

ARCHELIGHT

DESIGN & DEVELOPMENT REPORT

TEAM: HOW COULD THIS GO WRONG
BOURNEMOUTH UNIVERSITY FACULTY OF SCIENCE AND TECHNOLOGY 2021 LEVEL 5

Contents

Game Design Document.....	4
High Concept.....	4
Game Overview.....	4
Key Features.....	4
Player Motivation.....	4
Game Flow Summary	4
Plot and Setting.....	5
Background	5
Story.....	6
Story - Implementation.....	6
Character Visual Design	6
Asset Design.....	12
Environment Design	18
Lighting.....	20
Volumetric Lighting.....	20
Toon Shading	20
Particle Systems	23
Gameplay	26
Objectives	26
Challenge Structure	26
Upgrades.....	27
Combat	27
Puzzles	27
Music and Sound Effects.....	27
Mechanics	28
Physics.....	28
Player Character	28
Gun mechanics.....	28
Bullet Projectiles	28
Objects	29
Randomly Generated World.....	29
Interface.....	30
Head-up Display (HUD)	30
UI Elements:.....	30

Artificial Intelligence	31
Enemy AI	31
Skull Enemy (Close quarters)	31
Ranged enemy / Crystal enemy (Long and medium quarters)	31
NavMeshAgent	32
NavMesh / baking	32
Menus	33
Controls	33
Implementation for the Engine	34
Game Testing	35
Development Report	35
Management – Team Roles	35
Management – Communication	36
Management - Team Working Principles	37
Meetings	37
Planning	37
Design Implementation Techniques	38
Progress Tracking	38
Minutes Log	39

Game Design Document

High Concept

Archelight is a 3rd person action rogue-lite where the player battles through the ancient ruins of a civilisation in pursuit of knowledge and the rare mineral archelight.

Game Overview

- Game Name: Archelight
- Target Audience
 - The team has chosen teenagers with interests in 3rd person shooters and roguelikes. This is because the difficulty is a bit too high for people younger than that to figure out as well as the skulls and violence of the game. This would also appeal to people who like playing other rogue-like games, and third person games are a very common genre which means we are not limiting our target audience too much.
- Genre
 - Our game is a variety of a third-person shooter, a rogue-like and an RPG in one, since the game is third person and the main combat mechanic is shooting enemies before they get to you. Also with the pick ups that spawn around the map as well as the permanent upgrades that you can get throughout playing the game.
- Platforms
 - We are making this game for PC's since that is where the bigger player base is as well as the control scheme is made for the PC.

Key Features

- 3rd person action gameplay involving shooter elements
- Gameplay focused – story will be delivered through in-game collectibles similar to 'DOOM' or 'Risk of Rain 2'
- Rogue-lite gameplay
 - Game will feature permanent progression and progression within each playthrough.
 - Each playthrough the game will be different. Different enemies, different upgrade and different environments.

Player Motivation

Each playthrough should be unique, therefore the core gameplay loop must be enjoyable. The players goal is to survive within the ruins for as long as possible or until they escape. Some upgrades will persist between playthroughs to keep the player engaged for more than a single playthrough. Another motivation to continue playing would be finding collectibles to discover more of the game lore.

Game Flow Summary

This section will detail the process a player will go through when playing the game.

- Player loads game and is greeted with the main menu, the player will choose from:
 - Play Game
 - Collectibles
 - Options
 - Exit
- ✦ Exit is self-explanatory

- ✦ Options will be the usual sound, video and control settings.
- ✦ Collectibles will have a list of permanent items or lore the player has found throughout their time playing the game.
- ✦ Play game will take the player into a playthrough of the game, or their most recently started playthrough if they already began one

The gameplay loop will involve moving around the obstacles and enemies within the rooms to avoid taking damage while taking out the enemies with the players weapon, the revolver.

Once a room is cleared of enemies the player will be given a tunnel too a new room and a reward with a temporary character upgrade. Some examples of upgrades would be a larger ammunition reserve for the revolver, or an additional dash for the player.

After a certain amount of the rooms the player will encounter a boss enemy with increased health and lethality compared to regular enemies. Bosses will also feature new mechanics altered from the regular enemies which the player must adapt to.

Defeating this boss will lead the player to the next 'floor.' Each floor will increase in difficulty compared to the previous and be set in a new environment, and with a selection of new enemies.

The player will go through this loop until dying or completing the game.

Permanent upgrades will be acquirable throughout these playthroughs which will affect future playthroughs.

Lore and story elements will also be permanent collectibles which can be accessed in the main menu.

Plot and Setting

Background

The main character of the story is an archaeologist/adventurer who has taken on a job to excavate areas with a newly discovered crystal due to its unique properties as a very dense fuel source. The crystal has been colloquially called 'archelight' due to its bright appearance. During the excavation of a site an ancient civilisation lost to time was uncovered.

Exploring the ruins further leads the exploration team deeper underground, to a city which was still recognisable, almost as if a modern-day city were to be deserted of any life. It became immediately apparent that archelight had been used in this surprisingly advanced civilisation as fuel and decoration, the underground civilisation's ruins were abundant with archelight. Seeing all this begged the question, if a city so well preserved was devoid of life as if everyone simply vanished, what caused it to be buried underground and lost to time?

In the distance there seemed to be some sort of creature moving and with curiosity getting the best of the main character, they decide to get closer to try and get a better look at it. Upon closer inspection the creature had a very distinct exoskeleton, humanoid proportions and the skeleton had portions replaced with archelight crystal. The excavation team caught up with the main character and carefully they approached the creature. The team got within a few meters of the creature and initially the creature did not react to the presence of the team.

Suddenly without warning, the creatures body crumpled to a heap of bones, with seemingly no signs of them being adjoined mere moments ago. The team also noticed that the archelight lining the

streets was pulsing like a heartbeat as if it were alive. Thoroughly scared, the team decided to make a swift exit out of the ruins only to find the way which they entered was completely covered in archelight. Now skeletal remains were being reanimated and infused with archelight, but this time they absolutely recognised the group and began to move towards them menacingly.

Story

You as the main character are tasked with fighting off these archelight infused corpse remains and finding a way out of this ruined civilisation. Along the way the player will uncover the truth behind the mineral 'archelight' and the mysteries surrounding the ancient civilisation.

As the player progresses through the game, they will learn that archelight is not just a simple mineral, it is in fact a hivemind which can re-animate creature remains. The archelight did not consent to having itself be excavated and used as fuel so it must fend off the player.

The player will learn that long ago the civilisation which lived in these ruins had an interesting relationship with archelight, treating it as a god. The archelight seeks only to further its spread and assimilate living beings into the hivemind, which lead to the downfall of the civilisation of religious fanatics.

Later in the game the player will find still living breathing people, which had fully assimilated with the archelight, having the advantages of a human body but still being subjected to the will of the archelight. The player may face the moral dilemma of having to fight someone who looks similar to them but lives only to serve the archelight. Furthermore, they will question whether it was fair or right for them to mine and use the Archelight as fuel when the civilisation will cease to exist if the hivemind is destroyed.

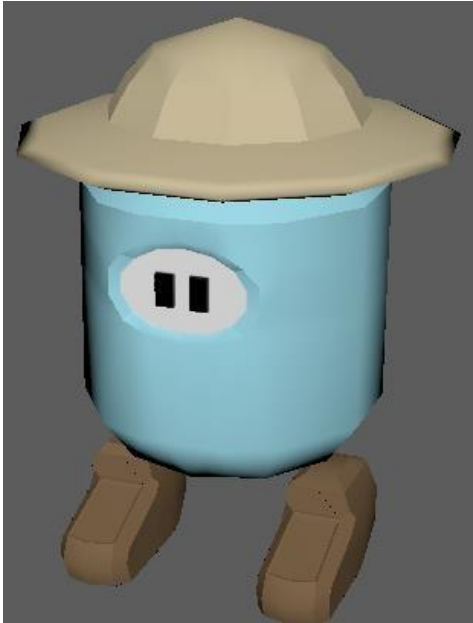
Story - Implementation

The story of the game will be delivered through collectible items which will be hidden like eastereggs throughout the levels. The player will be able to access these through the collectibles option in the main menu.

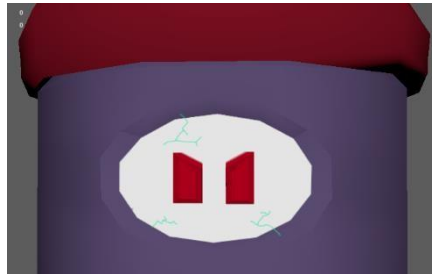
While the story is not the focal point of the game, the team still decided it was important to have a story for the game world. Games like 'Risk of Rain 2' employ a similar story telling method where the player can simply choose to enjoy the gameplay and pay no attention to the story, but players who become invested in the story will still have not the content they are looking for. The advantage of using this story telling method is that gameplay sessions can be uninterrupted for players who do not want to be sitting through cutscene after cutscene while story-driven gamers are still catered to.

