# Maintenance Guide

Pink Team

December 20, 2021

## Contents

## List of Figures

## List of Tables

## 1 Introduction

This report documents the processes and design philosophies that we recommend to maintain and manage this project as it continues to grow in scale. In this document we make the assumption that the audience has some experience in Object Orientated Programming but we assume no prior knowledge of the Unity game engine.

## 2 Developer Installation Guide

### 2.1 Unity Version

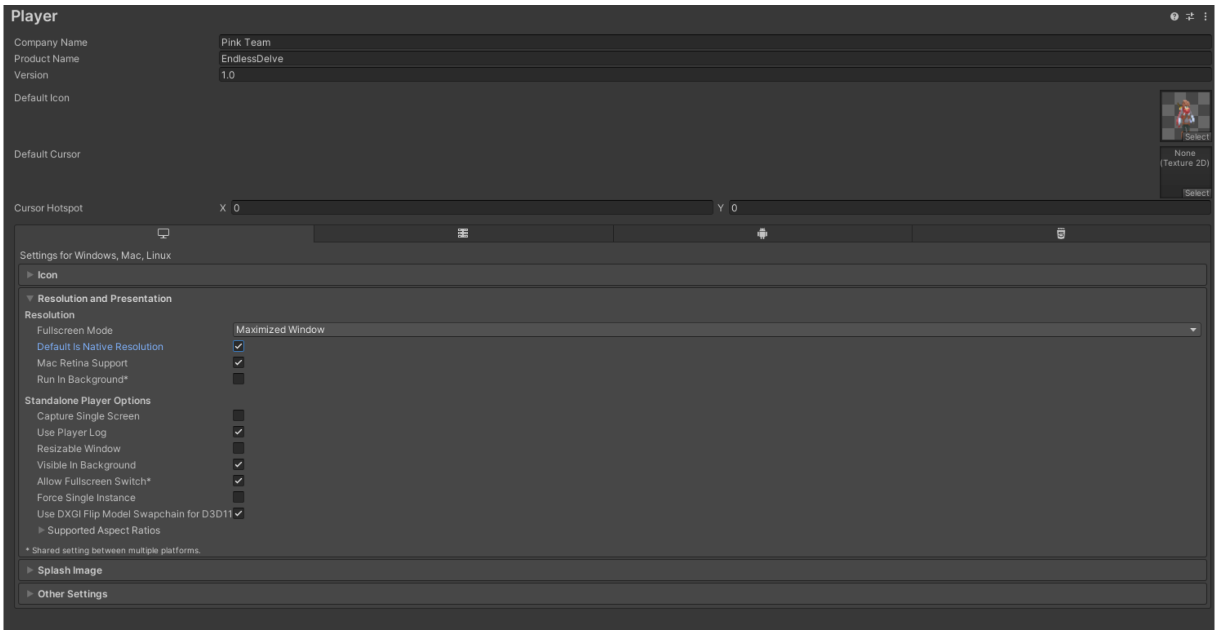
We are using Unity LTS (Long Term Support) version 2021.2.2. This ensures that the engine is well-tested and stable. Furthermore, this version is guaranteed to be supported for at least two more years. As a group we have decided upon using this version and we will not consider updating unless there is a new experimental feature that we need to include in our project. Developers should not update their version of the project as it forces everyone else to up/downgrade their project to the new version which can take some time.

### 2.2 Building the project

To configure how the game is build go to the tool bar at the top left of unity and select File →build settings. The game is currently supported for WebGL and Windows/Mac/Linux. You can select your preferred platform in bottom left platform menu. To configure the build in more detail select the *Player settings* button in the bottom left of the menu. In this screen you can edit the settings of the build including Company Name, Product Name, Version and launcher icon. More importantly you can configure default game settings like native resolution enabled or full-screen mode.

### 2.3 Unity Installation

To install and setup Unity, you must first download Unity Hub, from the official website of Unity. This can be accessed from `https://unity.com/download`. Install Unity Hub in a suitable location, and preferably create a desktop shortcut. Unity Hub is a general manager for the Unity Editor, and different versions of the editor can be installed for diffferent projects. To install the necessary editor version for this project, navigate to ”Installs” on the left hand side of the menu, and select ”Install Editor”, situated on the top right of this page, containing a blue background. From here, under the section titled ”Long Term Support (LTS)” select the editor version that we use in this project, which is 2021.2.2. Now the editor will install. Once installed, navigate to the gear icon which appears to the right of the installed editor version, and click it. This will open a small tab, with the options ”Add modules”, ”Show in explorer”, and ”Uninstall”. We also build and develop this game for MacOS and Linux platforms, and need the modules to do this. Select ”Add modules”, and tick the ”Linux Build Support (Mono)” and ”Mac Build Support (Mono)” options and hit ”Install”. Now, you have installed Unity, the required editor, and the modules used for this game.



**Player**

| | |
|---|---|
| Company Name | Pink Team |
| Product Name | EndlessDelve |
| Version | 1.0 |
| Default Icon | Select |
| Default Cursor | None (Texture 2D) Select |
| Cursor Hotspot | X 0 Y 0 |

Settings for Windows, Mac, Linux

- Icon
- Resolution and Presentation

**Resolution**

| | |
|---|---|
| Fullscreen Mode | Maximized Window |
| Default Is Native Resolution | ✓ |
| Mac Retina Support | ✓ |
| Run In Background* | ☐ |

**Standalone Player Options**

| | |
|---|---|
| Capture Single Screen | ☐ |
| Use Player Log | ✓ |
| Resizable Window | ☐ |
| Visible In Background | ✓ |
| Allow Fullscreen Switch* | ✓ |
| Force Single Instance | ☐ |
| Use DXGI Flip Model Swapchain for D3D11 | ✓ |

- Supported Aspect Ratios

* Shared setting between multiple platforms.

- Splash Image
- Other Settings

Figure 1: Build Settings Menu

## 2.4 Doxygen

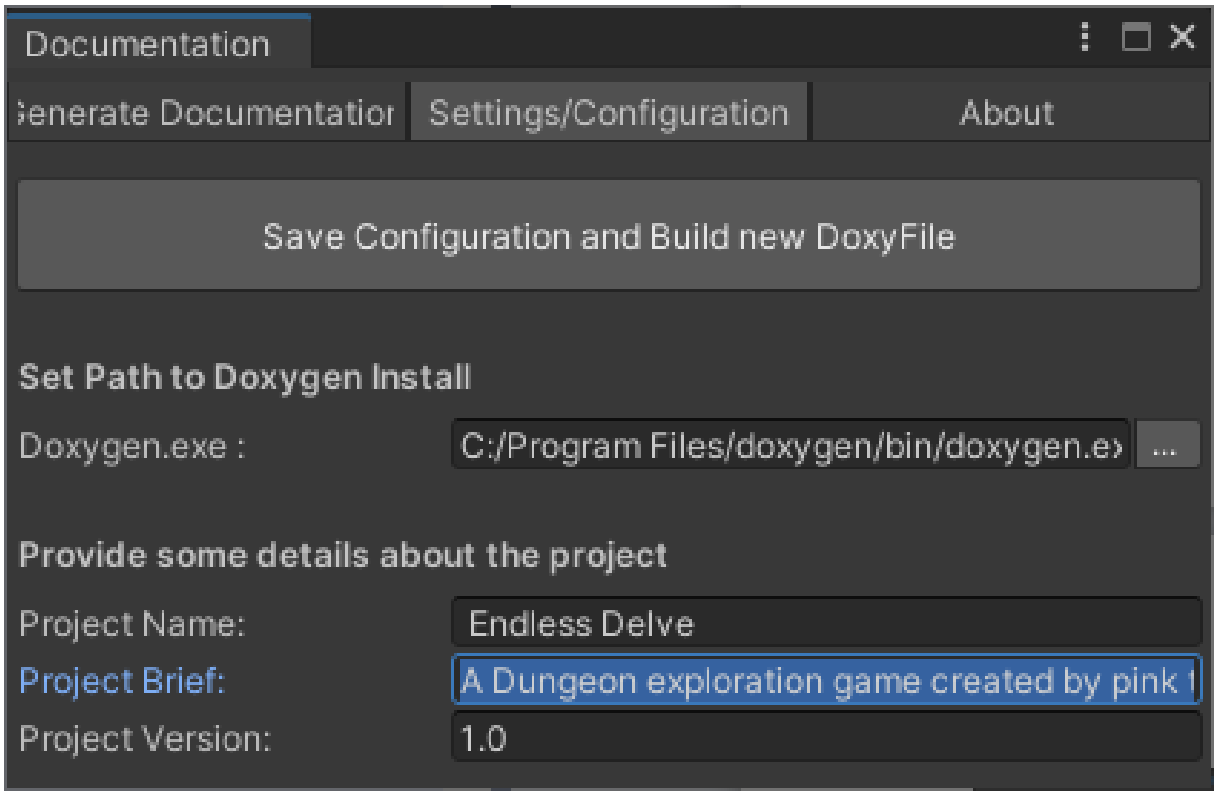
Doxygen is required to be installed by any developer that wishes to regenerate the Doxygen documentation for the code base. You can find the tool at the following link:`https://www.doxygen.nl/download.html`.

For windows devices the installation is a simple process, Doxygen comes as a self installing archive. Simply run the executable file and follow the dialog prompts to install doxygen Wizard.

Alternatively you can clone the git repositry at: `https://github.com/doxygen/doxygen.git`. This can be done by running the following commands in sequence:

- git clone https://github.com/doxygen/doxygen.git
- cd doxygen
- mkdir build
- cd build
- cmake -G "Unix Makefiles" ..
- make
- make install

Inside our Unity project we have pre-installed an editor plugin which automatically generates Doxygen documentation for all scripts within the project. To use this tool you need to open the plugin. To do this you select 'Window' → Documentation with Doxygen. →Settings/configuration.



Set Path to Doxygen Install

Doxygen.exe : C:/Program Files/doxygen/bin/doxygen.e› ...

Provide some details about the project

Project Name: Endless Delve

Project Brief: A Dungeon exploration game created by pink t

Project Version: 1.0

Figure 2: Configuration menu for Doxygen Plugin

From this window we need to set our path to the Doxygen install. To do this select the ”...” button and locate the location of where you installed Doxygen. By default on a windows device it is located at *C/Program Files/doxygen/bin/doxygen.exe*. You should now be able to produce Doxygen documentation. To do this navigate from setting/configuration$\rightarrow$Generate Documentation. From here you can click ”Browse Documentation” to open the Doxygen documentation for this project in your default browser. To update the Doxygen documentation you click the ”Run Doxygen” button. The Doxygen produced will be placed in a folder called docs in the projects main folder. To change the format Doxygen is outputted in you can change the Endless Delve/Docs/DoxyFile. One very useful change that can be made is changing the Extract_Private = NO. This will cause all private variables and methods to be ignored. This will make the Doxygen produced more similar to API documentation.

