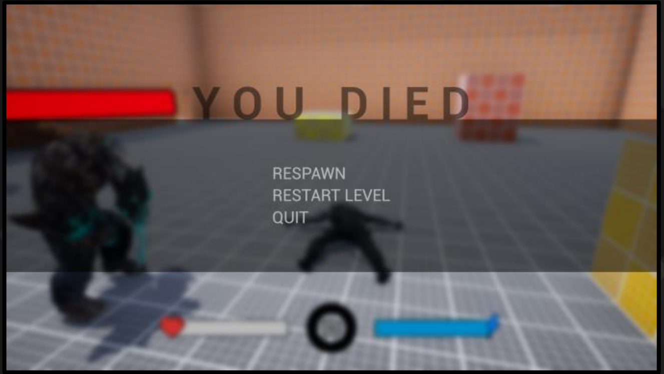
I primarily focused on bug fixing for this phase as this would be the last update to the demo before submission.

Notable bug fixes I completed:

- Enemies launching the player with attacks
- Combo meter would stay on screen too long
- Enemies interrupting player attacks left player unable to attack
- Limited FPS as it was causing some high-end PCs to run hotter due to very high frame rates

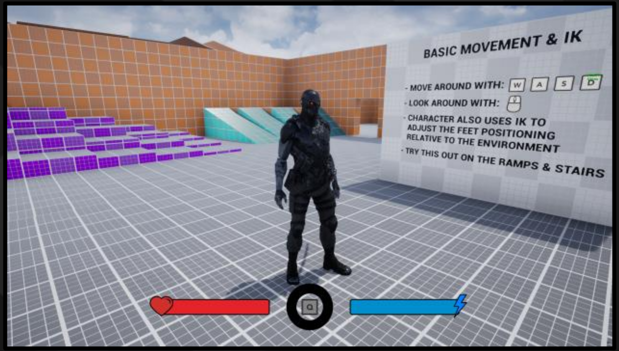
I also added a death respawn menu similar to the pause menu to give the player more choice after dying.

Enemies now have a 50% chance to attack instead of 33% as they felt slow with their attacks. They also move closer to make it easier for them to land their attacks.



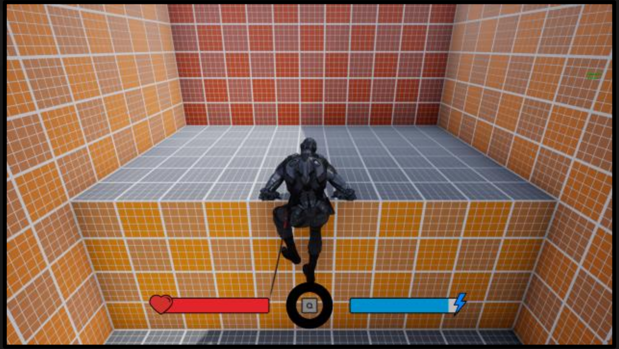
Death respawn menu

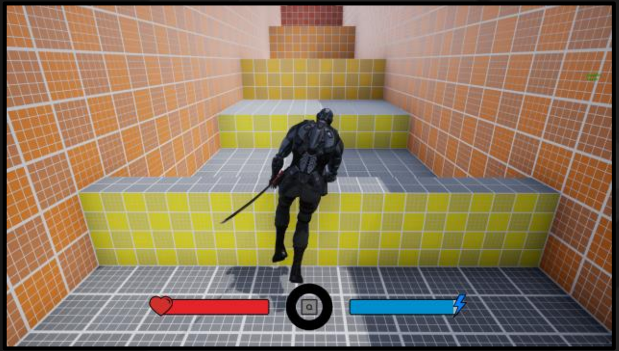
FINAL PROTOTYPE – RENDERS

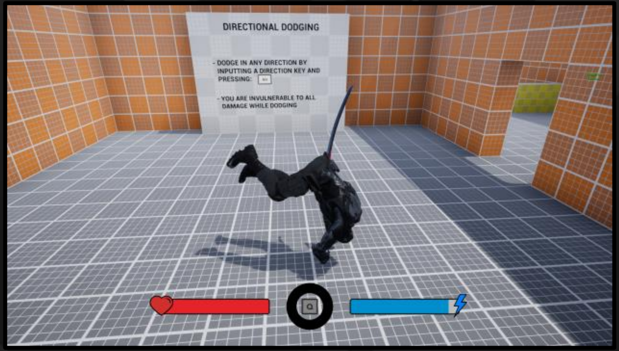

BASIC MOVEMENT & IK

- MOVE AROUND WITH: W A S D
- LOOK AROUND WITH:
- CHARACTER ALSO USES IK TO ADJUST THE FEET POSITIONING RELATIVE TO THE ENVIRONMENT
- TRY THIS OUT ON THE RAMPS & STAIRS