Character Visual Design

Name	Image	Description

Player		<p>The protagonist of Archelight. We opted to use a very simple design, while still showing exactly what the player is - hiking boots and a pith hat make it clear that the player is some kind of explorer/archeologist. Armed with a pistol and a torch, they dispatch the beings within the ruins whilst gaining new abilities that aid further exploration.</p> <p>The player is pale blue in colour since it is pale and light - it makes it clear that we are the protagonist. The capsule shape used for “human” characters was inspired by games such as Among Us and Fall Guys, to make the characters memorable.</p>
--------	--	--

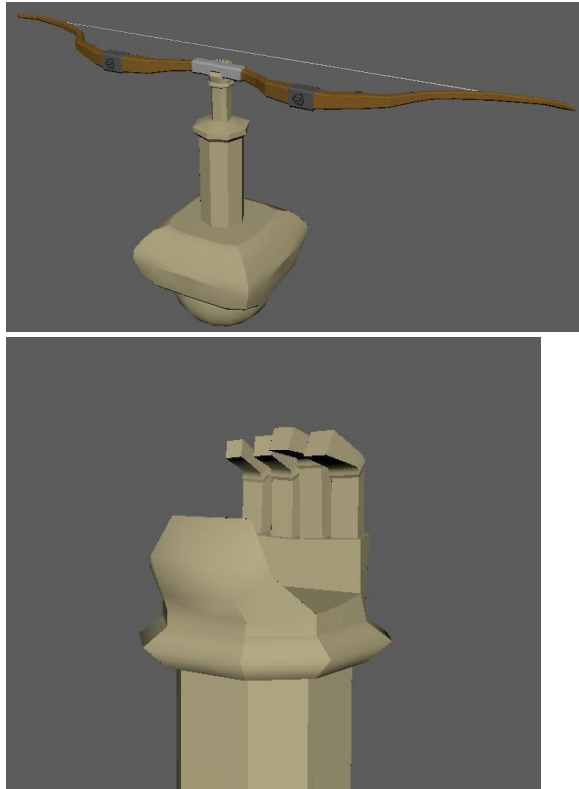
Archer



The enemy archers found stalking the open caves alongside the ghosts and automatons. With an angry red stare, and green cracks in their face, it could be inferred that they were once bandits or explorers similar to the player, who succumbed to the power of Archelight. Armed with a bow, they fire rapid shots that can prove deadly in great numbers.

The red bandanna implies that this enemy was a bandit, whilst the purple colouring suggests that they are walking corpses - perhaps already rotting.

Bow Roller



May seem like an inanimate ornament but is in fact an automaton that defends the caves and ruins. Powered by Archelight, it launches spectral arrows at greater distances than the archers. Slow and heavy, with an eerie hand shape gripping the bow. Perhaps slow due to minimal amounts of Archelight flowing into it. They are manufactured with the same stone material that makes up the columns and slabs throughout the ruins.

Claw Roller



The purple claw manipulates raw Archelight energy, enabling it to move around and expel multiple projectiles in a short range. Very powerful if they get close, so caution must be exercised when dispatching these automatons.


Complex technology beyond normal understanding powers it.

The claw is mounted on a roller, similar to the bow roller. The material used for the claw itself is unknown - a purple, sharp crystal.

Reanimated
Skull




Speedy projectile-like enemies that fly head first into their targets. They explode on impact and do a lot of damage, so the player will likely spend time running and gunning in order to dispatch these targets. When moving around, a purple trail is emitted from within the skull, implying some extra force at play manipulating it.

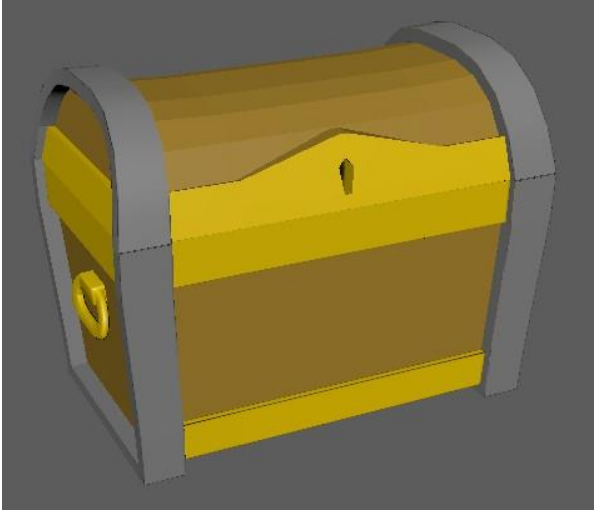
Arch Skull		<p>An amalgamation of bones held together by the will of Archelight, it charges relentlessly and releases additional skulls on impact, making it extremely deadly. The miniboss of the game. It is much larger than the normal Reanimated Skulls, and bleeds red particles from glowing red eyes, implying that it might be sentient. Very difficult to fight without collecting upgrades from earlier levels, it is the final enemy in our game demo.</p>
------------	---	--


Asset Design


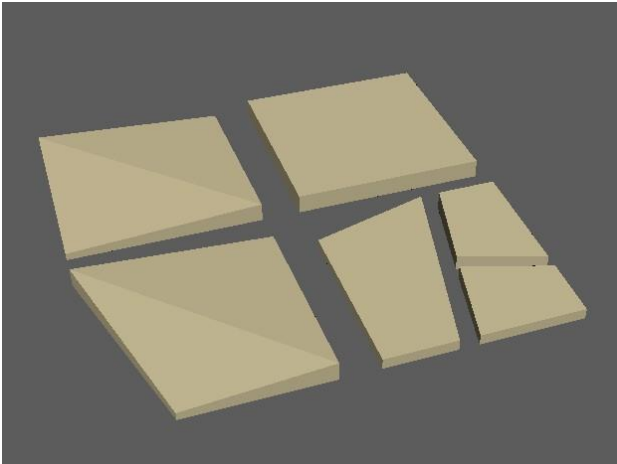

Name	Image	Description

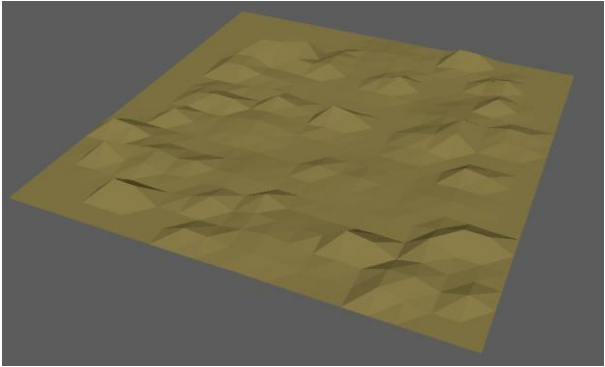
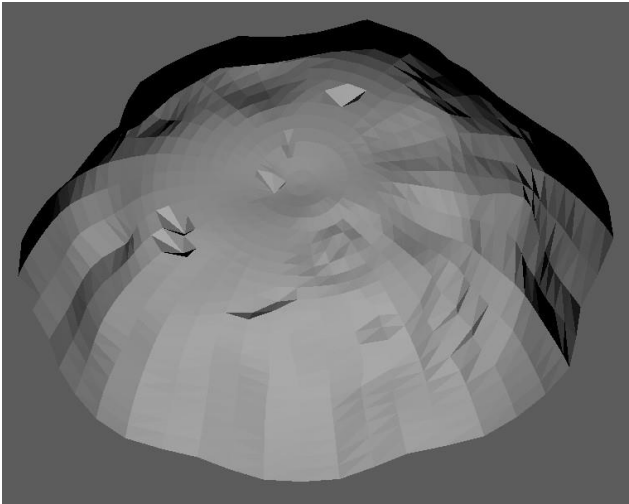
<p>Stalagmite</p>		<p>Present in the cave, they add to the aesthetic of a cave undisturbed for a great amount of time.</p>
-------------------	---	---


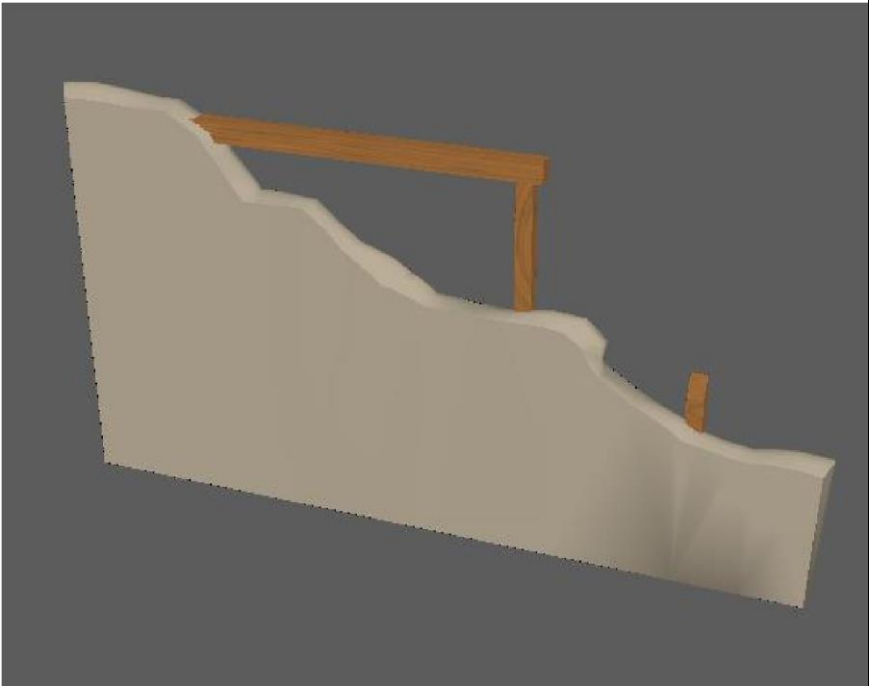
<p>Torch</p>		<p>The torch lights up the game's environment, as well as giving the player visibility in the cave and ruins. Large and durable, it can be found in some locations within the ruins.</p>
--------------	---	--


Chest		<p>Chests are placed within the game's level after each stage has been cleared. When opened, an upgrade will be presented to the player. The gold part of the chest stands out to the player and implies that it should be approached.</p>
-------	--	--

Column		<p>Remnants of the ancient civilisation, they take inspiration from ancient Greek designs. They stand tall randomly in the level.</p>
--------	---	---

Arrow		Both the Archer and Bow Roller enemies fire the arrow models to inflict damage. The arrow is barbed for maximum damage.
Slabs		Old floor slabs and tiles that have cracked over many years.
Rock		Fallen rock from around the cave ceiling. Can be used as a platform and used as cover.