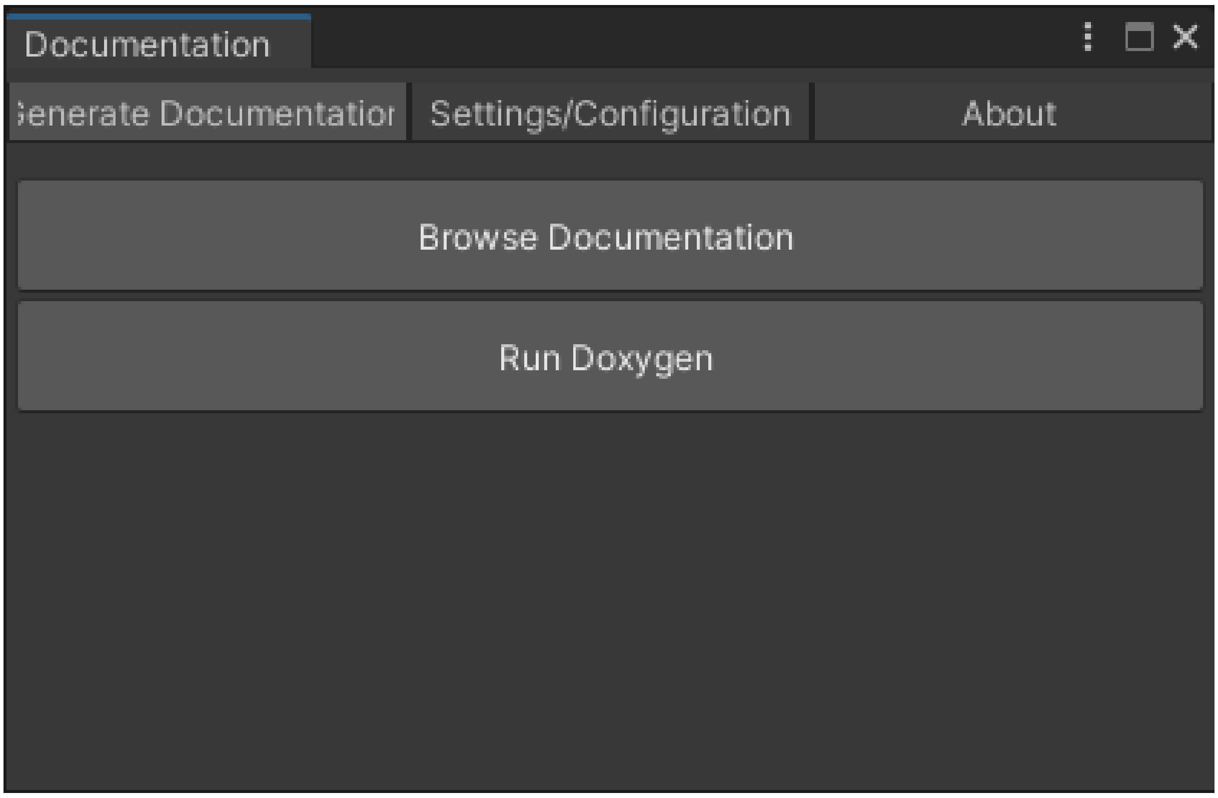


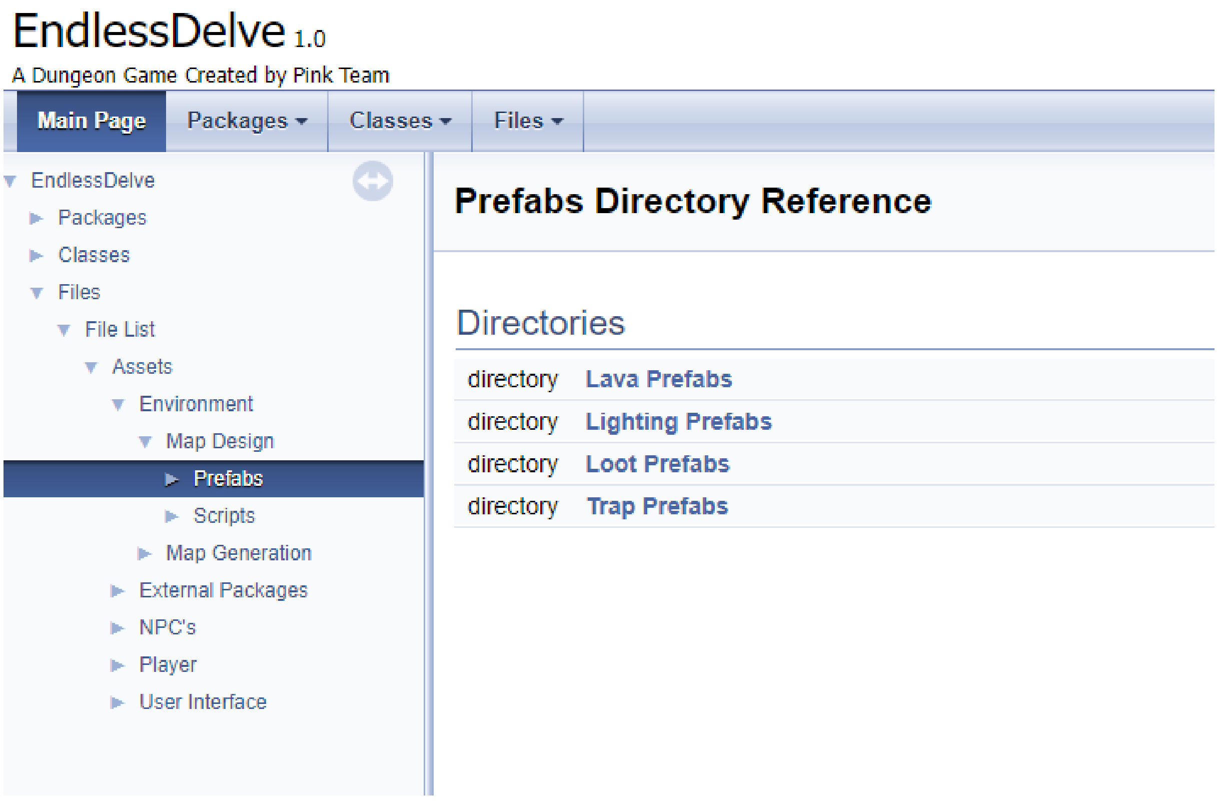
Figure 3: The Editor plugin used to configure the Doxygen Output

### 2.5 Accessing Doxygen Without Package/Unity Installed

We will also provide instructions for the documentation team to access the Doxygen documentation since they may not have access to an up-to-date version of unity. To open the Doxygen you can access it in the main folder of the project. It is located at *EndlessDelve/Docs/html/index.html* simply run this file and the Doxygen should open in your default browser.

## 3 Project Structure

Since we were following a Feature driven development structure we decided to layout our project structure based on features. In this structure each folder reflects the categories present in the Product Backlog's item name. For example, Any items labelled Player:XXXXX will be present in Player folder of our repository. As items in the product backlog are split into smaller more manageable items this should be reflected in the folder layout and sub folders should be created. Every feature folder is broken down into sprite(including animations), prefabs and scripts. To view the full project structure see the files section of our Doxygen documentation.


EndlessDelve 1.0

A Dungeon Game Created by Pink Team

Prefabs Directory Reference

Directories

| | |
|---|---|
| directory | Lava Prefabs |
| directory | Lighting Prefabs |
| directory | Loot Prefabs |
| directory | Trap Prefabs |

Figure 4: The File Listings for our project

## 3.1 Static Variable Classes

We use public static classes to store the variables relating to the player and map. This allows them to be accessed by different objects from other scenes without worrying about passing the reference to the player and map between too many objects and scenes.

### 3.1.1 Map Variables

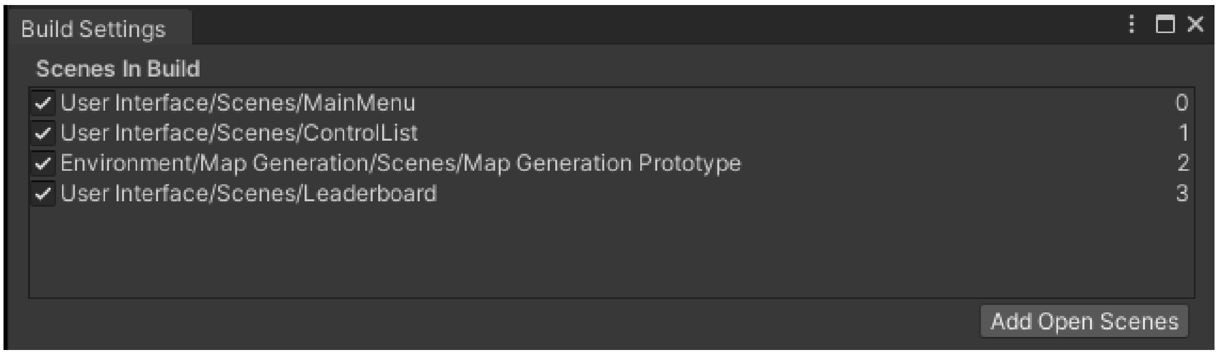
The map variables class stores the current difficulty of the level. It is also where we store commonly used layer masks so that we only have to update them in one place if new layers need to be added. All other classes access the layer masks from here.

### 3.1.2 Player Variables

The Player Variables script stores the players health, number of torches and insanity. It also stores the the max amount of all these values so they can be reset when a new dungeon is spawned. The HUD interacts with this class to display these values.

## 3.2 Scene Manager

Scene manager is used to control which scenes will be built and added to the game. The MainMenu, ControlList, MapGenerationPrototype and Leaderboard all need to remain in the build settings in the order shown below. To add a new scene we need to open the build settings menu by selecting file→build settings. From this menu you can either click and drag a scene into the hierarchy or click add new open scenes.



Build Settings

Scenes In Build

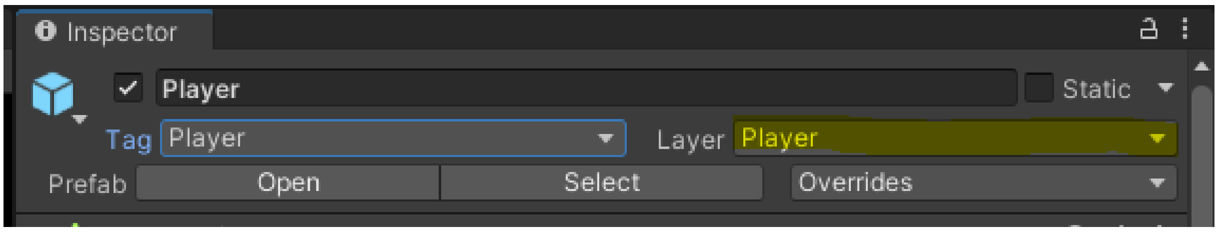
| | |
|---|---|
| ✓ User Interface/Scenes/MainMenu | 0 |
| ✓ User Interface/Scenes/ControlList | 1 |
| ✓ Environment/Map Generation/Scenes/Map Generation Prototype | 2 |
| ✓ User Interface/Scenes/Leaderboard | 3 |

Figure 5: Unity's scene manager menu

### 3.3 Layers

Layers in unity can be used to determine which objects are effected by certain functionality. They are most commonly used by cameras where we only want to render one part of a scene or by lights where we choose which parts of the scene we want to illuminate. In our project we make extensive use of layers to determine which types of objects our collisions should ignore. A Layer should now be removed unless you are certain that its functionality is no longer needed.

To add a layer to an object select it to open the inspector. At the top of the inspector menu select the Layer drop down menu and then select the layer you wish to add from it.



Inspector

✓ Player

Static

Tag: Player

Layer: Player

Figure 6: Adding a Layer to an object

Table 1: Table of all Layers used Project.

| ID | Name |
|---|---|
| 0 | Default |
| 5 | UI |
| 6 | Room |
| 7 | Spike |
| 8 | RoomConnection |
| 9 | BasicBlock |
| 10 | Rope |
| 11 | Loot |
| 12 | Ladder |
| 13 | Lava |
| 14 | Player |
| 15 | Mob |
| 16 | Border |
| 17 | Door |
| 18 | Torch |
| 19 | SwingTrap |

### 3.4 Tags

A Tag is a reference word which you can assign to one or more GameObjects. Tags are used to group objects that share common characteristics. For example all hostile NPC's can be given the enemy tag. We can check for collisions on all enemies instead of checking it against each mob individually. To add a new tag to the project, select any object and select the tag drop-down menu at the top of the inspector. Select add tag from the drop down menu.

Table 2: Table of all tags used in our Project.

| Tag | Where they have been used |
|---|---|
| Destructible | Define which objects can be destroyed when bombs are dropped |
| Collectable | Define which objects in the scene can be collected by the player |
| Enemy | Define which objects are hostile to the player |
| Door | Groups exit and entrance door into one tag |

## 4 Coding Practices

### 4.1 Git Hub Best Practices

- Never commit directly to main branch.
- Always commit a single feature at a time.
- GitHub commit messages should be descriptive and concise.
- Code should be reviewed by at least one other team member to ensure that it is consistent with coding standards and free of any logic errors.

## 4.2 Unity Best Practices

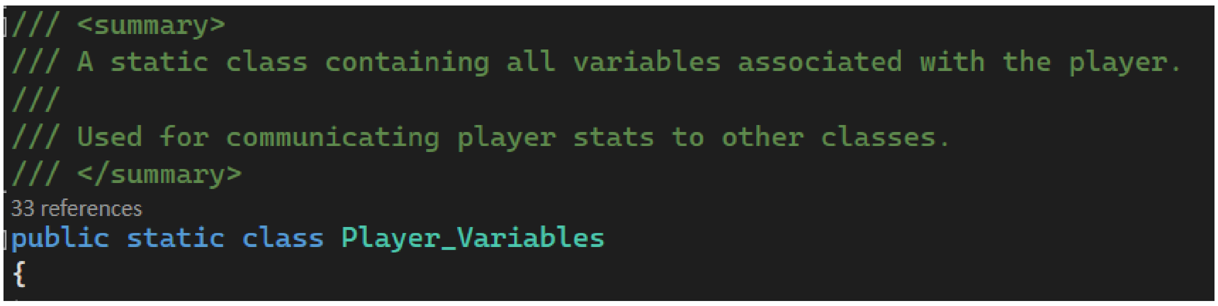
- Everything that will be displayed inspector should be given a default value.
- Gizmos should be included for debugging hit-boxes and collisions. They should be able to be toggled on/off individually for clarity.
- All player inputs retrieved in update, all movement done in fixed update
- Player.getComponent() should not be used in update/fixed update unless it is unavoidable.
- Animation logic should be separate from game logic.
- Features should be separated and contained within their own scripts e.g. the Patrolling enemy script is applicable to multiple mobs.

## 4.3 Doxygen

In this section we will provide all the information to continue producing detailed Doxygen documentation going forward. We will give code examples and then show how the changes are reflected in the documentation.

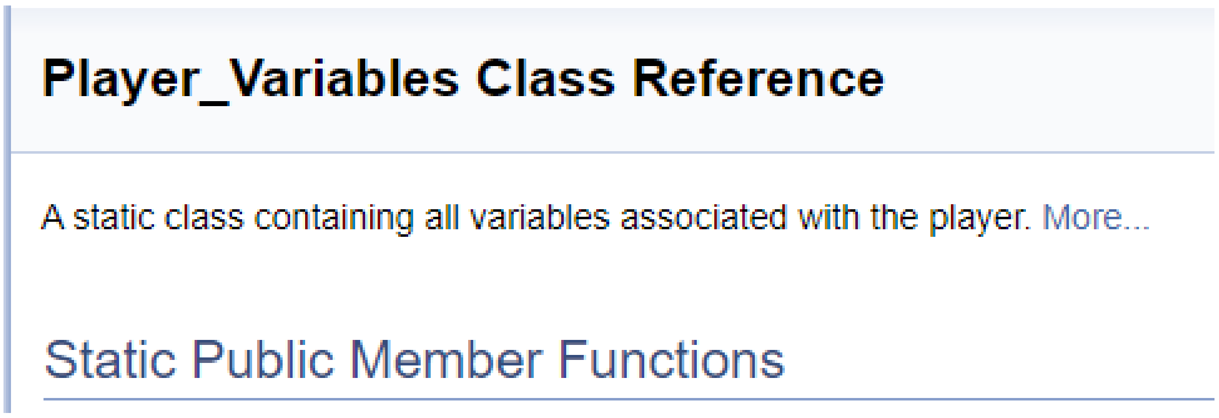
## 4.4 Commenting a class

For a class to be included in the Doxygen documentation there needs to be a Doxygen comment before the class definition. If you are using visual studio which, Unity highly recommends, you can automatically generate Doxygen comments by typing ///. There are several formats for comments that produce Doxygen comments. We are using the standard version that visual studio produces automatically since the majority of the team is developing in this IDE. This format should be kept consistent throughout the project for clarity.



```
/// <summary>
/// A static class containing all variables associated with the player.
///
/// Used for communicating player stats to other classes.
/// </summary>
33 references
public static class Player_Variables
{
```

Figure 7: A code example to specify a Doxygen Class Definition for the Player Variables class.


Player_Variables Class Reference

A static class containing all variables associated with the player. More...

Static Public Member Functions

Figure 8: An example Doxygen Class Definition for the Player Variables class.