Player model

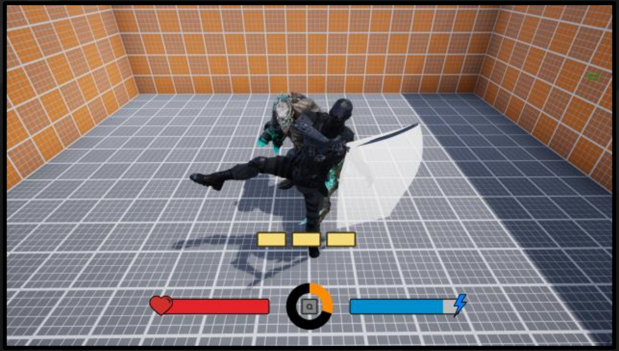

Climbing

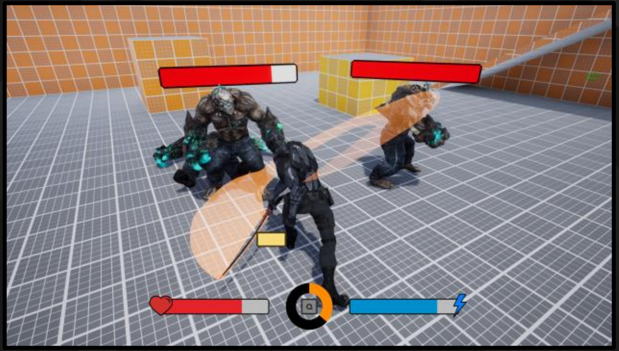

Vaulting


DIRECTIONAL DODGING

- DODGE IN ANY DIRECTION BY INPUTTING A DIRECTION KEY AND PRESSING:
- YOU ARE INVULNERABLE TO ALL DAMAGE WHILE DODGING

Directional Dodge


Attack combo


Overcharge attack

FINAL PROTOTYPE – DEMO

THIS VIDEO IS NARRATED SO PLEASE HAVE VOLUME UNMUTED

YouTube Link: https://www.youtube.com/watch?v=RL1EuZFDUPY

DEVELOPMENT LOG

WEEK #1 (31/01/22)

- Outlined skill gaps
- Brainstormed areas I wanted to cover

WEEK #2 (07/02/22)

- Started working on PDP document
- Continued brainstorming ideas for project

WEEK #3 (14/02/22)

- Created concept pitch for project
- Pitched presentation to lecturers for feedback

WEEK #4 (21/02/22)

- Catchup week for DES310

WEEK #5 (28/02/22)

- Created my learning contract

WEEK #6 (07/03/22)

- Missed week due to funeral

WEEK #7 (14/03/22)

- Finished learning contract
- Started research into game design and feel
- Started designing mechanics

WEEK #8 (21/03/22)

- Continued designing mechanics

    Basic movement, Vaulting & Crouching, Dodging, Melee Combat, and Enemy AI

DEVELOPMENT LOG

WEEK #9 (28/03/22)

- Set up Unreal Engine project
- Set up source control via Git and GitHub desktop
- Started basic character movement
- Added crouching, jumping and sprinting

EASTER WEEK #1 (04/04/22)

- Created draw/holster weapon mechanic
- Created combat move state

EASTER WEEK #2 (11/04/22)

- Imported new player model that matches UE4 skeleton
- Retargeted animations to new skeleton & setup character
- Added camera rotation for combat and standard movement
- First pass of climbing & vaulting implemented
- Added basic attacks
- Added some basic enemy AI

WEEK #10 (18/04/22)

- Started implementing inverse kinematics
- Used two bone IK system but was quite buggy
- Swapped to Control Rig and FullBody IK plugins which was more realistic and less buggy.