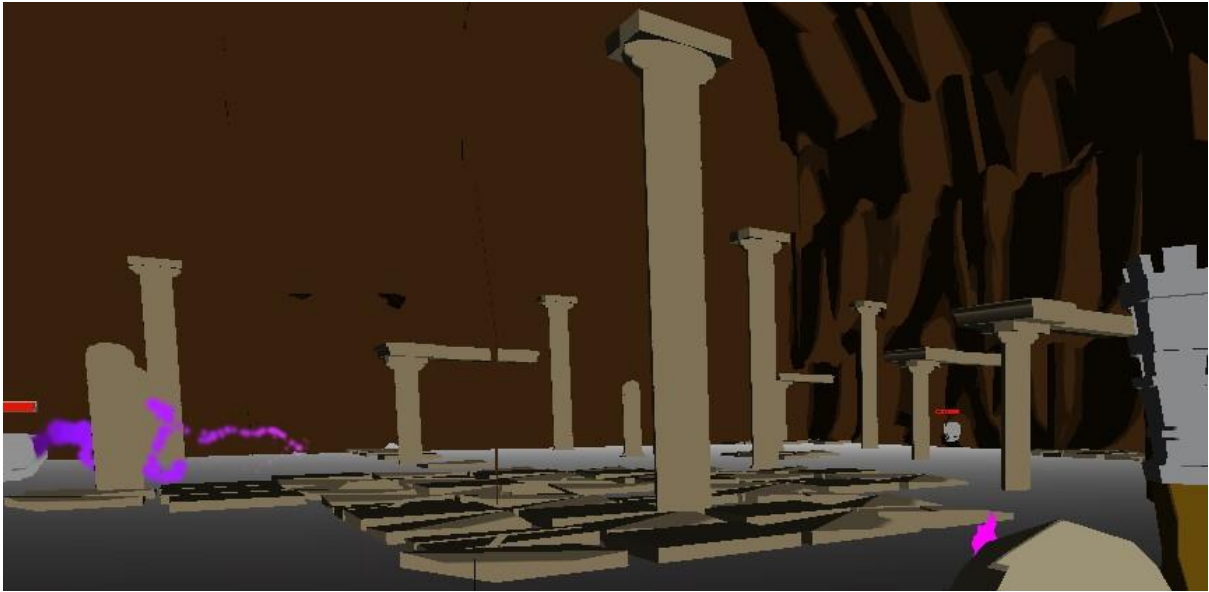
<p>Cave Ground</p>		<p>The uneven terrain on the cave floor. The player can use the bumps on ground as cover and to gain distance from land enemies.</p>
<p>Cave Shell</p>		<p>The outer shell of the cave. Large and open.</p>

Bow		<p>The bow wielded by enemies found within the cave and lost ruins. Seems to be made of wood, but somehow survived over hundreds of years. Has metal braces with an unknown sigil, and a metallic handgrip.</p>
Broken Wall		<p>The frame found inside the wall is exposed. The construction seems more primitive than some of the machinery present - maybe this was built more recently.</p>

Ruins		<p>Crumbled parts of the buildings and columns. They demonstrate the age of the structures within the ruins, especially when compared to the starting structure.</p>
-------	--	--

Environment Design





For the environment design, we did not stick to a specific historical period. But one of inspirations was the Ancient Greeks, as shown in the images above with the standing columns.



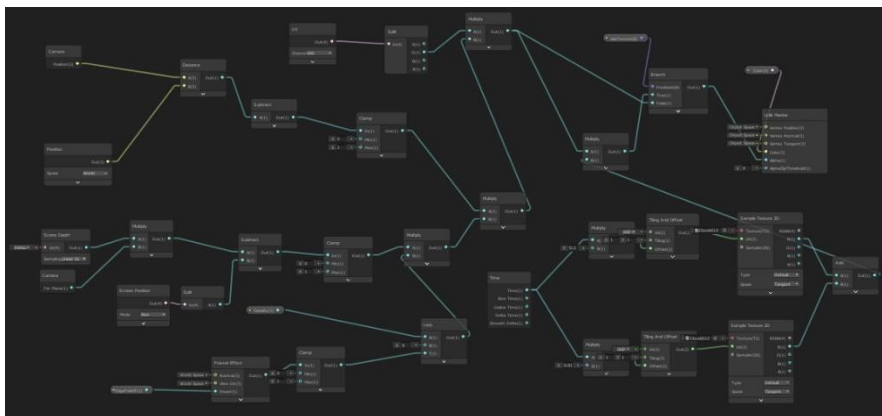
To add more realism to the cave, we added stalagmites and stalactites. This implies to the player that there has not been hardly any human presence within the cave in a very long time - it takes thousands of years for these to form.

Lighting

Volumetric Lighting



[Example: Volumetric Lighting]



[Graph Editor Image]

Since the game is set inside of a cave, the game would rely on light sources, one of them being light rays from the crack in the ceiling. The light beam was made from a cylinder with a material with a custom shader attached to it. The shader was made by using Unity's Graph Editor and with this, it allowed us to program the opacity, intensity, edge falloff and allowed the textures to move.

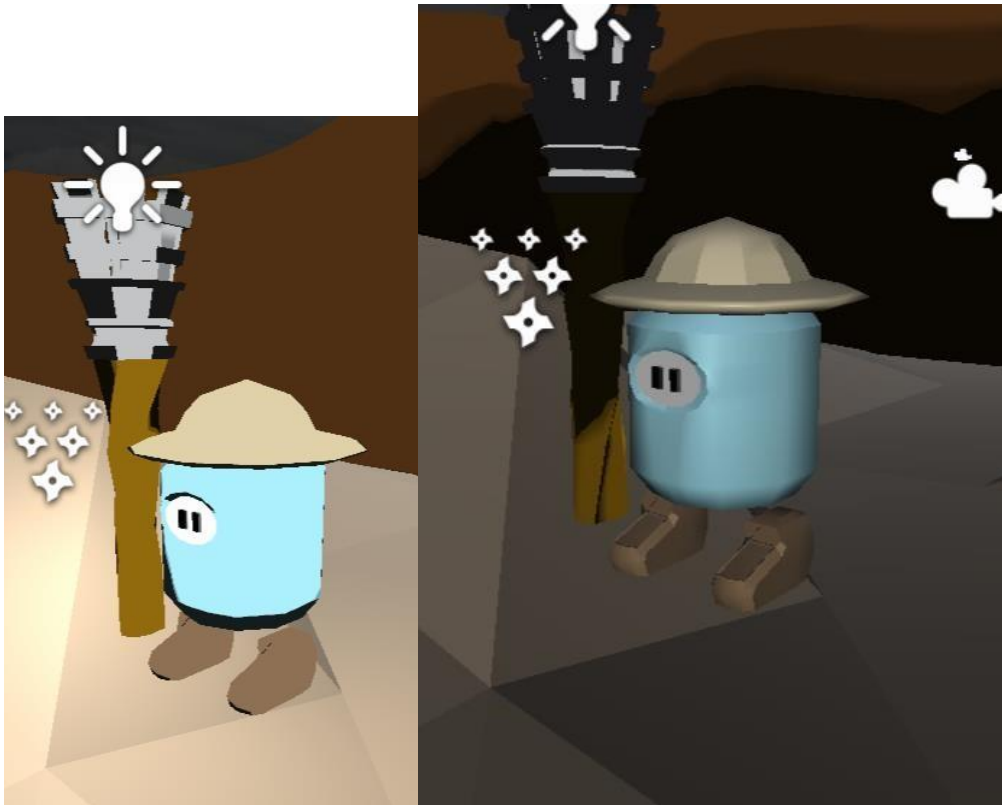
Toon Shading

To implement the cartoon-like cel shading into our game, we had to code a shader so that it would receive light from a directional source, and have specular reflections and rim lighting.