### 4.5 Commenting variables

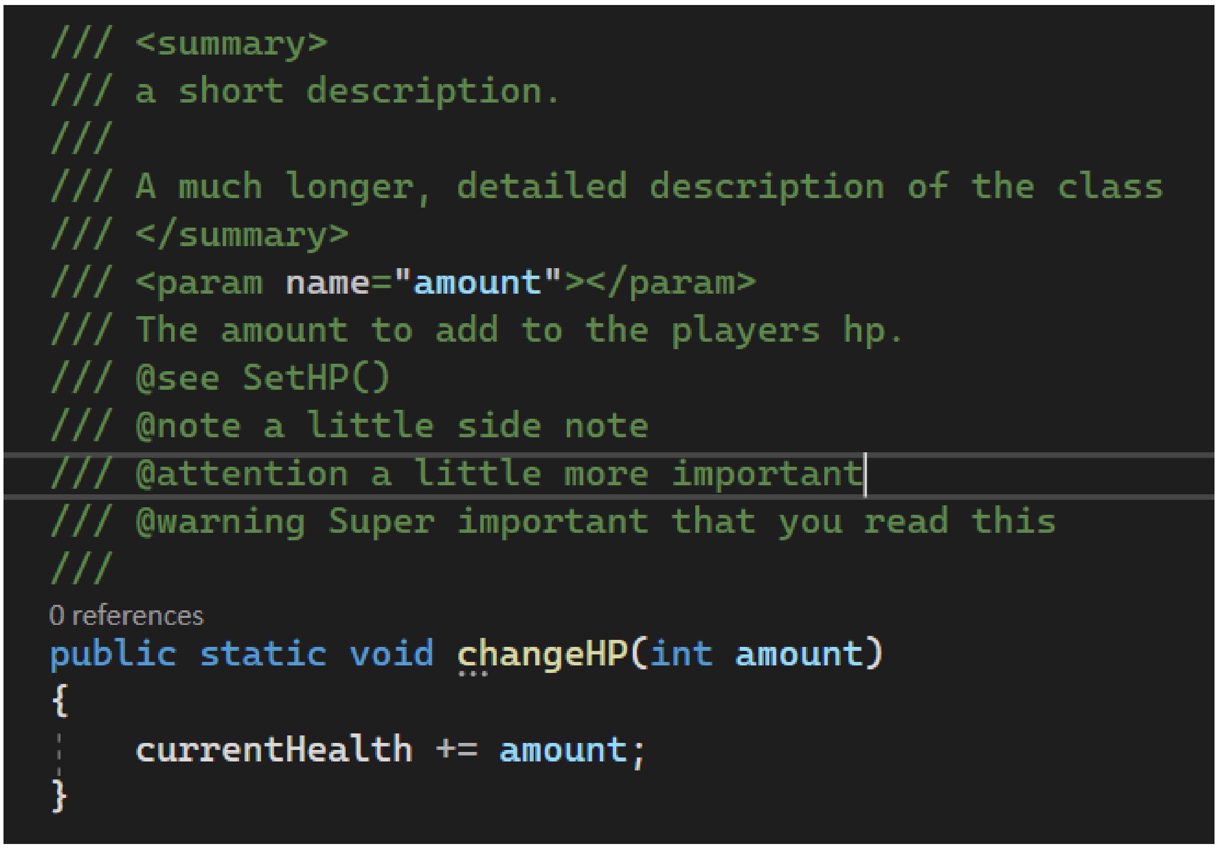
To include variables in the documentation we need to add /// before the variable. The comment given should be descriptive.

### 4.6 Commenting methods

To comment a method you simply type /// above the method definition and visual studio will generate the basic parametre and return values to fill in. To add a description to these values you comment underneath. We can also use @commands to draw attention to details developers should not miss.

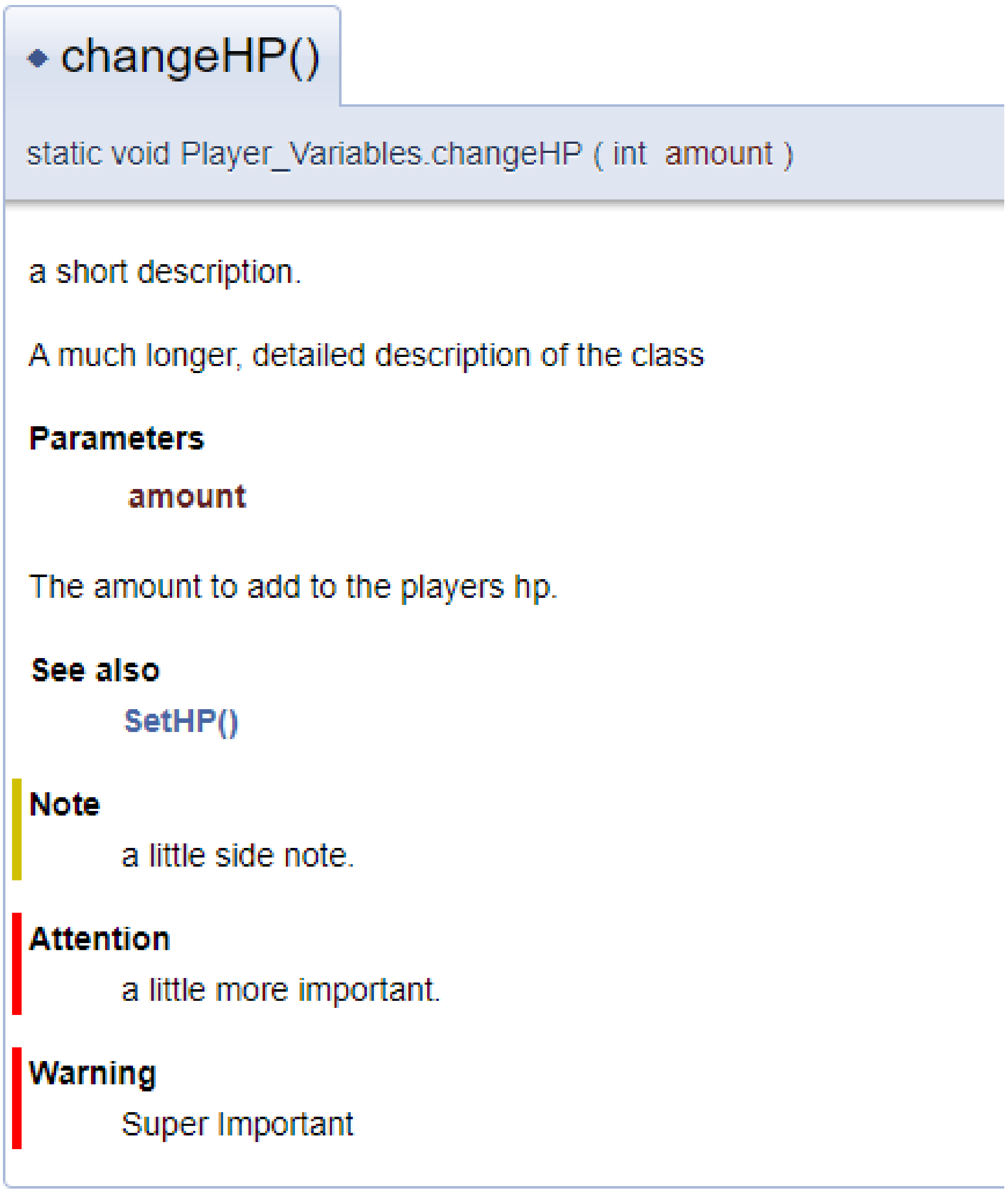
- The @see command can be used to point to another member function.
- The @note command can include a little extra note to tell the developer.
- The @attention command creates a important note for the developer to read.
- The @warning command creates a very important warning that should not be missed by the developer.

It is also possible to use markup language to highlight important key-word phrases



```
/// <summary>
/// a short description.
///
/// A much longer, detailed description of the class
/// </summary>
/// <param name="amount"></param>
/// The amount to add to the players hp.
/// @see SetHP()
/// @note a little side note
/// @attention a little more important
/// @warning Super important that you read this
///
0 references
public static void changeHP(int amount)
{
    currentHealth += amount;
}
```

Figure 9: A code example of how to comment a method in Doxygen, showing the commands that are available to a developer


◆ changeHP()

static void Player_Variables.changeHP ( int amount )

a short description.

A much longer, detailed description of the class

**Parameters**

**amount**

The amount to add to the players hp.

**See also**

SetHP()

**Note**

a little side note.

**Attention**

a little more important.

**Warning**

Super Important

Figure 10: An example Doxygen method showcasing how messages can be displayed to reader

## 5 Map Creation

The Map Spawner script is responsible for spawning all the rooms that populate the world. The script generates a critical path from entrance room to exit room. The script ensures that all rooms along this path are connected by at least one shared entrance. Rooms are semi randomly, semi manually designed. Some Specific tiles have a percentage chance to spawn such as traps, each block tile has a chance to spawn as a decorative tile and some spawn blocks spawn a random chunk of blocks to further increase randomness.

## 5.1 Room Design

All the tools that can be used to design a room can be found at *Assets/Environment/Map Design/Prefabs*. There are 4 types of rooms, LR, LRB, LRT and LRTB, LR is a room with a left and right exit, LRB is a room with a left, right and bottom exit etc. each Room created must have a RoomType script attached which just assigns the room its identifier type.
A rooms must have the dimensions of 10 blocks wide and 10 blocks high. They should be designed around the limits of player movement; players should be able to move from any entrance specified in name to any other exit of the room. Finally, there must be at least two adjacent room connections of the middle four tiles of a exit. Room designers should be mindful of this criteria when creating rooms.

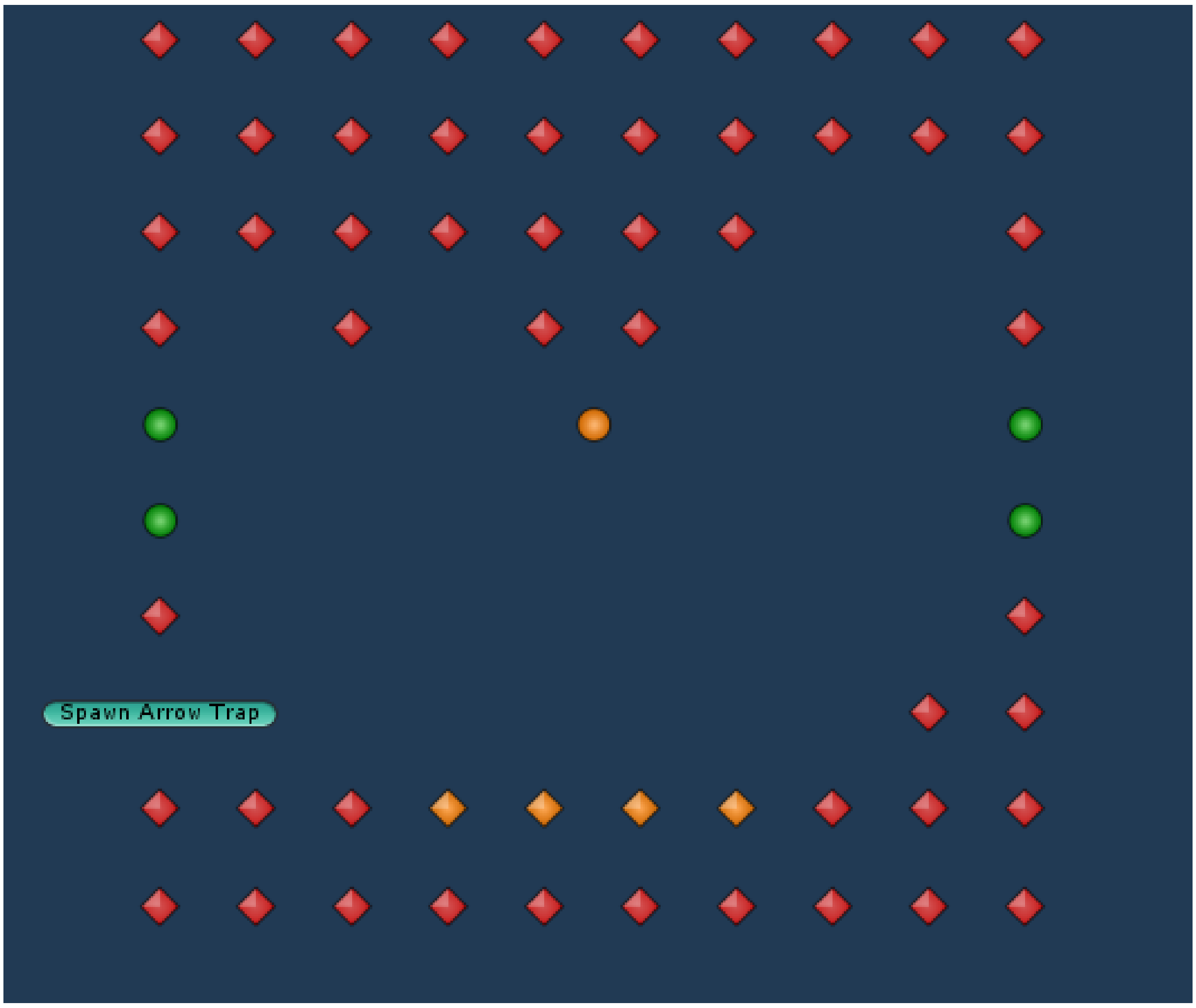


Figure 11: A room with left and right entrances before it has spawned the tiles

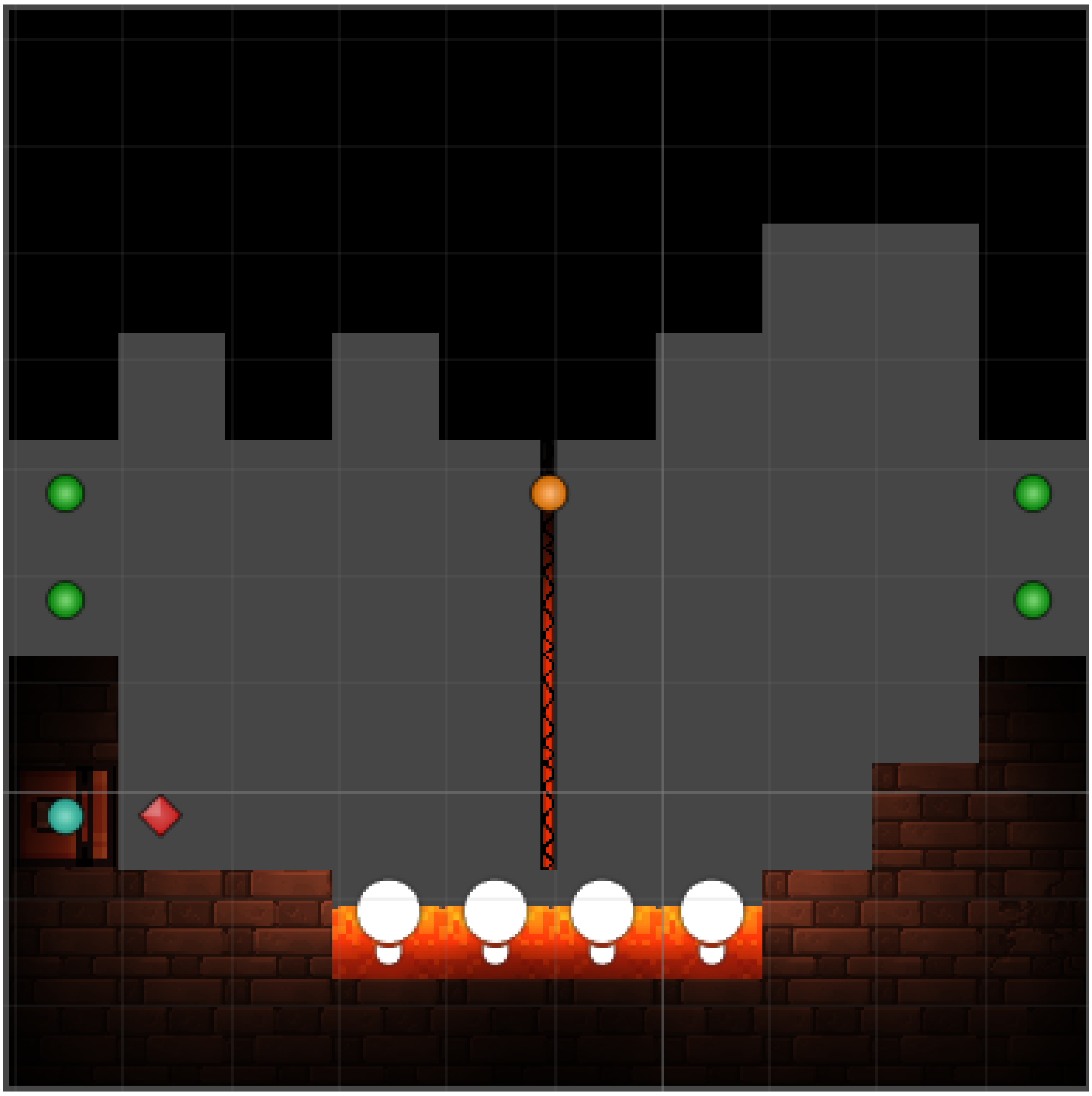


Figure 12: A room with left and right entraces after the spawnTiles have spawned their objects