WEEK #11 (25/04/22)

- Reworked vaulting and climbing mechanic
- Implemented attack combos for light and heavy attacks
- Reworked enemy AI starting with a test dummy for attacks

DEVELOPMENT LOG

WEEK #12 (02/05/22)

- Added controls interface
- First playtest and feedback
- Added climbing prompt
- Added dynamic stamina system
- Added healthbar, stamina bar, overcharge meter
- Added attack combo UI
- Completed new patrol enemy AI

WEEK #13 (09/05/22)

- Enemy and player can now damage each other
- Added basic respawn menu
- Updated enemy behaviour tree to adapt to enemy and player dying
- Added enemy healthbar
- Built new test level
- Created random move AI variation

WEEK #13 cont.

- Added sword trails
- Overcharge now does damage
- Second playtest
- Created pause menu
- Created checkpoints
- Updated healthbar, stamina bar and overcharge meter UI

WEEK #14 (16/05/22)

- Third playtest
- Bug fixes
- Death respawn menu added
- Improved enemy attack rate
- Worked on PDP and submission documents

WEEK #15 (23/05/22)

- Finished PDP documentation

REFERENCES

- Apsland, M. (2020). Vaulting & Climbing (Parkour) Part 1 – Unreal Engine 4 Tutorial. Available at: https://www.youtube.com/watch?v=h_Cr_azdsDE [Accessed: 13 April 2022]
- Berbece, N. and Paun, P. (2021). Xelu's Free Controllers & Keyboard Prompts. Available at: https://thoseawesomeguys.com/prompts/ [Accessed: 04 May 2022]
- Blizzard Entertainment. (2016). Overwatch. [Video game]. Blizzard Entertainment
- CD Projekt Red. (2015). The Witcher 3: Wild Hunt. [Video game]. CD Projekt
- Clements, J. (2002). The Myth of Edge-On-Edge Parrying in Medieval Swordplay. Available at: https://www.thearma.org/essays/edgemyth.htm [Accessed 25 March 2022]
- Cloy Toons. (2015). Animation: Body Mechanics-Jump. Available at: https://cloytoons.wordpress.com/category/creative-strategies/animation/ [Accessed: 24 March 2022]
- Crunchbase. (no date). Mixamo. Available at: https://www.crunchbase.com/organization/mixamo [Accessed 21 May 2022]
- Dimensions. (2021). Climbing Available at: https://www.dimensions.com/collection/climbing [Accessed: 25 March 2022]
- Dimensions. (2021). Crouching – Men (Front). Available at: https://www.dimensions.com/element/crouching-men-front [Accessed: 24 March 2022]
- enziop. (2021). mixamo_converter. Available at: https://github.com/enziop/mixamo_converter [Accessed: 28 April 2022]
- Epic Games. (2012). Unreal Engine Style Guide. Available at: https://cdn2.unrealengine.com/BrandingMedia/Documents/UE4Styleguide-455099555.pdf [Accessed: 17 February 2022]
- Epic Games. (2018). Paragon: Twinblast. Available at: https://www.unrealengine.com/marketplace/en-US/product/paragon-twinblast [Accessed: 11 April 2022]

REFERENCES

- FromSoftware. (2011). Dark Souls. [Video game]. Namco Bandai Games
- Fullerton, T. (2019). Game design workshop : a playcentric approach to creative innovative games. 4th edn. Boca Raton, FL: CRC Press.
- Game Maker's Toolkit. (2015). Secrets of Game Feel and Juice. Available at: https://www.youtube.com/watch?v=216_5nu4aVQ [Accessed: 19 March 2022]
- Game Maker's Toolkit. (2018). What Makes a Good Combat System? Available at: https://www.youtube.com/watch?v=8X4fx-YncqA [Accessed: 19 March 2022]
- GitHub. (no date). GitHub Logos and Usage. Available at: https://github.com/logos [Accessed: 20 May 2022]
- Hammerhead - Animations. (2018). Rolls and Dodges Animation Set. Available at: https://www.unrealengine.com/marketplace/en-US/product/rolls-and-dodges-animation-set/?sessionInvalidated=true [Accessed: 24 March 2022]
- Hungercalling. (no date). Sword Combo Animation. Available at: https://www.kindpng.com/imgv/owmhwx_sword-combo-animation-hd-png-download/ [Accessed: 25 March 2022]
- Kharitonov, V. (2022). Thermal Katana. Available at: https://sketchfab.com/3d-models/thermal-katana-fb401eace2f447f8a25ea7527e0833bf [Accessed: 18 March 2022]
- Lantz, P. (no date). Applying IK Constraints in UE4. Available at: https://peterlantz3d.wordpress.com/home-2/problem-solving/applying-ik-constraints-in-ue4/ [Accessed: 20 May 2022]
- Lemarchand, R. (2021). Playful production process : for game designers (and everyone). Cambridge, MA: MIT Press