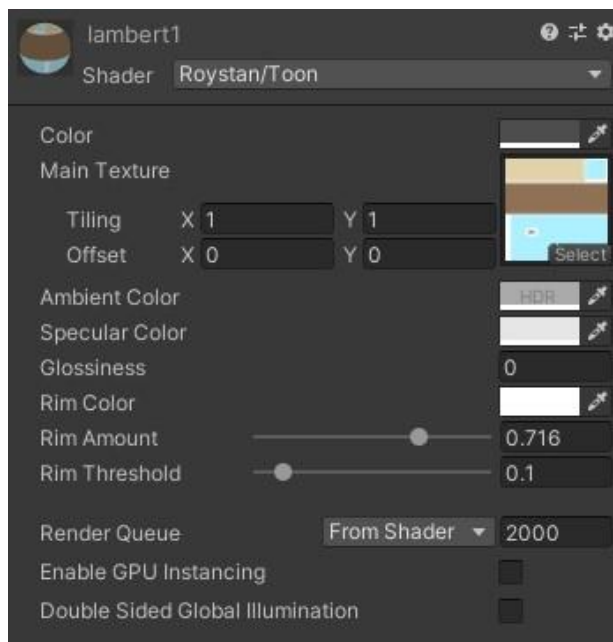
The shader calculates:

- The amount of light hitting the surface of a given object
- Where the directional light is coming from
- A set ambient light that affects all surfaces equally

- The direction from which a given object is being viewed
- Where to cast shadows based on existing values within Unity



The resulting lighting for the depicted player model is a lot sharper. When applying the shader to a material, we can change the 'glossiness' value of it to affect the specular lighting.

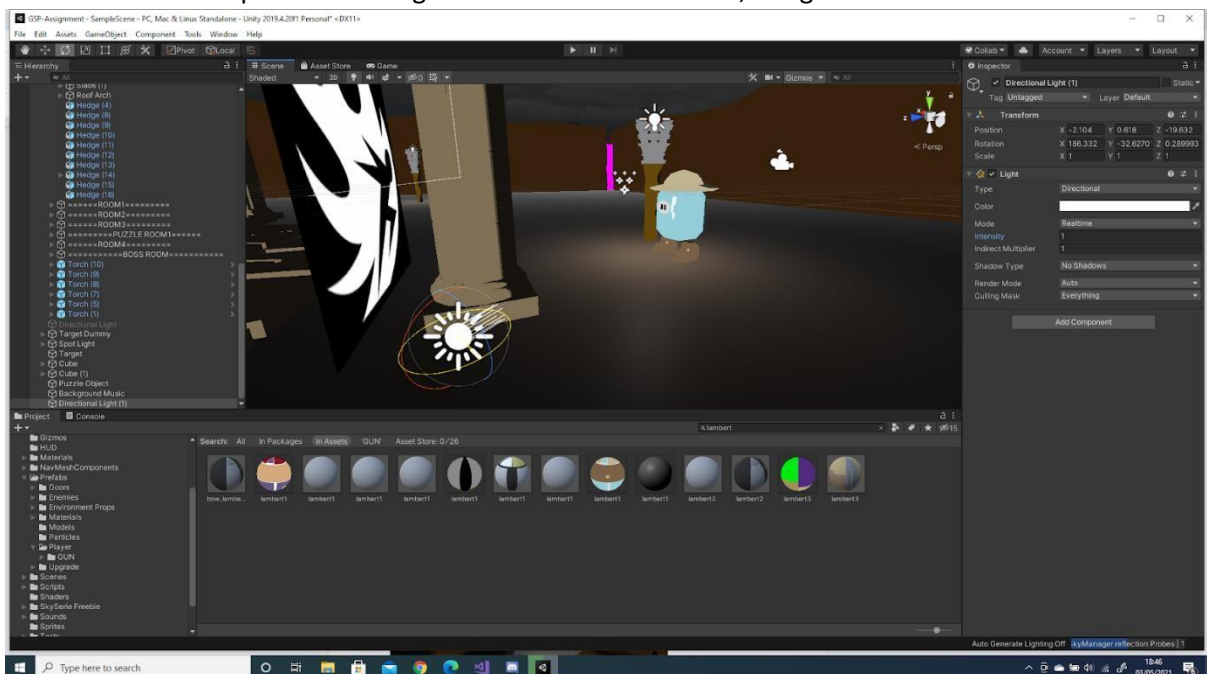


```

_AmbientColor("Ambient Color", Color) = (0.4,0.4,0.4,1)
_SpecularColor("Specular Color", Color) = (0.9,0.9,0.9,1)
_Glossiness("Glossiness", Float) = 32
_RimColor("Rim Color", Color) = (1,1,1,1)
_RimAmount("Rim Amount", Range(0, 1)) = 0.716
_RimThreshold("Rim Threshold", Range(0, 1)) = 0.1

```

Other variables found in the shader can also be modified, such as the specular colour, rim colour, and other values depicted in the figure above. In this instance, the glossiness is set to 0.

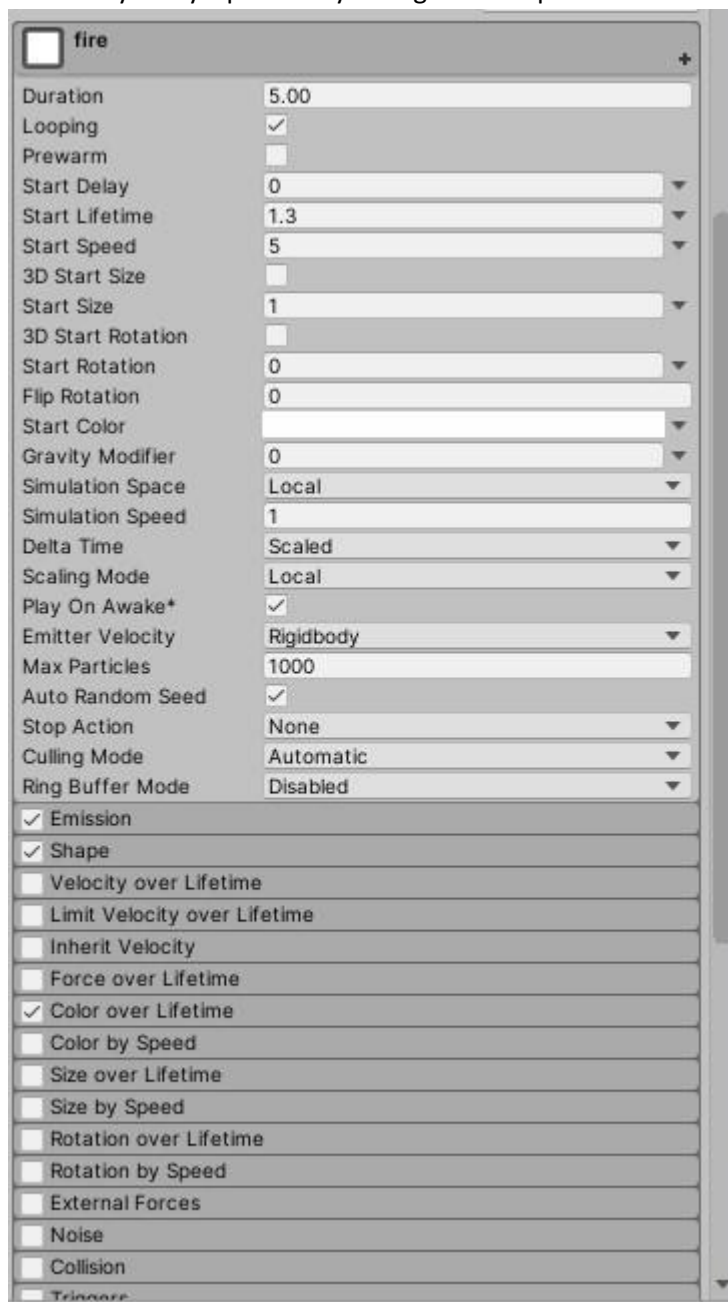


Increasing the glossiness adds a reflectiveness to the material, visible in the side of the players face and shoes in this instance.

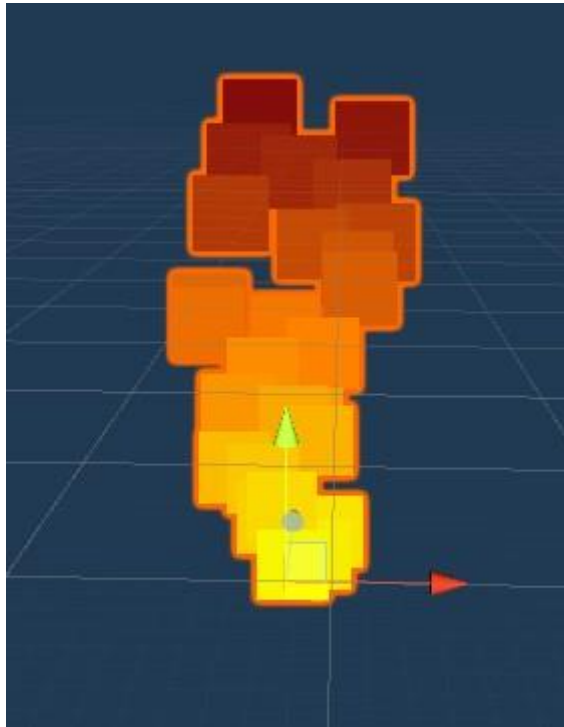
As referenced previously, we used flat coloured textures - the cel shading works best with minimally detailed textures and materials.