## 5.2 Visualising Room Design with Gizmos

To make sense of the empty gameobjects that are populated to create rooms we have assigned Gizmos to each spawnObject. Gizmos are used to give visual debugging or setup aids in the Scene view. The Colour and shape of a Gizmo informs room designers of the object type it will spawn.

| **Key** | **Gizmo Used** |
|---|---|
| Basic Block | Red Diamond |
| Room Connection | Green Circle |
| Rope | Orange Circle |
| Lava | Orange Diamond |
| Spike | Blue Diamond |
| Arrow Trap | Blue Circle |
| Spawn Ladder | Yellow Diamond |

### 5.3 Creating a New Room

Before attempting to edit or create a room you should reconfigure your grid snap points to move in 0.5 increments. This makes the correct placement of tiles a much more efficient process. To do this select the magnet icon attached to the scene view. Set the grid size values of both X and Y to 0.5. Holding Ctrl whilst moving an object will now move in 0.5 (the correct)increments.

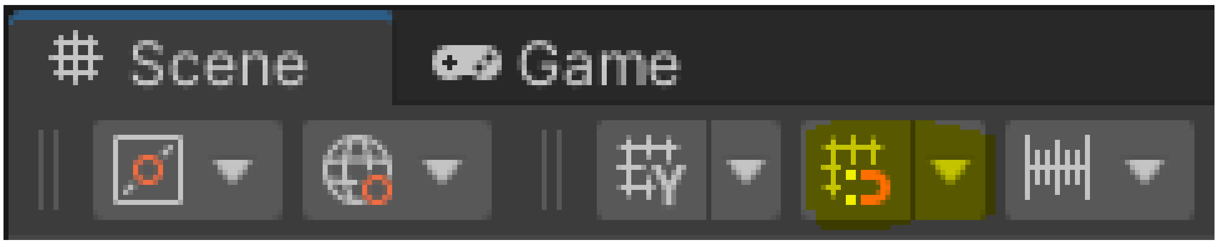


Figure 13: A Guide to enable grid snapping increment

The easiest way to create a new room is to duplicate one of the existing rooms at *Assets/Environment/Map Design/Prefabs/Room Prefabs*. Select one of the rooms in one of the room type sub-folders. Press Ctrl+D to duplicate it. Its name should then be changed to a unique ID i.e. LRXX where XX is a unique ID number. To edit this room double click the prefab to open the prefab editor. In this prefab editor you can delete/move/duplicate any of the spawn objects. You can also add any of the spawn tiles from *Assets/Environment/Map Design*. If ropes are added the length should be specified in the SpawnRope script attached to the object. Once you have finished creating the room you need to add Connection-Points to any open tile on the entrance ways of the rooms. To add this room into the spawning algorithm you need to add it into the corresponding RoomType template found at: *Assets/Environment/Map Design/Prefabs/Room Prefabs*. This will automatically get added to the room spawning algorithm.

## 6 Testing Environments

Test scenes should be created for major features that need to be tested quickly and iteratively. For example, in the game currently to test if the player can interact with the exit door the developer has two options. Firstly, they could move through an entire level to reach the exit. Alternatively, they would have to enter the scene view and manually adjust the player game object position. Both processes are very time consuming especially if it takes many attempts for the developer to achieve the desired functionality. We recommend creating a test scene to be able to test a feature quickly.

Like game objects, test scenes should maintain modularity. If the scene becomes to large or is responsible for testing too many features then it loses its efficiency. We have therefore created a separate test scene for experimenting with player/mob interactions.

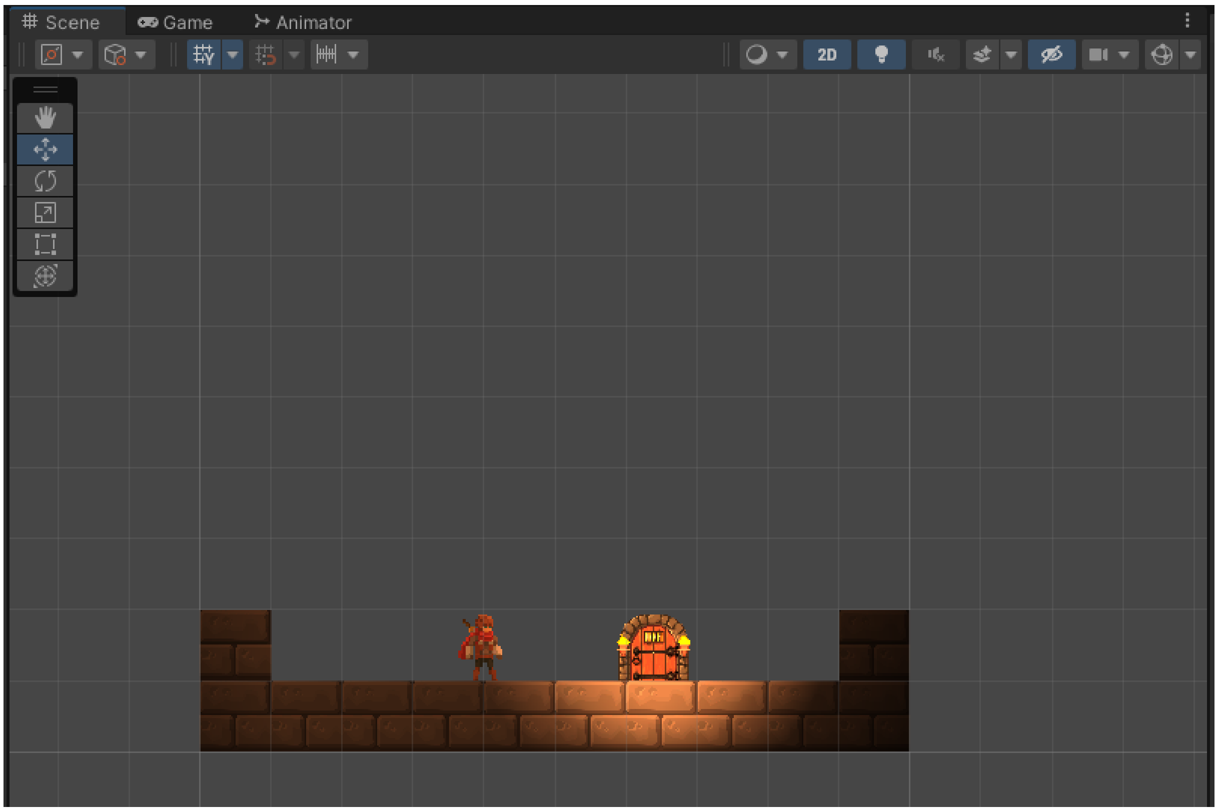


Figure 14: A Simple test environment created to check if the player can interact with the exit door

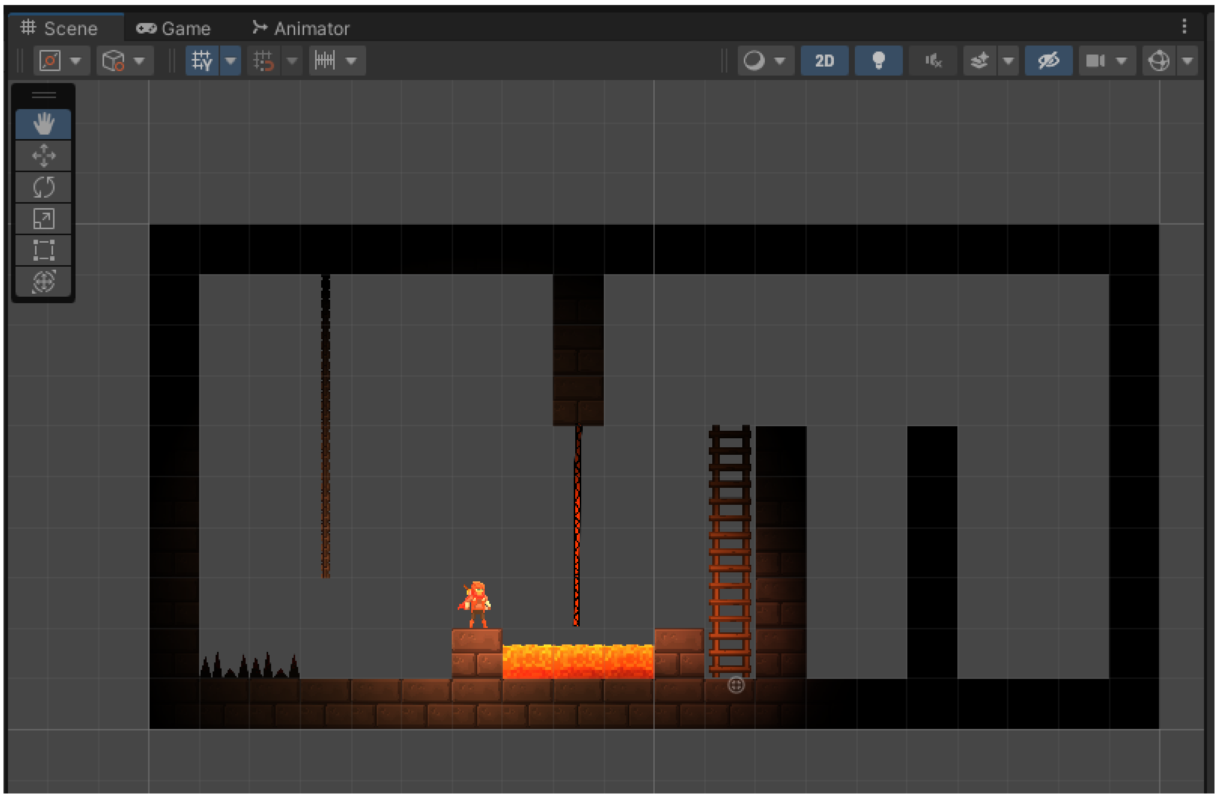


Figure 15: A Test Environment containing all objects a player can interact with, within a condensed area for ease of testing.

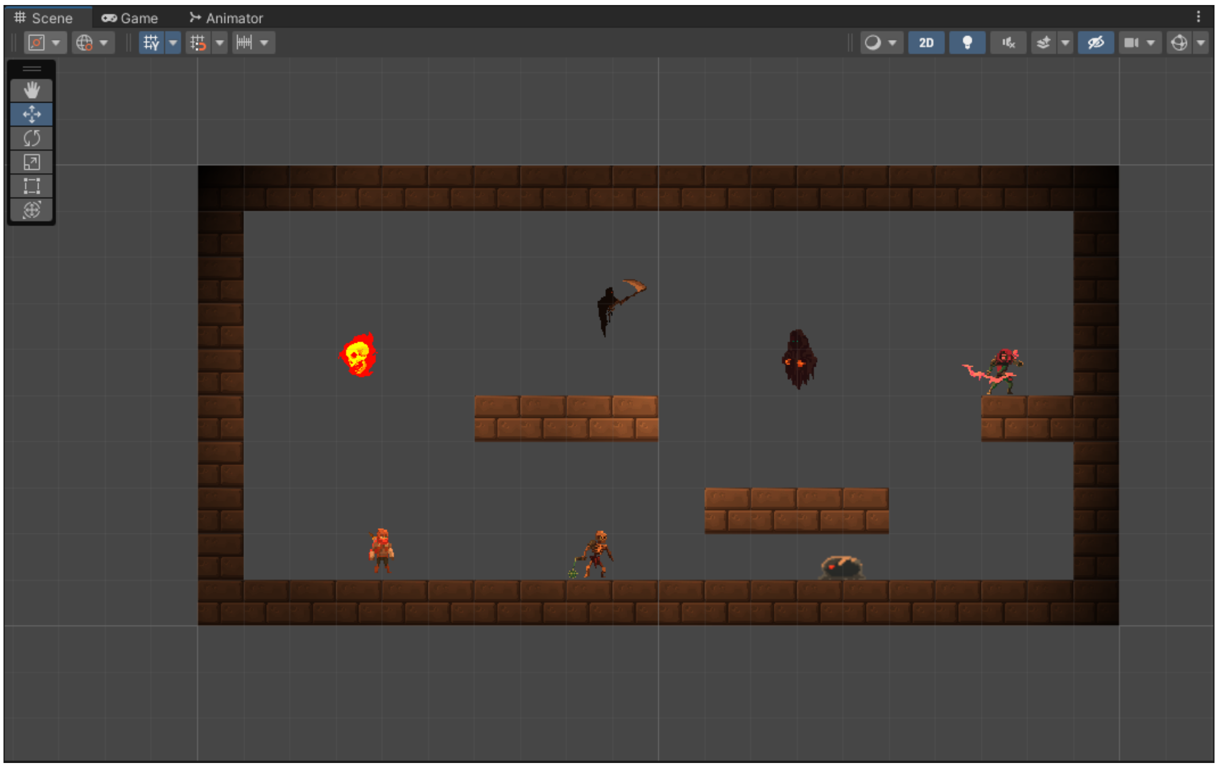


Figure 16: A Test Environment containing all mobs that a player can interact with, within a condensed area for ease of testing.

Test scenes are also useful in testing the subjective features of the game. An example feature to be tested is the effect that the angle limits of the hinge joints of ropes effect the physics engine. Developers can quickly create a scene with multiple labelled ropes with differing angle limits set. This allows the developer to experiment quickly and find out what rope angle limits produce the desired behaviour to maximise the ”usability” of the player. We found that the small amount of development time spent creating these scenes greatly reduced the time we spent testing the game.