REFERENCES

- McDonald, J. (2021). Taking an Axe to God of War Gameplay. Available at: https://www.youtube.com/watch?v=kX8Jn3XPoWQ [Accessed: 18 March 2022]
- MCV. (2017). Vaulting and ballistics update comes to PUBG test servers – does it make the game better for players and viewers? Available at: https://www.mcvuk.com/esports/vaulting-and-ballistics-update-comes-to-pubg-test-servers-does-it-make-the-game-better-for-players-and-viewers/ [Accessed: 18 March 2022]
- Norman, D.A. (2013). The design of everyday things. Rev. and expanded ed. London: MIT Press.
- Pav, A. (2010). Parkour – Speed Vault. Available at: https://www.deviantart.com/alexpav/art/Parkour-Speed-Vault-162005554 [Accessed: 25 March 2022]
- Phoenix Labs. (2019). Dauntless. [Video game]. Phoenix Labs
- Santa Monica Studio. (2018). God of War. [Video game]. Sony Interactive Entertainment
- Schell, J. (2020). The art of game design : a book of lenses. 3rd edn. Boca Raton, FL: CRC Press.
- Sheth, M. (2019). Evolving Combat in 'God of War' for a New Perspective. Available at: https://www.youtube.com/watch?v=hE5tWF-Ou2k [Accessed: 17 March 2022]
- Ubisoft Montreal. (2013). Assassin's Creed IV: Black Flag. [Video game]. Ubisoft
- Uisco. (2020). How To Make Parkour Vault System In Unreal Engine 4. Available at: https://www.youtube.com/watch?v=zEUNS3eQ-ZM [Accessed: 13 April 2022]

REFLECTIVE STATEMENT

Overall, I believe my project was a success as I was able to deliver the more than my planned minimum viable product. I analysed games that inspired the project, and conducted research to learn more about design and production. This was used to influence my designs, development and iteration which ultimately led to my final prototype.

The research and analysis I engaged with was very valuable in leading my designs and helping understand what can work and what to avoid in design. This was useful for my project specifically as melee combat can have huge differences between games. I believe I was able to design my melee combat well due to the analysis of mechanics and themes already present in many games. I could have more accurately followed some research, however when I fell behind schedule I found it difficult to incorporate every theme I wanted to from my research.

My original design work was not my strongest work as I had fell behind schedule at this stage due to a large workload in DES310. I did not use any paper prototyping and had minimal iterations in my designs which would have likely helped improve the final prototype. If I retook the project, I would improve my management skills further by creating a weekly work schedule that considered both modules to ensure I spent an appropriate amount of time on both projects as it was easy to spend too long on DES310.

REFLECTIVE STATEMENT

As a whole, my development stage was a success. I split this into 8 “phases” across weeks 9 > 14 with an agile workflow where I could iterate on mechanics as I went along. This allowed me to adjust the project as needed as I came across limitations or improvements whiles implementing features. I completed three playtests in these stages in phases 6, 7 and 8. These were very valuable as they provided an outside look at my project and how it played out. Looking back, holding more playtests or feedback sessions, even in design stages with no prototypes to test out would have improved the overall result.

Due to falling behind earlier on, I was unable to implement any sound effects as I had more important features to implement. I believe this would have helped round off the project and improve the game feel providing further feedback to the player based on their actions. I would have also liked to implement more visual effects along with the sword trails to make attacks more impactful.

I am happy with the final prototype overall, I do believe I created too ambitious a scope early on and should have taken a safer approach where I could expand the project later if I had additional time. I have enjoyed learning more about an iterative design process, and working with the Unreal Engine and visual scripting. I believe I have improved my skills like I had aimed to do at the start of this project. In the future I would like to continue this project and investigate more unique mechanics now that I am more comfortable working in Unreal Engine.