Particle Systems

All of the particle systems have a voxel/cartoon theme; particle systems add detail to our models and game world as well as special effects for certain actions in the game. Each particle system was created by Unity's particle system game component



There are many options for creating a particle system meaning that we could create many particle systems for very specific things.



The fire particle system is used for the torch model along with the lighting game object, the particle system adds more detail to the model as well as show that there is a fire entity coming out of the torch.



The Arrow particle system is meant to make the arrow projectile clearer for the player as it is choreographed by the particle system. This makes it so that the player can dodge the arrows and not accidentally run into an arrow due to unclearness.




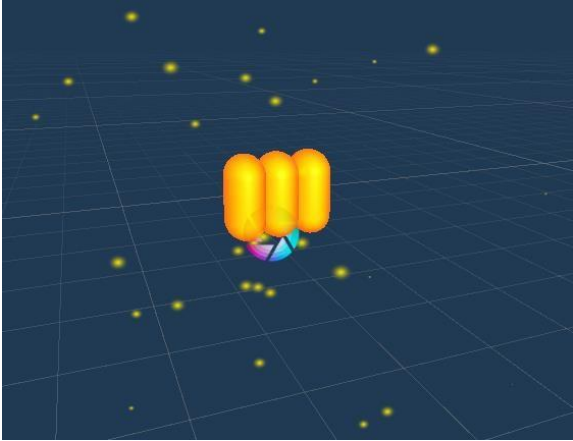
The skull particle system is used for the same reason as the arrow. It is also used for aesthetic reasons as the particle system makes the skull more sinister.



The particle effect on the boss at the end also is meant for aesthetic reasons as it makes the boss appear evil and scary.



This particle effect both uses the arrow and skull enemy particle effect to create a very unique trail that is meant to simulate some sort of dark magical power coming from the orb.

	<p>This particle effect is meant to simulate a muzzle flash of a gun going off. It is a very simple particle effect and only appears for a very short amount of time. There is a light on the particle effect which lights up a small amount of the area.</p>
	<p>Each upgrade in the game has a particle effect, this particle effect spins around with the upgrade and is meant to signify to the player that it is a powerup.</p>

Gameplay

Objectives

In Archelight, there are both main objectives and side objectives that the player can optionally do. The core objective is to survive in each room and move onto the next until they've reached the boss room in which they need to defeat the boss. However doing the core objective on it's own can be challenging hence the side objectives are there to help aid the player throughout the game. These side objectives are the puzzles and exploration to find extra upgrades for their character.

Challenge Structure

The overall challenge structure is ultimately down to the randomness of the game as all the enemies and upgrades are randomly generated. This means that there is no definite difficulty - this means that the generator can favour or disfavour the player. They could get a room filled with enemies or a room with very little enemies or get a good run of upgrades. However, the player can do the side objectives to tone down the challenge by obtaining more upgrades.

Upgrades

There are a number of different upgrades the player can obtain throughout the playthrough some give them stat boost and some give a completely unique ability. Upgrades such as fire rate, reload time give the player more damage per second whilst movement upgrades such as dash cooldown decrease or double jumping gives the player more mobility and allows them to dodge enemies and traverse the world easier.

Upgrade	Buff/debuff	Type
Dash Upgrade	Reduced the dash cooldown by 1 second	Utility
Double Jump Upgrade	The player can now double jump	Utility
Fire Rate upgrade	The player can shoot faster by 0.22 seconds	Damage Per Second Boost
Reload Upgrade	The player reloads faster by 1 second.	Damage Per Second Boost
Reload max capacity upgrade	The player can now shoot more bullets by two before reloading	Damage Per Second Boost

Combat

Combat is the main interaction and gameplay of Archelight, we wanted to make the combat as entertaining and rewarding as possible and we wanted the upgrade system to make character builds and runs fun. This does mean having a character strong enough to beat the game very easily, many players who play the roguelike genre enjoy building an overpowered character to defeat the game.

The only limited resource in our game is health, we did not want to include an ammunition resource as that would take away from the face pace gunplay and make another limitation the player has to think about. We did this as we wanted to make combat the main gameplay and did not want resource management to be a factor.

Puzzles

Puzzles are a secondary objective in Archelight, they give the player another chance to get more upgrades for their character. There will be a couple of puzzles on the map as too many puzzles would give too many upgrades to the player. We want to maintain a healthy amount of upgrades.

Puzzles will be easy to implement within our game world, we can quite easily put in a jump puzzle or a little puzzle for the player to solve.

Music and Sound Effects

There is going to be music in our game that will be in the background while the player is playing the game. This will not be too loud so that the player cannot hear the game sound FX's. And then in the

menus there will be a setting if the player wants to change then. The music could also then change depending on the level that the player is on to give more immersion to the player experience.

Mechanics

Physics

Many of the gameobject in Archelight require physics such as the revolver, player, the puzzles and bullet projectiles.

Player Character

We used the familiar control layout of WASD to move left, right, forwards and backwards, as well as a jump which is executed with the spacebar and a dash which is executed with the shift key.

Taking advantage of Unity's GameObject transform and rigidbody components, we can move the player character using physics. We manipulate force in a direction to move the character's rigidbody; this is how the playermodel jumps and dashes. For the movement of running we moved the transform of the player to increase or decrease values; each value representing forward, backwards, left and right.

Gun mechanics

To create the gun mechanics, we used Unity's Physics Raycast which casts a ray at a point of origin, the point of origin is whatever the camera is looking at and the raycast returns a bool if it interacts with any colliders. We set it so that it returns a bool if it hits an enemy and if it does then we get the enemy's script component and call a function to damage the enemy.

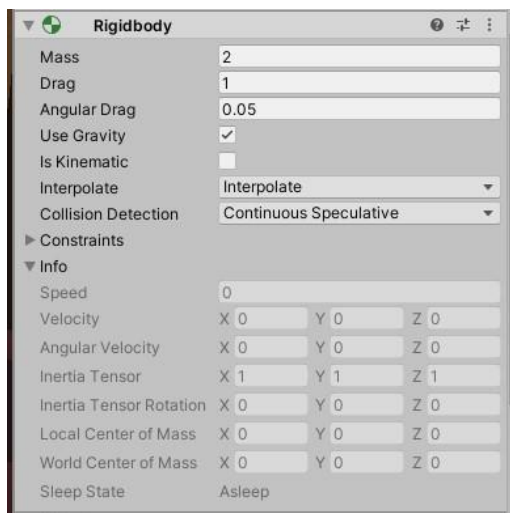
```
RaycastHit hit;
if (Physics.Raycast(cam.transform.position, cam.transform.forward, out hit, range, layerMask))
{
    //Debug.Log(hit.transform.name);

    Enemy target = hit.transform.GetComponent<Enemy>();
```

Parameters: The cam.transform.position is the starting point of the ray which is the camera position, cam.transform.forward is the direction the ray is going, out hit is what the raycast hits, range is the max distance the ray goes and the layer mask is a integer represent a layer in Unity which the raycast ignores. We use this for invisible collider triggers.

Bullet Projectiles

Many of the enemies in Archelight use bullet projectile physics. To achieve this, we used Unity's built in physics and rigidbody components and found the values to simulate bullet projectiles in our game with gravity and mass.



```
if(!alreadyAttacked)
{
    AudioSource.PlayClipAtPoint(arrowSFX, transform.position);
    Rigidbody rb = Instantiate(projectile, bulletSpawn.transform.position, Quaternion.identity).GetComponent<Rigidbody>();
    rb.AddForce(transform.forward * 34f, ForceMode.Impulse);
    rb.AddForce(transform.up * 8f, ForceMode.Impulse);

    alreadyAttacked = true;
    Invoke(nameof(ResetAttack), attackSpeed);
}
```

(example code and rigidbody values)

Objects

The world has three different types of gameobjects: static, dynamic, and invisible triggers.

Static game objects are used for when the player does not need to dynamically interact with the player. Example of these game objects are but not limited to: floor, walls, props and other parts of the environment

Dynamic or non-static objects are game objects that move. A lot of the enemies in the game require some sort of movement meaning that they need to be non-static otherwise they will be stuck on the spot, the movement creates difficulty as the enemies can move and dodge player attacks. A lot of other non-static objects are: bullets, puzzles, game objects with animation such as a chest, etc.

Invisible triggers are invisible gameobjects with a collider and if another gameobject touches them then something will happen. An example of this is in Archelight, as soon as a player enters a room they touch a collider trigger and this sets a timer for the next door to open.