Once a feature produces the desired functionality in a test scene it should be confirmed that its behaviour is consistent in a real-game scenario; Test scenes sole purpose is to speed up development times and is not a replacement for acceptance and unit testing.

## 6.1 Creating A test Environment

If the folder containing the feature you are trying to test does not contain a ”TestScenes” folder, then you should create one. In this example we will demonstrate the process of creating a scene to test player combat. We will first create a new scene and add the player prefab. To create a new scene, right click the project folder section, select create and then scene. We now need to create a room in the scene to contain the test.

We have prepared several prefab test rooms which can be placed into the scene. They can be found at *Assets/Test Scene Assets/Test Rooms*. Simply click and drag the desired one into the scene. In this example we will use the ”CompletelyEmpty” test room.

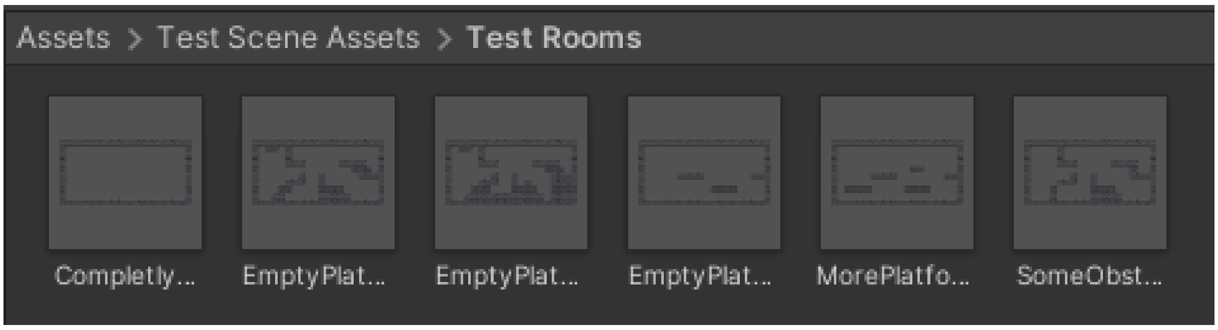


Figure 17: A project folder view of premade test rooms available in the project.

To add the player prefab that is used in our main scene we need to navigate to *Assets/Player/Prefabs/Player.prefab*. From here you can click and drag the player into the room. Next we will add the mob we wish to test into the scene. In this example we will use the skeleton located at: *Assets/NPC's/Hostile NPC's/Skeleton/Skeleton enemy.prefab*. We can now test the combat of the player by running the scene and attacking the skeleton.

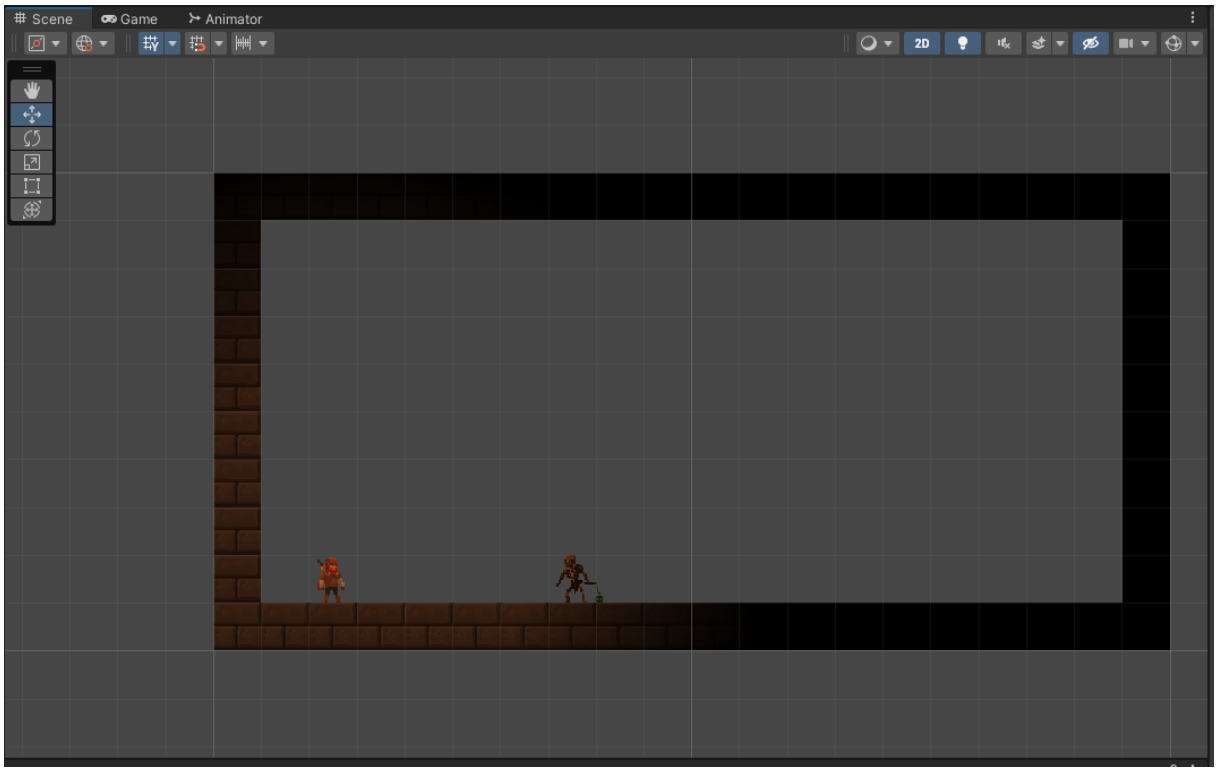


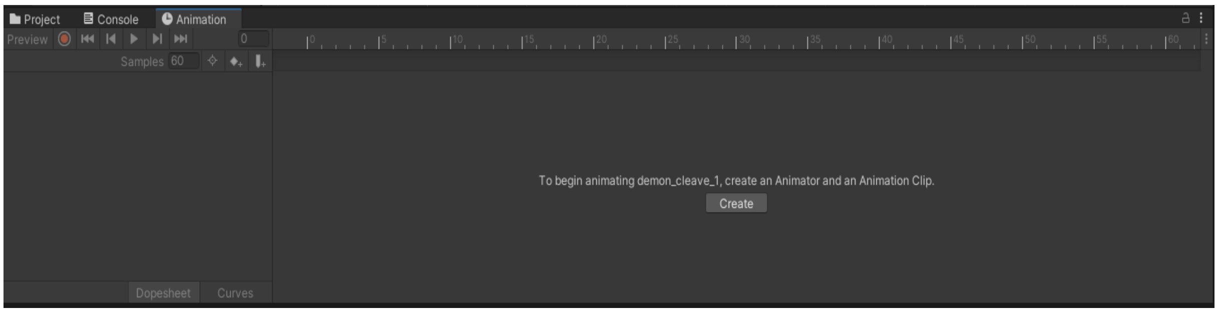
Figure 18: How the worked example of player combat test scene should look so far.

However, once the skeleton dies we have to restart the scene to test again. A better solution is to write a simple script that spawns a new mob from an array of mob prefabs when the previously spawned mob dies. The 10 minutes spent creating this script allows for unlimited testing without the need to restart and the scene can be used to test any mob. We recommend this design philosophy is followed for the creation of future test scenes. All scripts that are exclusively used for testing should be labelled as TEST_Scriptname

# 7 Scenarios

## 7.1 Adding an Animation Clip

A Folder should be created for each animation in the features sprite folder. The folder should be called ObjectName_AnimationName. The object should be selected and then Window→Animation should be selected.



To begin animating demon_cleave_1, create an Animator and an Animation Clip.

Figure 19: How an animation window looks if you select a game object without an animator

If the object does not have an animator or animation clip attached then they will be automatically generated by clicking create. You are now ready to add animations. Click and drag the sprites that make up the animation into the animation window.

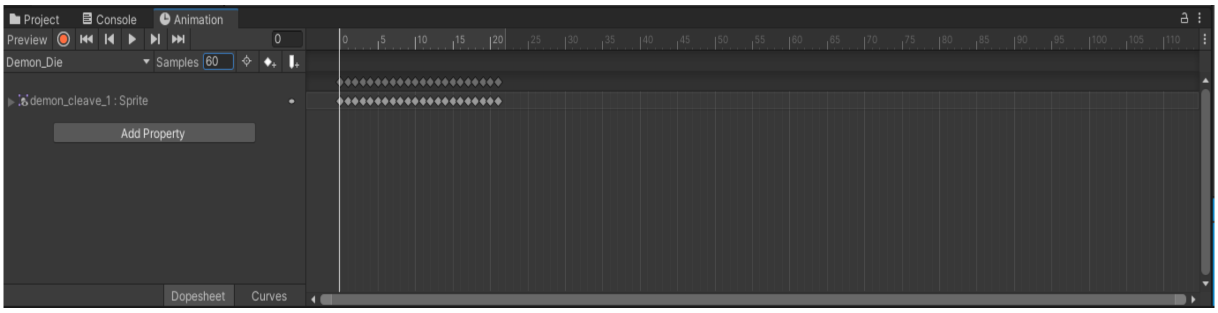


Figure 20: How an animation window looks if it has been filled with sprites.

Each Diamond represents a frame of the animation. The bar along the top of the window shows the length of the animation. Changing the number of samples in this window changes the length of the animation. We found that a Sample rate of 12 worked best for most pixel art animations. When searching for assets or designing them, its best to seek animations with more frames since you have more control over the sample rate. To add a new animation clip to an object select the drop down box containing the current animation and select create new clip.

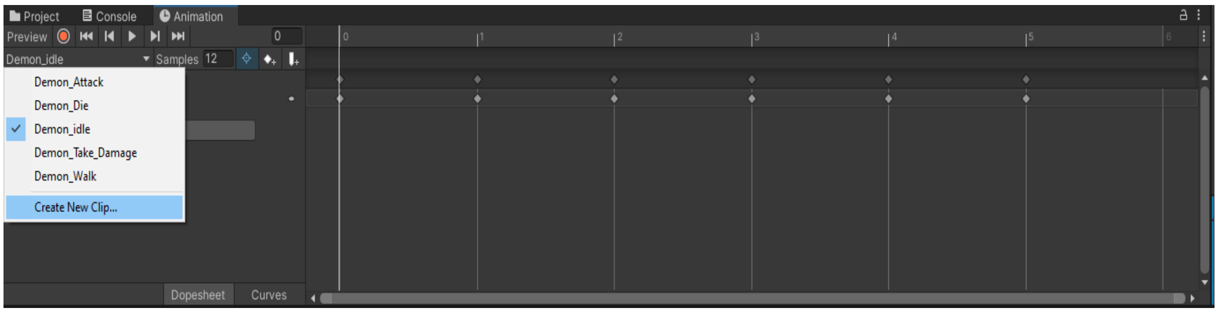


Figure 21: How an animation window looks if it has been filled with sprites.

The new clip should be placed in the folder containing the animation and should be named the same as the folder. Animations that should only be played once such as death animations should have looping mode turned off. All animations should have no exit time. This can be changed by selecting the animation clip and disabling Loop Time.

## 7.2 Adding a New Sprite

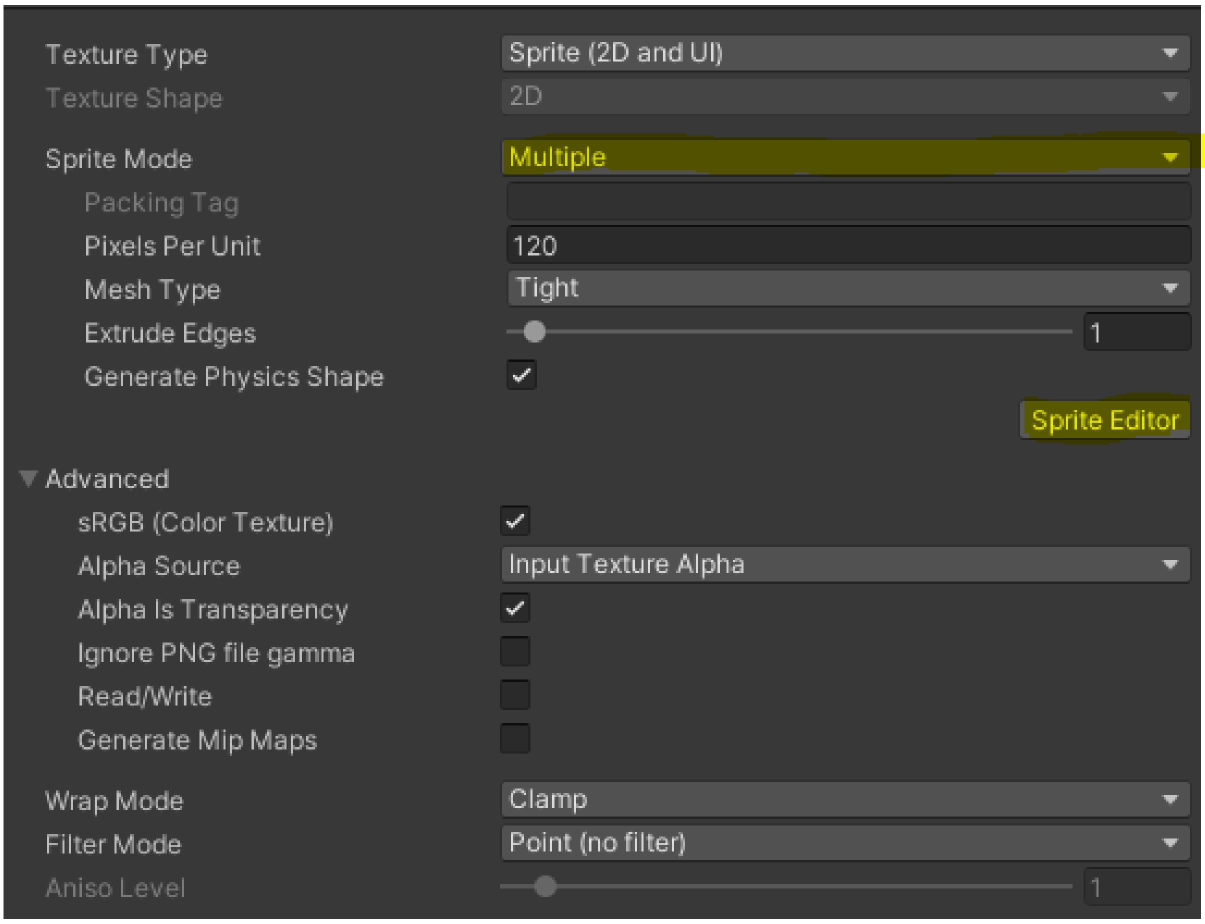
To add a new sprite to the game find or create the folder containing the feature you are adding the sprite for. For example to add a new mob, navigate to *Assets/NPC's/Hostile NPC's* and create a new folder called the mobs name. Inside this folder create a sprites folder and place the image there. Unity automatically adds a sprite renderer to the image when it is placed into file.

Since we are extensively using lighting and the Universal Render Pipeline in the project we must make it so that the sprite is effected by lighting. To do this we set the material to be sprite-lit-default. This is done by selecting the sprite to open the inspector for the object. In the inspector we select the material of the sprite renderer and then search for *sprite-lit-default*.
If the added sprite is pixel art we set the filter mode of the sprite to point no filter in the inspector. This removes post processing effect on the image making it much sharper, much more suitable for pixel-art aesthetic.

To scale the size of the sprite in game you can change the pixels per unit in the inspector. The pixels per unit determine how many pixels of the sprite correspond to one unit in the world. To make a sprite fill exactly one unit the pixels per unit should be set to the size of the image and the width and height of the image should be equal.

## 7.3 Dealing with Multi-Sprites in Unity

It is common for animations assets to be offered as a sprite sheet containing multiple sprites in one image. Unity has an inbuilt tool for dividing these sheets that will be explained here. Add the sprite sheet in the same way you would add a normal sprite. The process is detailed in the section Adding a New Sprite. In the inspector set the sprite mode to multiple and then select the sprite editor. From here select the slice tool at the top of the screen. Usually automatic is the best type to select. Occasionally you will have to manually set up where the sprite sheet will be sliced. The best way to do this is to set type to Grid by cell count and set the columns to the number of images in the sprite sheet. If the cells are positioned correctly click slice and apply. If it is not possible to slice the sprites evenly you may have to manually edit the position of each image using a image editing software such as Paint.Net or Photoshop.



Texture Type: Sprite (2D and UI)
Texture Shape: 2D
Sprite Mode: Multiple
Packing Tag:
Pixels Per Unit: 120
Mesh Type: Tight
Extrude Edges: 1
Generate Physics Shape: ✓
Sprite Editor
Advanced
sRGB (Color Texture): ✓
Alpha Source: Input Texture Alpha
Alpha Is Transparency: ✓
Ignore PNG file gamma:
Read/Write:
Generate Mip Maps:
Wrap Mode: Clamp
Filter Mode: Point (no filter)
Aniso Level: 1

Figure 22: A demonstration on opening the sprite editor.

## 7.4 Adding a New Mob

To add a new mob into the game you should create a new folder in *Assets/NPC's/Hostile NPC's* referencing the new mobs name. In this folder you should create a sub folder for sprites. In the sprites folder a new sub-folder should be made for each animation to be added. The animation folders should be called AnimationFirstName_AnimationSecondName. To create each animation you should follow the guide in section Adding an Animation Clip. The animations should then be linked together following Animation Controller.

Once you have an animated mob asset we can begin to implement its functionality. Firstly, We need to create a prefab for the mob. To do this drag the object into the scene. Firstly, a Rigidbody2D and a BoxCollider2D should be added to the mob prefab. This can be done by selecting the mob in the world and clicking add component in the inspector.

If the mob you are adding doesn't have any special behaviour and it is a ground mob you can attach the groundMobController Script located at Assets/NPC's/Hostile NPC's. You then need to add a reference to the walk and attack animations of your mob into the groundMobController script. This is done by clicking and dragging the animation clips into the inspector.
If there is a special behaviour involved in the mob then a new controller script will need to be created and attached to the object.

The mob will also need the enemy collision script attached to it, located at *Assets/NPC's/Hostile NPC's/EnemyCollision.cs*. Once the script is attached you need to set all variables in the inspector except the audio sources which are optional.

Finally the mob needs to be added to the spawning system. To do this copy your mob prefab. Next navigate to the mapSpawner prefab located at *Assets/Environment/Map Generation/Prefabs/Map Spawner.prefab*. Open the prefab and then add a new array entry into either the flying mob prefabs or the ground mob prefabs by clicking the '+' symbol.

### 7.5 Adding New Treasure

To add a new piece of treasure to the game navigate to *Assets/Environment/Map Design/Prefabs/Loot Prefabs/Treasure Prefab*. The easiest way to create a new treasure prefab is to duplicate an existing one, Change the sprite and change its treasure value. To do this select an existing prefab and click "Ctrl+D". Rename the newly created prefab by changing the value at the top of the inspector. If you want to create a different coloration of the existing gems select the color option in the sprite renderer. Choose the desired color and click apply. Alternatively, you can assign a new sprite in the sprite renderer to create a new type of object. If you wanted to add a more complex type of treasure such as a chest you would also have to create a controller script and add it to the prefab. For the chest example the control script may handle opening the chest when the player attack/interacts with it.

All treasure items must have an attached treasure script so that the player collision class can retrieve the value of the treasure when colliding with it. If you duplicated the object there will be one already attached to the prefab, simply change the treasure value to the desired amount. Finally, you need to add the treasure to the map spawner object. navigate to *Assets/Environment/Map Generation/Prefabs/Map Spawner.prefab* and select the map spawner prefab to open its inspector. Click and drag the treasure prefab you created into the Treasure Prefabs array of the world spawner script. The treasure is now fully added to the game.

## 8 Corrective Maintenance

Although we have been very strict about following good coding practices and adhering to our design stage, there are a few instances where our standards fell to try and produce a working increment quickly to demonstrate to the customer. This has proven to be short sighted and needs to be corrected before these classes can be expanded in the future. The most notable example of this is the player_Controller Class.

### 8.1 Player Controller

The player_controller class has far too many responsibilities and does not strictly adhere to the most updated version of the CRC cards created. The class contains over 900 lines of code and has simply become unmanageable. This class needs to be broken down into several smaller classes. Currently there are three classes responsible for controlling the player Player_Collisions, Player_Rope_Controller and Player_Controller. Below is a table demonstrating how we recommend the class be split up.

Another mistake we made was controlling all player animations in code. This was done to avoid developers unfamiliar with unity having to learn the animation controller tool and finite state machines. However, Due to the scale the project has grown to, we feel it is necessary to begin using this tool to separate animation logic from game logic. Therefore in the following subsection a demonstration of how to animate a simple character using Unity's animator is given. The process demonstrated here should be applied to the player controller before it is expanded any further. This section will go into further detail than previous scenarios since the process here is very unique to unity developers. Furthermore, it is important that this is done right since it is at the forefront of the user experience.

| **Scripts** | **Resposibility** |
|---|---|
| Player_Combat | Tracking attack combos, Sheathing and unsheathing weapons. Manage attack inputs. |
| Player_Ladder_Controller | Climbing up Ladders, Sliding down ladders, attaching to ladders. |
| Player_Rope_Controller | Climbing up ropes, Sliding down ropes, attaching to ropes. |
| Player_Damage | Receiving Damage from other objects, Setting player knocked down or stunned, Player dying |
| Player_Movement | Basic Movement, Jumping |
| Player_Advanced_Movement | Wall Jumping, Ledge Climbing, Wall sliding, Ground Sliding. |
| Player_Advanced_Inputs | Spawning Torch's, Interaction Button |

### 8.1.1 Animation Controller

An Animator Controller allows you to arrange and maintain a set of Animation Clips and associated Animation Transitions for a object. The tool is similar to finite state machines where we transition from one animation to a different animation when some game conditions occur. The easiest way to add an animation controller and an animator to an object is to open the animation menu. The default short cut to do this in unity is "Ctrl+6" alternatively you can select Window →Animation →Animation. If you select a game object that does not contain a animation controller the animation window will prompt you to create an animator and animation clip.

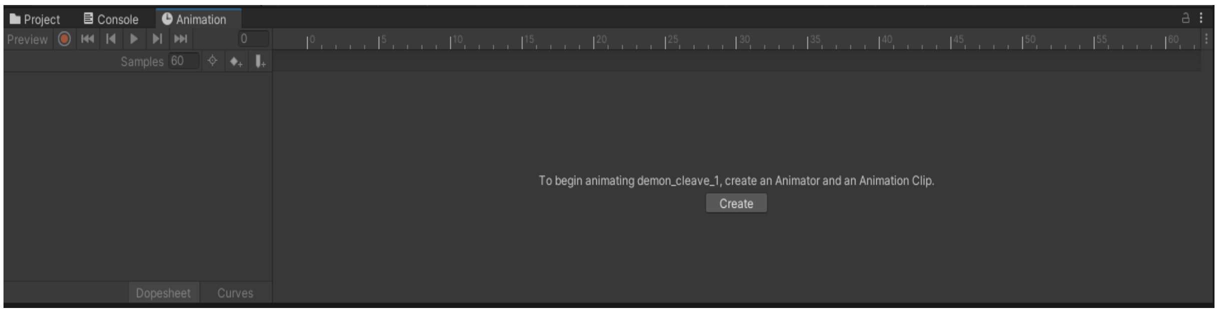

To begin animating demon_cleave_1, create an Animator and an Animation Clip.

Figure 23: How an animation window looks if you select a game object without an animator

**Demonstrating Animating the Demon Character:** Outlined is a demonstration of animating the demon character. Added an animation clip and animator follow the same methods above. Five animation clips have been to this object following the scenario demonstrated in the Adding an Animation Clip section. The animations added are attack, walk, idle, take damage and die. To view the animation window you need to select the toolbar and navigate to window →Animation →Animator.

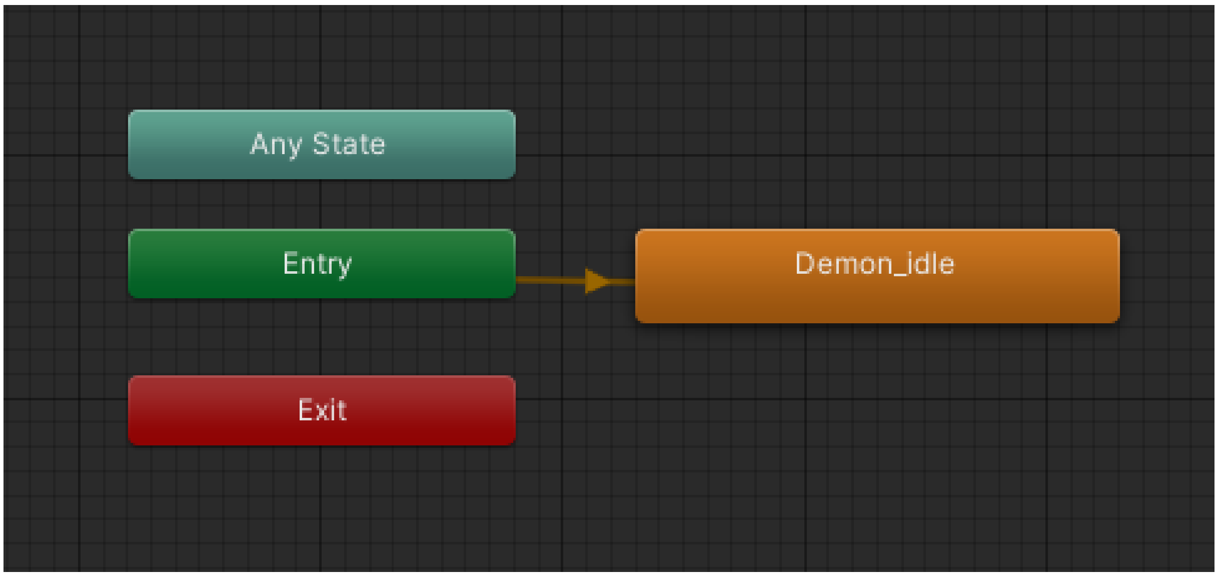


Figure 24: An example of the Animator window when first added to the Demon Object

To add an animation clip to the animator you need to click and drag the animation clip from the project window into the graph. the following graph shows the demon animator after all clips have been added. The orange state represents our default animation, it is what will begin playing when we start the game. The positioning of the clips on the graph has no impact on functionality but it is recommended you position them intuitively placing clips that will have transitions close together.

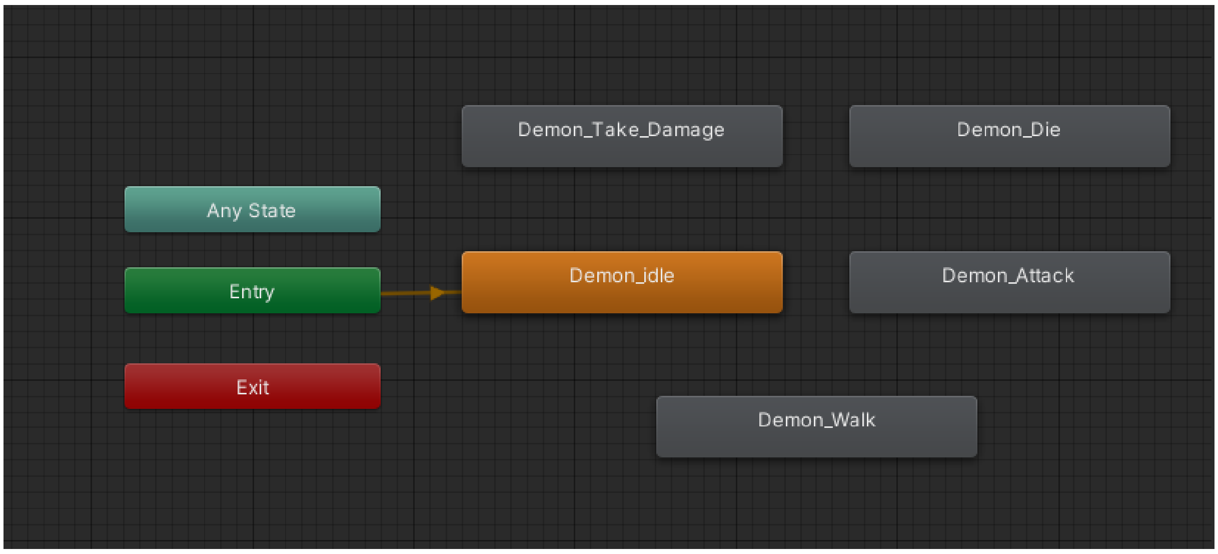


Figure 25: The Demon Animation controller example with all states added

We now need to start considering adding transitions between the states. We will begin by adding a transition from the Demon_idle state to the Demon_Walk State when the Demons speed is greater than zero. To make a transition right click a state and select *Make Transition*. This will create an arrow that you can attach to the state you want to transition to. We now need to define when this transition between states is made. But first we must add parameters to the animator so that it knows if we are moving. To add parameters select the parameters tab and click the + symbol. The following diagram highlights the position of this because it is quite well hidden.