Randomly Generated World

Each room in Archelight apart from the boss room is randomly generated using a simple random generator script created by us. We made the random generated by creating a 7x7 grid of spawn points in the room and the spawn points have a script which randomly selects a number from a list of game objects and if it lands on that game object it will instantiate it. These gameobjects are world props and enemies. Many roguelikes such as The Binding of Issac use a similar system and we took inspiration from that game.

```

Unity Script | 0 references
public class LevelGenerator : MonoBehaviour
{
    public GameObject[] objects;
    public int rand;

    Unity Message | 0 references
    void Start()
    {
        rand = Random.Range(0, objects.Length);
        Instantiate(objects[rand], transform.position, Quaternion.identity);
    }
}

```

Interface

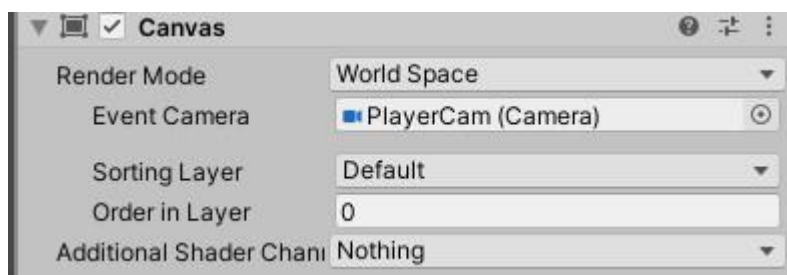
Head-up Display (HUD)

Archelight's gameplay is fast paced as the player is nearly constantly being attacked and has to be prepared with every room. This means that the HUD has to be clean and precise; it has to be easy to read and gather information for the player, this allows the player to put all their concentration on the combat.

UI Elements:

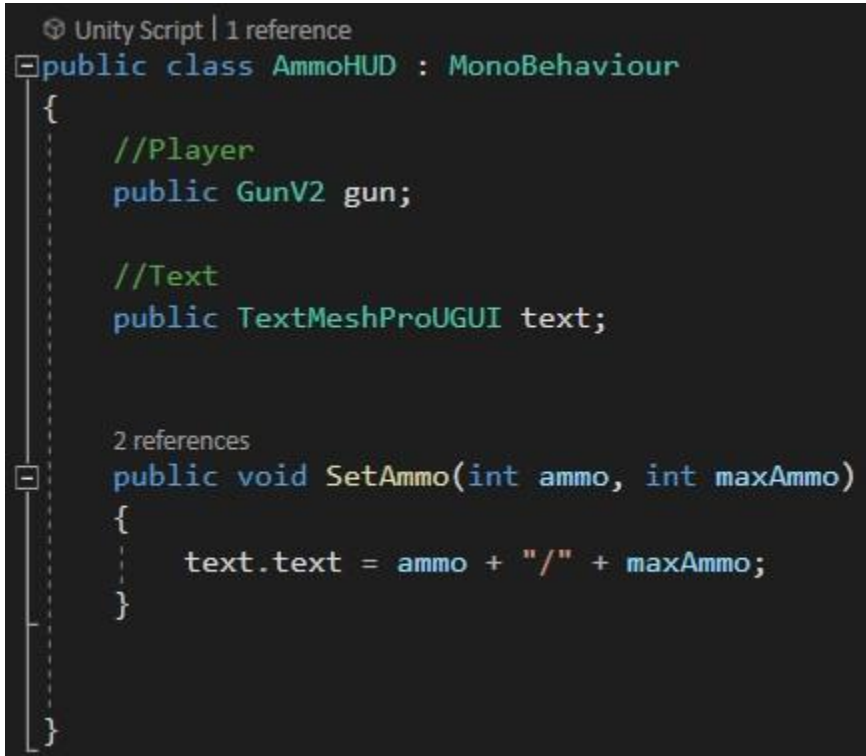
- Crosshair - A crosshair should be simple and not be obstructing anything, this means we need to find a good crosshair which the player can see but not be blocking anything. A dot was chosen as a crosshair as it was easiest to aim whilst promoting clarity.
- Current ammo and max ammo - We need to tell the player how much ammo they have as well as have a dynamic integer stored just in case the player receives an upgrade and we need to update the amount.
- Health Bar - Show accurately the amount of health the player has left.
- Dash Cooldown - Show how long the amount of time it takes to use the dash cooldown again.
- Text - We need text to tell the player what they can interact with and what the upgrade they picked up does.

Using Unity's inbuilt UnityEngine.UI library, we can program all of these UI elements to work to our complete intention. Unity also provides many components that help us create the UI such as Canvas is an orthographic UI element that stores all UI components. But they are not limited to the 2D plane. We can use UI in the world space and give enemies their own unique health bar without them appearing in the orthographic UI meaning they could clutter the UI.



Components we will use:

- Text / TextMeshPro - we use this to display the text for the ammo, health, upgrade text and interact text.
- Image - for sprites, such as the ammo, crosshair and dash cooldown.
- Slider - A simple slider that can be universal, we use this for the health bar.



```

Unity Script | 1 reference
public class AmmoHUD : MonoBehaviour
{
    //Player
    public GunV2 gun;

    //Text
    public TextMeshProUGUI text;

    2 references
    public void SetAmmo(int ammo, int maxAmmo)
    {
        text.text = ammo + "/" + maxAmmo;
    }
}

```

(example code for Ammo)

Artificial Intelligence

Enemy AI

There are two AI enemy types in Archelight, each have their own unique style of attack, movement, and stats. Adding in different types of enemies makes it so that the player is constantly in action, as we have a close quarter enemy and a long-distance enemy.

Skull Enemy (Close quarters)

The skull enemy has two states:

- Patrolling - the skull will randomly choose a walk point and walk to it; this loops until the skull enemy has found the player
- Attack - Once the player is within the skull's sight range, the skull will move towards the player at a fast speed then explodes once it collides with the player. When the skull explodes, a sound effect and a explosion particle is instantiated at the skull's location

Ranged enemy / Crystal enemy (Long and medium quarters)

The ranged enemy has a bow meant to shoot the player from a long distance whilst the player is distracted with the skull enemy, they move slower than the skull and have a slow fire rate to compensate for the range.

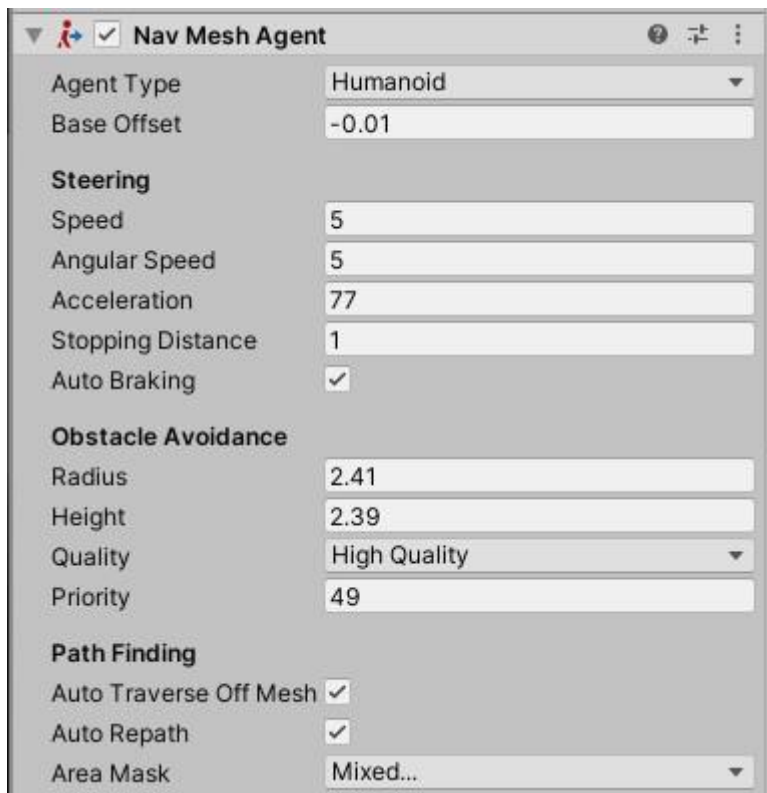
The crystal enemy has a special attack in which spreads similar to a shotgun, they have to get fairly close to the player to hit one of their attacks, if the player runs back the lower chance of the attacks hitting, however getting close to them will damage the player tremendously.

They both share the same script states:

- Patrolling - same as the skull
- Chase - Once the player is in sight range, it will chase the player until the player gets into the enemy's attack range
- Attack - Enemy starts attacking the player, if the player runs away, it goes back to the chase state.

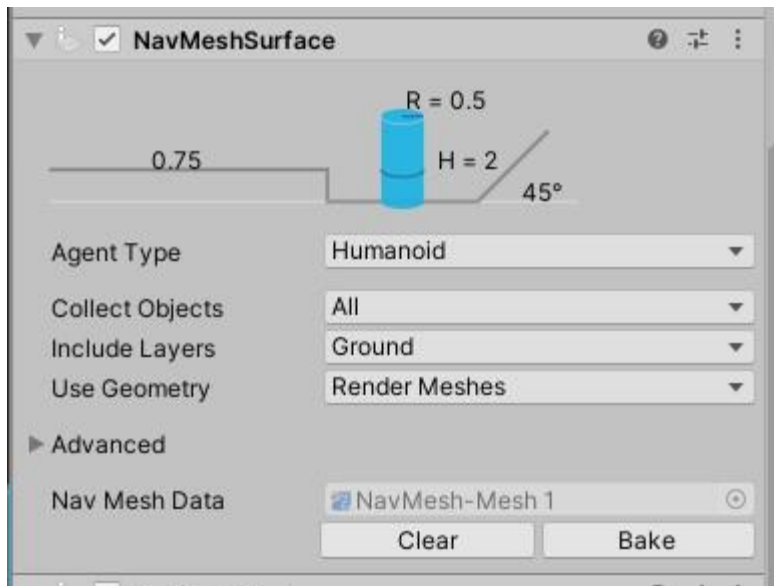
NavMeshAgent

Unity has their own AI components called NavMeshAgent which is essentially a moving character that can navigate a scene using NavMesh. You can adjust the speed, acceleration, stopping distance, etc. As well as the NavMeshAgent's obstacle avoidance and path finding.



NavMesh / baking

In order for the NavMeshAgent to work we need to bake the plane of our game world without it the NavMeshAgent would have no idea how to navigate through the plane.



Menus

The menus for the game will be rather simple since there is not any complicated inventory system, but the team are going to add in a start menu with an additional options menu so that they can toggle the music on and off as well as the other sound settings, then the main menu itself will have a simple play and exit buttons as well. Then once the player is in the game they can press “esc” which is a very common key for a menu and is also close to the controls for the player. And within this menu, they can restart the run as well as go back to the main menu. Also, if we have time and make multiple levels, in the main menu there will also be an option for selecting which map/level they want to play.

Controls

The controls for Archelight use the familiar and recognisable WASD standard format for most first person and third person games. This is so that the player does not have to learn any new controls and just start playing.

Key and action:

W = Forward

S = Backwards

A = Left

D = Right

SPACE = Jump

SHIFT = Dash

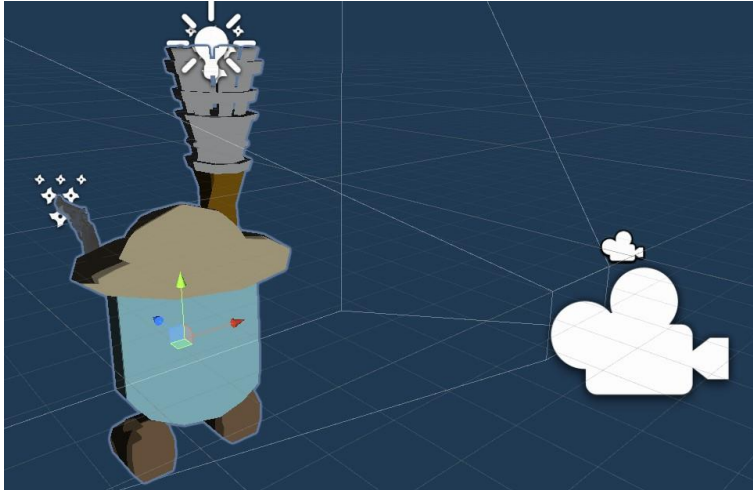
R = Reload

E = Interact

LEFT CLICK = Shoot

Implementation for the Engine

We decided to make the game third-person so the camera would be placed on the shoulder of the player model with the player shooting at the middle of the screen. We implemented this using multiple cameras within the Unity engine to get the view for the player correct.



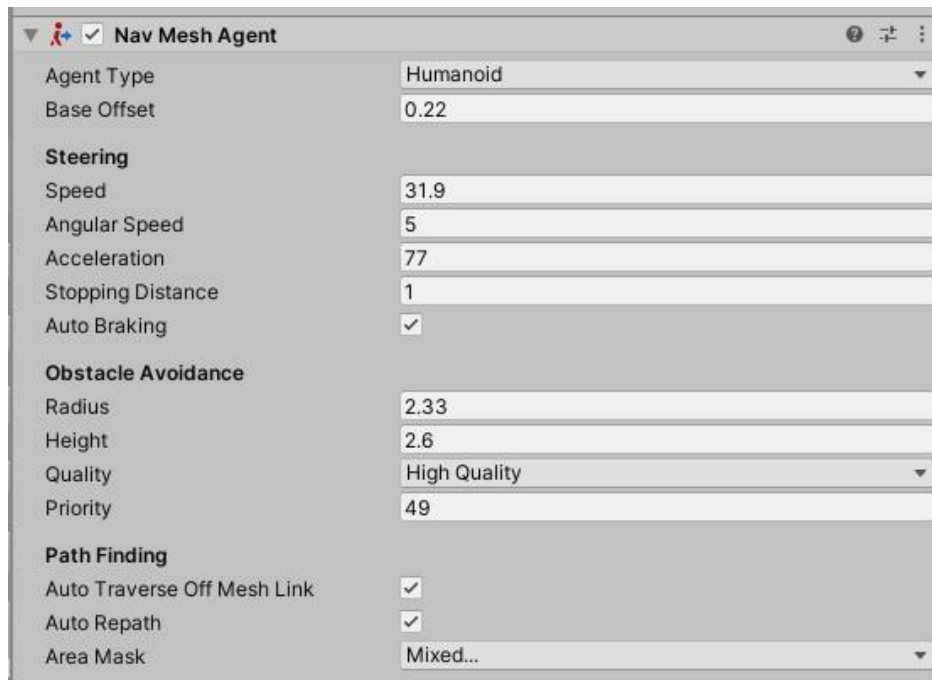
However, as a result of this style of camera position, scripting the shooting was difficult because trying to get the bullet objects to fly towards the middle of the screen while the player was moving their mouse around usually broke the code and the bullet would not work. That is why we used the Unity engines "Raycast" functions which casts a ray across the scene and checks for colliders/collisions. This is how we got our shooting to work because now the gun does not actually make physical bullets in the world, but the player can still shoot things.

```
RaycastHit hit;
if (Physics.Raycast(cam.transform.position, cam.transform.forward, out hit, range, layerMask))
{
    //Debug.Log(hit.transform.name);

    Enemy target = hit.transform.GetComponent<Enemy>();
    TargetPuzzle T = hit.transform.GetComponent<TargetPuzzle>();

    if (target != null)
    {
        target.takeDamage(damage);
    }
}
```

For the enemies in the game, we did not just want them to be static or move in set paths, so we implemented a simple AI for the enemies that would make them move around the scene with randomly created paths which used Unity's "NavMesh Agent" which was then used for all the enemies. However, this required all the floors to be set to the area mask so that the different enemies would stay in the rooms that they are meant to be in. More detail can be seen in the AI section of this report.



Game Testing

Throughout the game creation process, there was constant testing of newly implemented game mechanics to make sure these new mechanics would not break the game. As a result, throughout the game being created, we did not experience too many big bugs that we could not fix. The tests that were carried out to test new mechanics varied from just running the code and seeing if the expected outcome would happen or not to writing extra lines of code which would output a word to the console to make sure the code was calling the correct script functions that we expected. If a bug were encountered, more of these could be added to narrow down where this specific bug was located and then we could work to fix it before adding anything else in. By constantly testing we made sure that the mechanic that we just implemented did not break any pre-existing mechanics in the game, which also helped narrow down where the bugs would occur.

Another way of testing our game we did was play testing it. This allowed us to see how the game was doing up to that point, as well as pushing the game to the limit and trying to actively break it. Such as running into objects to make sure collision worked or shooting at the bottom and top of enemies to make sure their collision boxes were correctly sized. Play testing also allowed us to make sure the power ups that we added to the game were balanced so that one upgrade would not break the game. This meant that even if you got lucky and got all damage buffs, it would not mean the game loses all of its difficulty.

Development Report

Management – Team Roles

'How Could This Go Wrong' consisted of 5 members, three of whom being Games Software Engineer (GSE) students and the other two Games Design (GD) students. Team members listed below:

- Edward Truong (GD)

- George Mendoza (GD) • Leon Sen (GSE)
- Lewis Robertson (GSE)
- Nathan Hale (GSE)

At the groups inception it was collectively decided Hale would be the team leader. Here are the responsibilities the group decided to give the leader:

- Ensure project progress through open communication and democratic decision making.
 - Organising meeting date and times.
 - Overviewing current progress and establishing deadlines.
 - Appropriate delegation of tasks
- Main line of communication with the group's mentor Simant Prakoonwit.

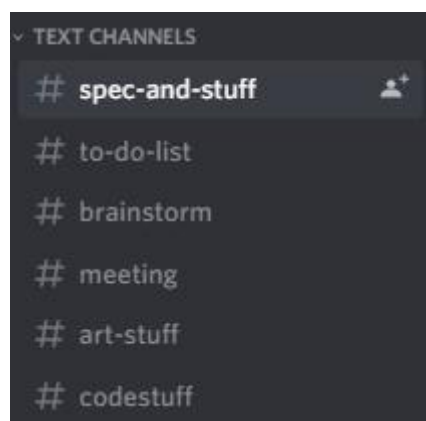
The other four members of the team had roles specifically pertaining to game development based on their course. GD student's jobs mostly consisted of creating models and art used in the game while GSE students were mostly scripting.