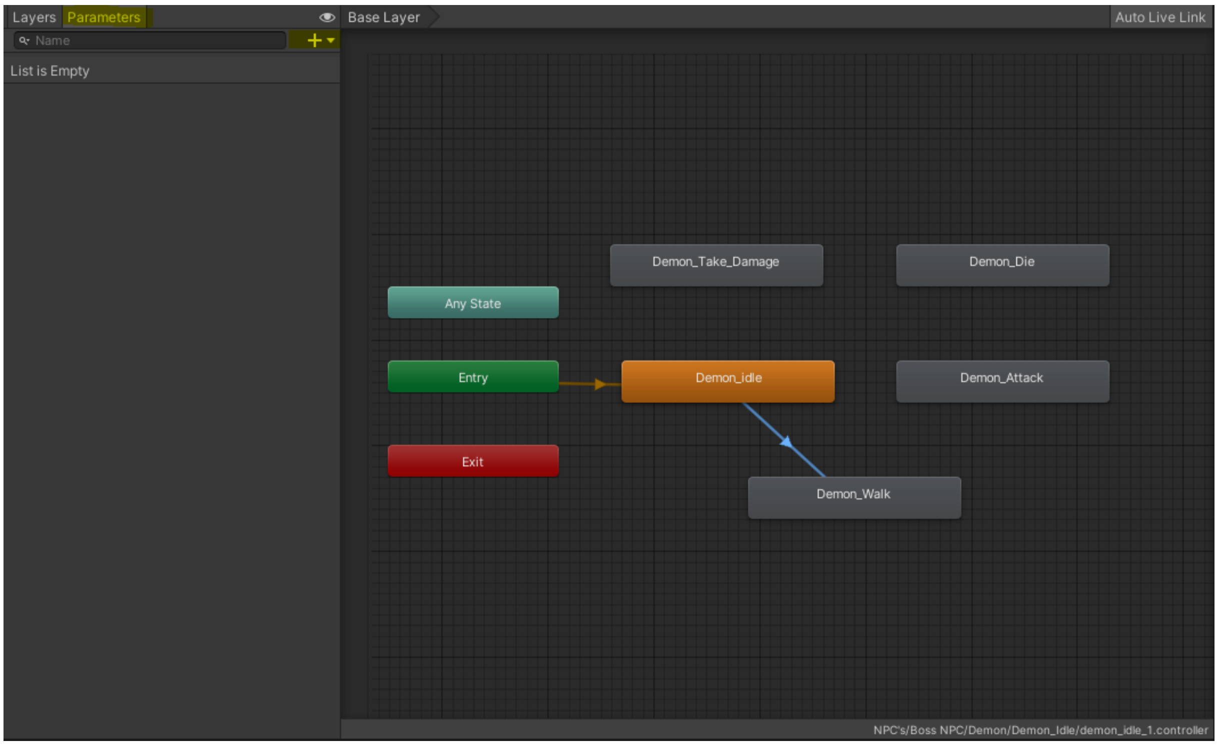
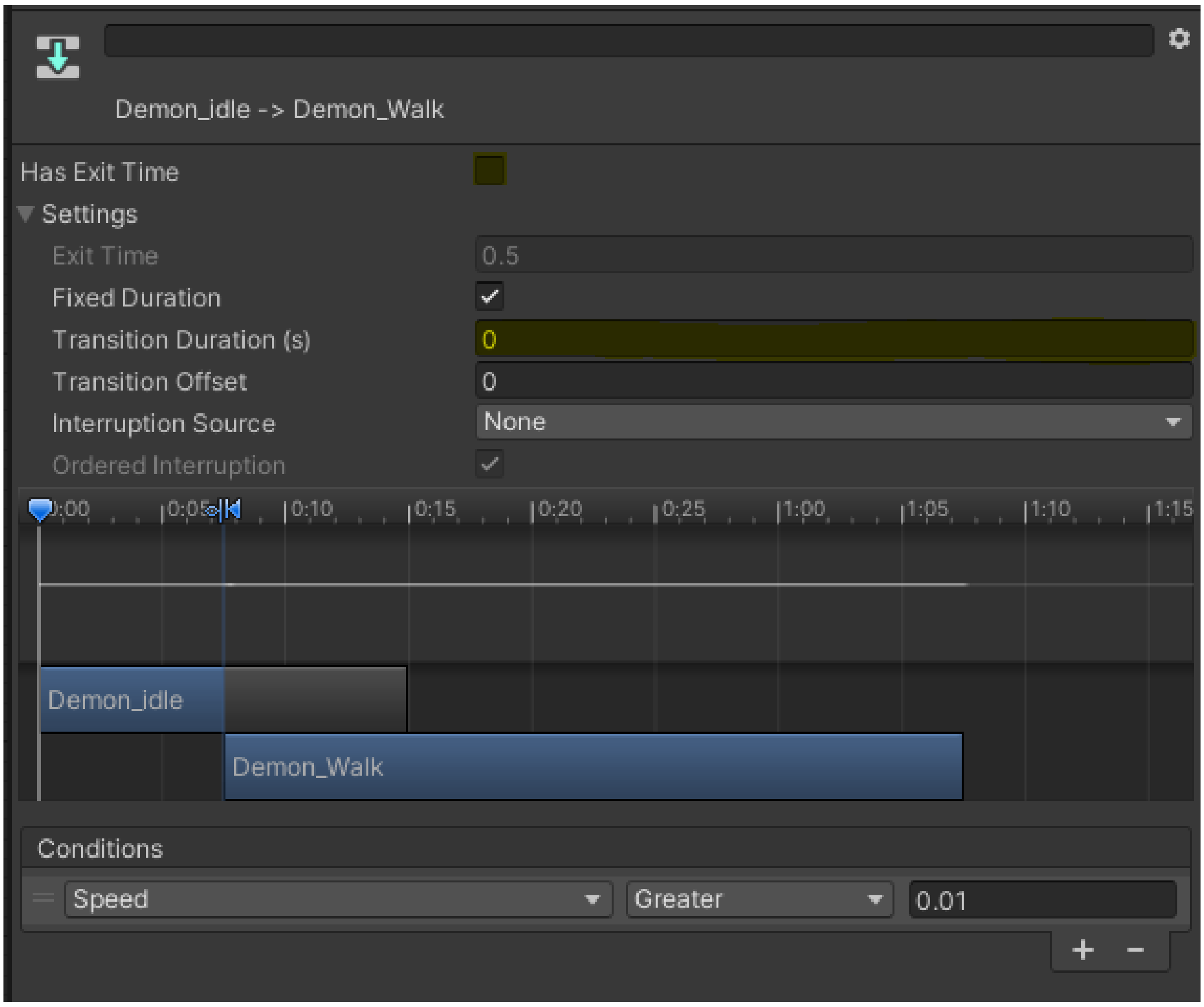


Figure 26: A Diagram highlighting where to add parameters to the animator

After selecting the + symbol we want to add a float ’Speed’ and set its value to 0f. You can change the name of a parameter by double clicking it.

We can now use the speed parameter to set the condition that we transition from idle state to walking state. To do this select the transition arrow that was created. This will show the details of the transition in the inspector window. Select the + symbol under the conditions tab. We want to set this transition to take place when the player begins moving so the condition will be set to ’Speed’ ’Greater’ and ’0.01’. We also need to disable exit time in this tab and expand the settings menu and set transition duration to 0. This ensures that the object switches animation immediately and will likely be consistent across all animations in our entire game.


Demon_idle -> Demon_Walk

Has Exit Time

Settings

Exit Time 0.5

Fixed Duration ✓

Transition Duration (s) 0

Transition Offset 0

Interruption Source None

Ordered Interruption ✓

Conditions

Speed Greater 0.01

Figure 27: A diagram demonstrating how to remove exit time from animations

We would also like to be able to transition back from the Demon_walk state to the being Demon_Idle state when the mob has finished moving. To do this follow the same steps as before but set the condition for the transition to be ”Speed”, ”less than”, ”0.01”. The animator will now transition between states whenever the condition is met automatically. To test this is functioning correctly you can dock the Animator window next to the running game. Then change the speed parameter in the animator window to 1. The mob should start playing the walk animation. If you set the speed back to 0 the mob should return to the idle animation. This workflow is especially helpful when working with large animation state machines such as the player controller.

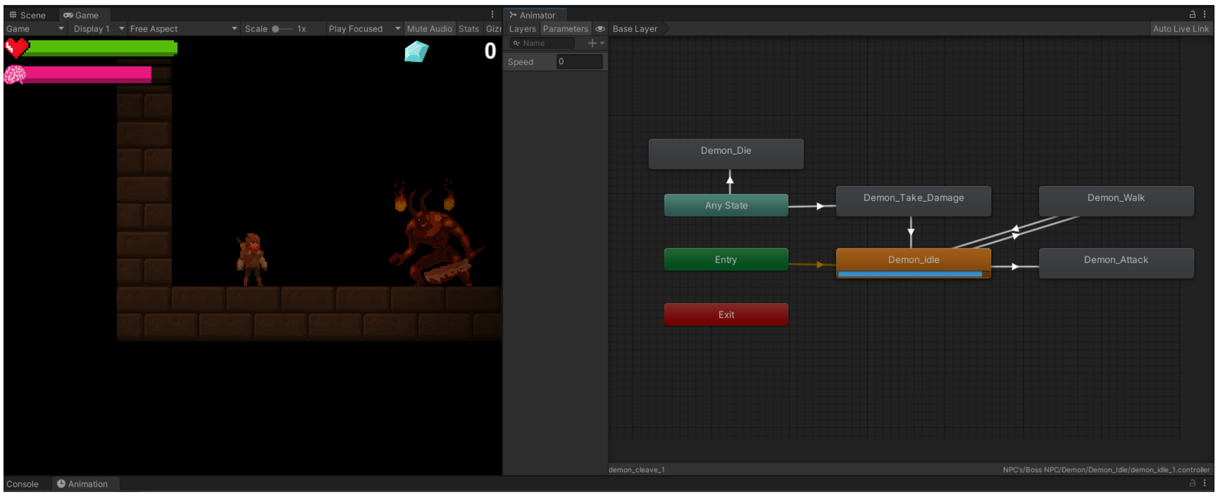


Figure 28: An example workflow to debug the animator. On the left is the running game, On the right is the animator. By varying the parameters in the middle you can see the effect they have on animation transitions live without writing any code

Now we have confirmed the animation controller is working, we need to update the speed parameters in the controller script for this object. To do this we need to store a reference to the objects animator in the control script. This is done by adding a '[Serialized Field] private Animator animator' variable to the control script. To add the reference to this variable you need to select the object so that you can view its inspector. Click and drag the animator component into the newly created slot in the controller script. using this reference we can update the parameter speed in the control script by adding animator.SetFloat("Speed", currentSpeed). The animator only accepts positive values so we need to ensure that the currentSpeed is positive using Mathf.Abs(currentSpeed). We can repeat this process for every state that needs interactions until the state machine is complete.

There will be some states that we want to transition into from any other state. In our example we will want to transition into the Demon_Die state as soon as the mobs hp reaches 0 regardless of the current state of the object. To achieve this we create a transition from the blue Any State node into the Demon_Die state.

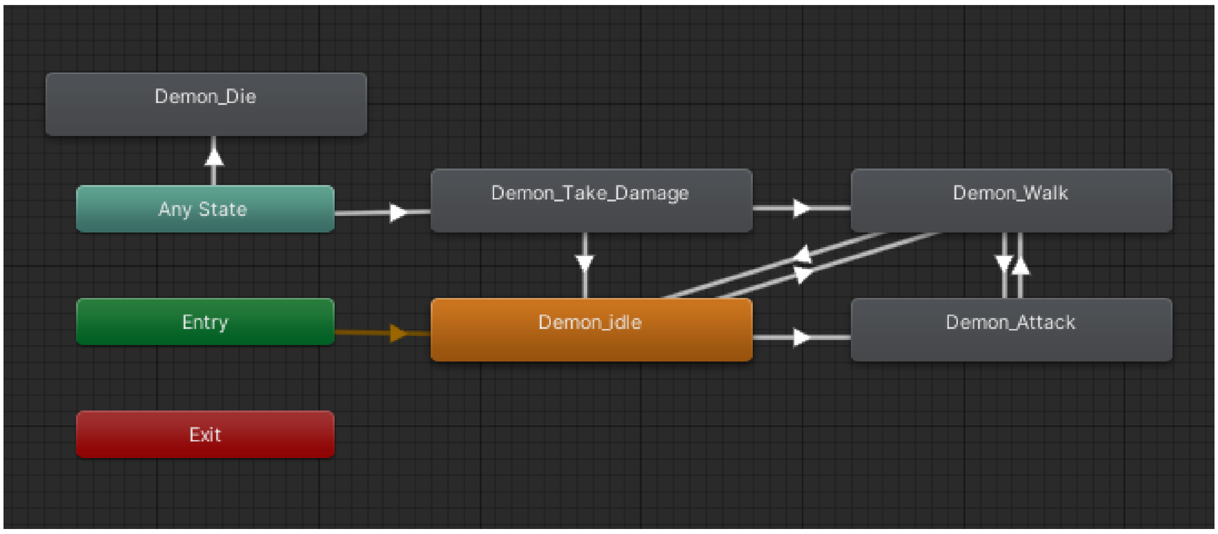


Figure 29: The completed animator used in the Demon demonstration.