Edward Truong (GD)	Art and Design – Modeling and Texturing Report Writing – Art/Design sections
George Mendoza (GD)	Art and Design – Modeling and Texturing Report Writing – Art/Design sections
Leon Sen (GSE)	Lead Programmer - Scripting Report Writing – Mechanics/Gameplay sections
Lewis Robertson (GSE)	Programmer - Scripting Report Writing – Mechanics/Gameplay sections
Nathan Hale (GSE)	Group Leader – Meeting supervisor and organiser Lead Report Rriter – Oversee all sections and make adjustments/additions as necessary Programmer - Scripting

Further specifics on jobs carried out by each member can be found in the Gantt chart.

Management – Communication

All communication within the team was carried out on the application 'Discord.' Discord allowed the team to easily setup chats and organise information about the game into subsections. Below is a list of text channels:



These channels are designed to do these things respectively:

- Refer to the assignment brief and content provided by the university.
- Monitor progress and list action items decided upon in the meetings.
- Collate general ideas to be expanded upon and implemented.
- Any content pertaining to the current/past meetings, dates/times of meetings and logs of minutes.
- Ask questions about aspects of the game.

Management - Team Working Principles

Below are some team-working principles the team agreed were important and how they are adhered to throughout the project's lifecycle.

- Clearly defined and achievable goals ○ The Gantt chart was used to fulfil this requirement, assigning each member tasks which were discussed and agreed upon. Time periods for each task were estimated and adjusted throughout the projects lifecycle as unforeseen circumstances arose.
- Fair decision-making process ○ This was achieved by democratically deciding upon details concerning the project; including how tasks were assigned to individual mechanics implemented within the game.
- Active communication/Accountability ○ The group chat was active throughout the project's lifecycle even with off-topic conversation. Meeting times and deadlines were all available as 'pinned messages' within the chat so all members knew when and what the meetings would be about, and which parts of the project needed to be worked on.

Meetings

Meetings were used throughout the project lifecycle for various reasons.

Early on meetings were held frequently and often to discuss and agree upon the initial game idea, agree upon our expectations for the project and schedule the project to manage workload. There meetings were held weekly on Wednesdays with unofficial additional discussion in the group chat.

Later on in the lifecycle when everyone's role was defined within the project and knew what was expected of them, official meetings were no longer needed and meetings were scheduled as and when needed rather than having a set time each week. Much of the relevant information needed at this point was discussed within the group chat such as what progress had been made and what needed to be completed next.

All official meetings had a minutes log which detailed the meeting topic, points brought up and what was expected of the team members in regards to what should be completed prior to the following meeting. Minutes of the previous meeting were the first topic of every meeting, to discuss the work which had been completed in the interim period.

The meetings were successful in ensuring consistent progress was made on the game throughout the time we were given, avoiding excessive crunch at the end of production.

Planning

Planning for the project was done through the Gantt chart and deadlines in the group chat. The team collectively decided that whilst some tasks needed a clear goal and deadline, other aspects of the project were more nebulous as this was a creative project. The team believed for this reason the

agile project management methodology would suit best (more details can be found in the 'Design Implementation Techniques' section). For some aspects of the game the team worked off a vague idea and implemented it based off how they game had progressed. Some initial ideas were altered and/or scrapped as the vision for the game changed over-time.

Design Implementation Techniques

The project implementation technique which fit our needs best was the agile technique. Agile is defined by its ability to recontextualise the projects goal as developments occur which fit perfectly with the teams' ideals when creating the game. As the team progressed in the project and initial ideas were deemed too difficult/time consuming to complete, or in scenarios where we underestimated what we were able to do given our knowledge and resources, the final goals were altered. As an example, Agile helped us change our design and implementation of the level generation when we found a more suitable way to implement it. Agile allowed us to make changes to the initial design due to its nature of implementation and design being integrated and able to be altered as needed.

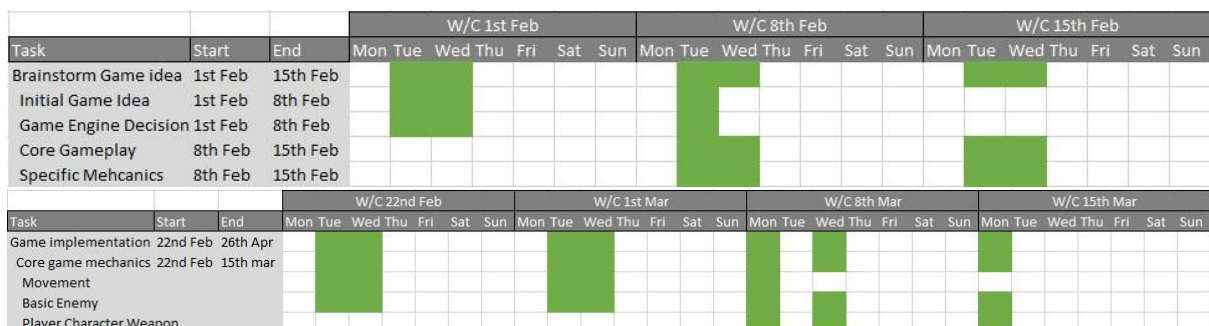
The section below provides further detail about how over the course of the project our goals changed.

Progress Tracking

To track our project progress, we initially used Gantt charts. The goal of these charts was to have a clear and defined goal all team members could work towards during early development. This worked well early on however as we got further in the project, we came to realise that we did not need a rigid idea of our final game and we could implement content based on what met the assignment brief or what we found interesting. Gantt charts were successful in creating the building blocks of our game but ended up feeling limiting later in the project once the groundwork had been completed. The team decided collectively that the members would implement content they personally took an interest in which also contributed to the assignment brief. The team leader was responsible for making sure the unplanned implementations were appropriate in achieving the initial brief.

Taking this approach ensured that team members were satisfied with the work they were given, rather than creating content which was pre-planned at the beginning of development without hindsight on the developments of the game over-time.

Below are screenshots taken from the Gantt chart.



Another way the group tracked progress was through weekly meetings with the group mentor, Simant Prakoonwit. In these meetings the mentor would ensure we are making progress on the game. The team created a presentation for the 4th week of production to display the game ideas thus

far so Prakoonwit could provide feedback and criticisms on the project. Finally, we were tasked to demo our game for Prakoonwit on the 9th week of production to once again ensure progress of the project.

Finally, informal meetings were held throughout the final few weeks of development to make sure each team member knew what was being worked on, what their role was and our deadlines for various tasks.

Minutes Log

Below are all Minutes ordered chronologically; collated by Nathan Hale over the course of the project. Minutes for 24/02 were overwritten and the file was lost.

Team Meeting

02/02/21
4:30-6:00 | 1:30

Meeting called by: Nathan Hale

Type of meeting: Discussion

Note taker: Nathan Hale

Timekeeper: Nathan Hale

Attendees: All members

Minutes

Agenda item: Brainstorm game ideas

Discussion:

- Brainstormed game ideas
- Elected team leader
- Created to-do list
- Selected game engine.

Conclusions:

- Decided upon game idea
- Elected Nathan Hale as team leader
- Game engine used will be Unity

Action items

- Presentation for week 3/4
- Game design document

Person responsible

All members
All members

Deadline

16th February
TBD

Team Meeting

10/02/21
12:15 – 1:45 | 1:30

Meeting called by: Nathan Hale

Type of meeting: Discussion

Note taker: Nathan Hale

Timekeeper: Nathan Hale

Attendees: All members

Minutes

Agenda item: Brainstorm further ideas and begin draft GDD

Discussion:

- Brainstormed game ideas
- Draft GDD

Conclusions:

- Refined upon game idea
- Put game idea into a GDD

Action items

- Presentation for week 3/4
- Game design document

Person responsible

All members
All members

Deadline

16th February
TBD

Team Meeting

17/02/21

12:30 – 1:45 | 1:15

Meeting called by: Nathan Hale

Type of meeting: Discussion

Note taker: Nathan Hale

Timekeeper: Nathan Hale

Attendees: All members

Minutes

Agenda item: Create presentation for 23/02/21

Discussion:

- Discussed ideas for presentation

Conclusions:

- Created Presentation

Action items

Person responsible

Deadline

- Game design document

All members

TBD

Team Meeting

03/03/21
12:30 – 1:15 | 0:45

Meeting called by: Nathan Hale

Type of meeting: Discussion

Note taker: Nathan Hale

Timekeeper: Nathan Hale

Attendees: All members

Minutes

Agenda item: Discuss scheduling, make targets for following week.

Discussion:

- Scheduling of tasks to create game

Conclusions:

- Members assigned tasks and mini meetings scheduled to work on game

Action items

Person responsible

Deadline

- Make starting area of game
- Game design document

All members
All members

09/03/21
TBD

Team Meeting

17/03/21
12:30 – 13:00 | 0:30

Meeting called by: Nathan Hale

Type of meeting: Discussion

Note taker: Nathan Hale

Timekeeper: Nathan Hale

Attendees: All members

Minutes

Agenda item: Discuss scheduling, make targets for following week.

Discussion:

- Scheduling of tasks to create game demo for 24/03

Conclusions:

- Members assigned tasks and mini meetings scheduled to work on game

Action items	Person responsible	Deadline
• Make prototype/demo	All members	24/03/21
• Game design document	All members	TBD