# 9 Future Expansions

## 9.1 Unused Animations

There are several unused animated assets included in our project which may help inspire future directions for the project. All animations have been fully converted into animation clips. To add these to the project you simple need to create a script which controls the spawning/behaviour of the asset. The following tables show each unused asset's base sprite, Includes a list of animations available to the asset and suggests the intended use for the asset in future expansions. The file location of the asset is displayed in the first row of the table.

| Assets/NPC's/Hostile NPC's/Reaper | | |
|---|---|---|
| Reaper | Unused Animations | Potential use |
| | • Reaper_Appear<br>• Reaper_Attack<br>• Reaper_Dissapear<br>• Reaper_Idle_1<br>• Reaper_Idle_2<br>• Reaper_Summon<br>• Reaper_Summon_appear<br>• Reaper_Summon_Death<br>• Reaper_Summon_Idle | The original plan for this asset was to fade the mob in using the Reaper_appear animation when the player is on low instantity. Once the mod was faded in it would then use the Reaper_Attack Animation before dissapearing using the Reaper_Dissapear animation. This would apply extra pressure for the player to reach the exit when on low instanity. |

Table 3: Reaper Mob, All unused animations

| Assets/NPC's/Hostile NPC's/Archer | | |
|---|---|---|
| Archer | Unused Animations | Potential use |
| | • Archer_Duck<br>• Archer_Shoot<br>• Archer_Die<br>• Archer_Roll<br>• Archer_Idle<br>• Archer_Run | A ranged mob that can be added to the mob spawning algorithm. The firing functionality could use the scripts created for the arrow projectile spawned by the arrow trap for a quick implementation.<br>The control script included in the file contains the basic functionality for firing at a player when they are in line of sight. |

Table 4: Archer Mob, All unused animations

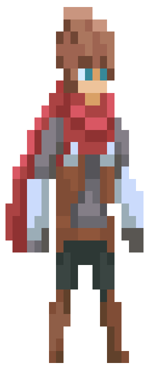
| Assets/Player/Sprites | | |
|---|---|---|
| Player | Unused Animations | Potential use |
|  | • Player_Bow_Jump<br>• Player_Bow_Shoot<br>• Player_Spell_Cast<br>• Player_Spell_Channel<br>• Player_Ledge_Climb<br>• Player_Ledge_Grab<br>• Player_Swim_Idle<br>• Player_Swim | The bow and spell animations are included with the intention of adding treasure chests around the map which would drop loot such as a bow or spell book.<br>The Ledge climb was an advanced player movement item included in the backlog. The function that tests if a player is able to ledge climb is included in the player_controller class but the functionality was not implemented further.<br>Swimming was an optimistic inclusion in case we ever got fluid simulation working as intended. |

Table 5: Player Character, All unused animations

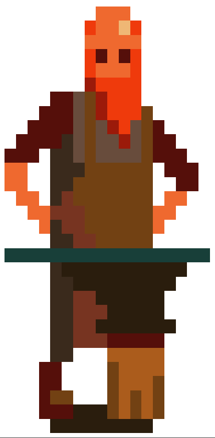
| Assets/NPC's/Friendly NPC/Black Smith | | |
|---|---|---|
| Blacksmith | Unused Animations | Potential use |
|  | • Blacksmith_Bored_1.<br>• Blacksmith_Bored_2.<br>• Blacksmith_Idle.<br>• Blacksmith_Greeting.<br>• Blacksmith_Work_1.<br>• Blacksmith_Work_2. | The intention of this asset was to add a shop room to every dungeon level. The player could invest their treasure to buy items that would make them stronger. There is currently no preliminary code created for this asset. |

Table 6: Blacksmith, All unused animations

| Assets/NPC's/Boss NPC/Demon | | |
|---|---|---|
| Demon | Unused Animations | Potential use |
| | • Demon_Attack<br>• Demon_Walk<br>• Demon_Die<br>• Demon_Take_Damage<br>• Demon_Idle | The intention for this asset was to use it as a boss once the player reaches the end of the dungeon. Since this is no longer necessary this could be added to the mob spawning system and the Ground_Mob_Controller can be added to control its behaviour. This could be a very quick way to add another mob to the game. |

Table 7: Demon Boss, All unused animations

## 9.2 Accounting for the Leader-board's Website Failure

We encountered an issue in our final week where the website that stores the database for our leader-board was unavailable. This bug should be fixed with the highest priority since it provides a very poor user experience.

A potential solution is to catch the error then store the results of the run we are trying to submit to the database locally. we could then try write a script which submits all local results the next time the game is run. We could also store the last known leader-board results locally so that the player is never aware of an issue. This would involve setting up persistent memory.

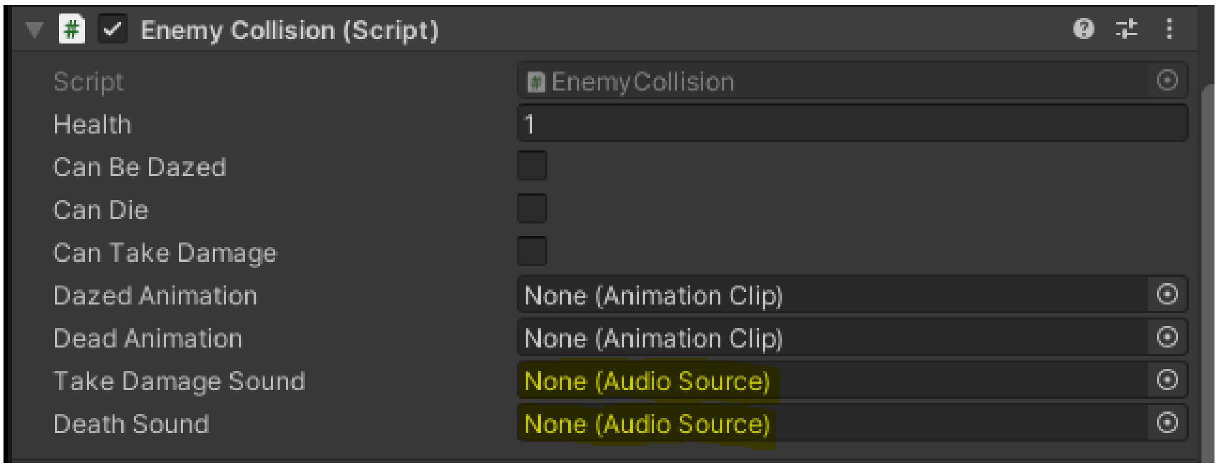
## 9.3 Adjusting the Room Spawning Algorithm

Currently rooms all connect along the center of their border. This is a less than ideal solution as room designers need to be explained the details of this and be mindful of this fact when designing new rooms. This significantly reduces the amount of creativity that can be applied when designing rooms. Room connections objects have been added to each room in the position where they have an opening. The room spawning algorithm in the world spawner script needs to be updated to check if rooms connections correctly overlap. If they do not a new room needs to be spawned until room connection is successful. An initial idea to implement this functionality is to place an overlap area across the borders of two rooms and see if they are two room connectors in any given slot. The world spawner script can be found at Assets/Environment/Map Design/Scripts/WorldSpawner.cs

## 9.4 Adding Game Audio

We decided at the beginning of the project that we would include audio for all interactions in the game. However, functionality was always prioritised over quality of life improvements and audio was never implemented. As we were expecting to include sound effects, audio sources are present and linked with functionality in most places of the project. As a result implementing an audio system would be a low-development cost with high user-experience reward.

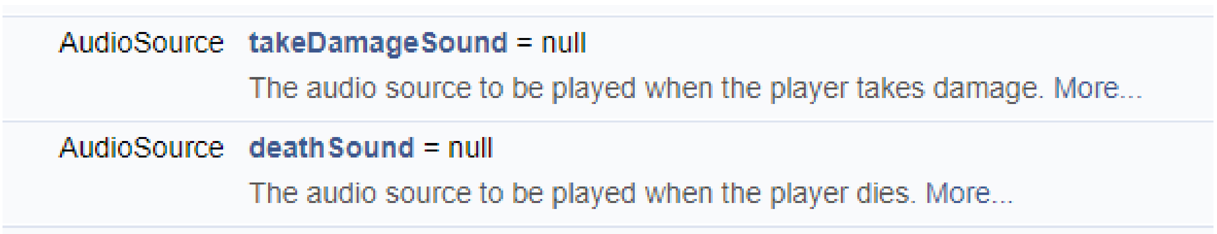
Audio sources simply need to be found or created and then added to the corresponding scripts. Audio assets should be placed in the sound folder within a feature. To check if audio functionality has already been implemented for a given asset you need to select it in the hierarchy. From there you can view each of the attached scripts in the inspector to see if it contains an empty audio source. Alternatively you could open the Doxygen documention for a script and search for audio sources there.



Enemy Collision (Script)

|  |  |
|---|---|
| Script | EnemyCollision |
| Health | 1 |
| Can Be Dazed |  |
| Can Die |  |
| Can Take Damage |  |
| Dazed Animation | None (Animation Clip) |
| Dead Animation | None (Animation Clip) |
| Take Damage Sound | None (Audio Source) |
| Death Sound | None (Audio Source) |

Figure 30: A Diagram showing where audio sources should be added in the inspector



AudioSource **takeDamageSound** = null
The audio source to be played when the player takes damage. More...

AudioSource **deathSound** = null
The audio source to be played when the player dies. More...

Figure 31: Audio sources found in the Doxygen documentation

## 10 Conclusion

As a final note, part of the maintenance of this project involves updating this document frequently; any changes in project structure, design philosophy or tools used should be reflected clearly in this document.

# Endless Delve - User Installation Guide

December 2021

## Contents

## 1 Introduction

This installation guide assumes that the user is familiar with downloading content from a website, navigating to the folder where the content has been downloaded and either running an executable file, or running an installer. If you can't do these things, find someone who can! Additionally, the user must ensure the file for the correct platform is downloaded. Three different target platforms exist:

1. Windows
2. MacOS
3. Linux

The user is responsible for making sure the files downloaded are compatible with, and run on the operating system of their own machine.

This document contains the system requirements to run the game, followed by the installations for the different operating systems.

## 2 System Requirements

### 2.1 Unity Recommended Requirements

Please note that the system requirements specified are dictated by what Unity (the underlying game engine used to create this game) suggests be the minimum system requirements. As this game is very lightweight, it is possible you can run it on a slower system, but this is not guaranteed! Also, ensure that your system has **at least half a gigabyte (0.5 GB) of free storage space space** in your computer.

| **Operating system** | **Windows** |
|---|---|
| **Operating system version** | Windows 7 (SP1+), Windows 10 and Windows 11 |
| **CPU** | x86, x64 architecture with SSE2 instruction set support. |
| **Graphics API** | DX10, DX11, DX12 capable. |
| **Additional requirements** | Hardware vendor officially supported drivers.<br>For development: IL2CPP scripting backend requires Visual Studio 2015 with C++ Tools component or later and Windows 10+ SDK. |

| **Operating system** | **macOS** |
|---|---|
| **Operating system version** | High Sierra 10.13+ |
| **CPU** | Apple Silicon, x64 architecture with SSE2. |
| **Graphics API** | Metal capable Intel and AMD GPUs |
| **Additional requirements** | Apple officially supported drivers.<br>For development: IL2CPP scripting backend requires Xcode. Targeting Apple Silicon with IL2CPP scripting backend requires macOS Catalina 10.15.4 and Xcode 12.2 or newer. |

| **Operating system** | **Linux** |
|---|---|
| **Operating system version** | Ubuntu 20.04, Ubuntu 18.04, and CentOS 7 |
| **CPU** | x64 architecture with SSE2 instruction set support. |
| **Graphics API** | OpenGL 3.2+, Vulkan capable. |
| **Additional requirements** | Gnome desktop environment running on top of X11 windowing system<br>Other configuration and user environment as provided stock with the supported distribution (such as Kernel or Compositor)<br>Nvidia and AMD GPUs using Nvidia official proprietary graphics driver or AMD Mesa graphics driver. |

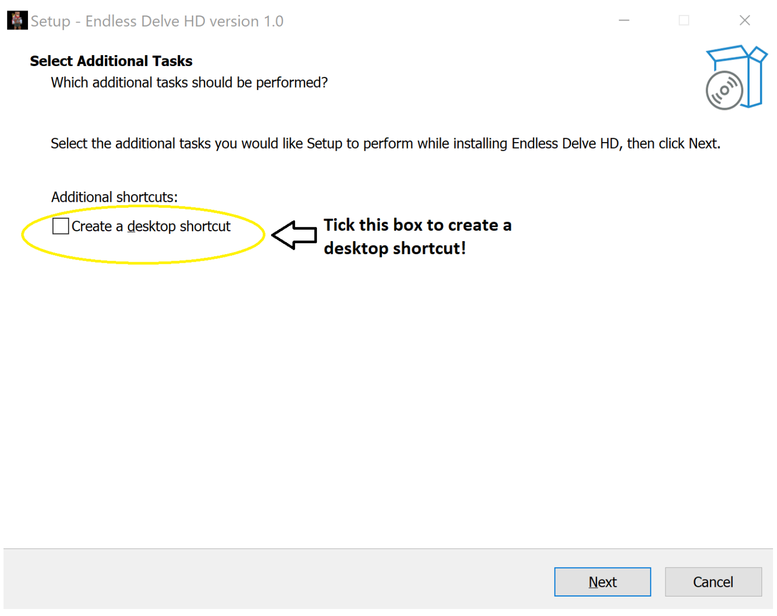
### 2.2 Resource Consumption

Resource consumption was monitored on a Windows machine running. This game uses at most 200 Megabytes (MB) of memory (RAM), and around 2% of the CPU on a modern laptop (2.6 GHz clock speed, 6 cores). It can comfortably run on an integrated graphics chip, so no need to worry if you don't have a GPU! Also, this means, if your machine does not match the minimum requirements, there is a good chance it may still run! Give it a try, and if it doesn't, refunds are available!

# 3 Installations

## 3.1 Windows Installation

Windows installation is done through an installer, which can be downloaded from:
`https://ufile.io/8fxion74`
The installation process is very standard, but pictures have been attached to guide you through the process! Download the setup, and run it. The User Account Control System will ask you to give authorisation for the installer to go ahead. Select yes, and proceed to the actual setup.


Setup - Endless Delve HD version 1.0

**Select Additional Tasks**
Which additional tasks should be performed?

Select the additional tasks you would like Setup to perform while installing Endless Delve HD, then click Next.

Additional shortcuts:
☐ Create a desktop shortcut

**Tick this box to create a desktop shortcut!**

Figure 1: Create a desktop shortcut!

Figure 1 above shows the box that must be ticked if you would like a desktop shortcut. This is suggested, as it will allow you to launch the game from your desktop, and you can delve deeper in Endless Delve on a whim!

Following this, click Next, and on the final page hit Install, on the bottom right! Your game will install, and you are now ready to play! If you wish to play immediately, just leave ”Launch Endless Delve HD” ticked, and close the window by selecting ”Finish”.

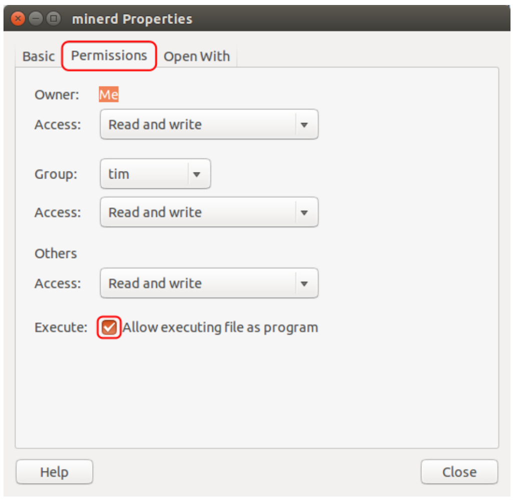
Have fun!

If you ever wish to uninstall, you can do this by navigating to ”Control Panel” from the Windows icon on the bottom left of your screen, and selecting ”Uninstall a program”. From here, simply search for Endless Delve HD and select uninstall.

## 3.2 Linux Installation

The Linux build can be downloaded from:
`https://ufile.io/upaqfnmg`
The file downloaded is in a .Zip, but you can automatically extract this in Linux. To extract the .Zip file into your current directory after downloading the game, simply right click it and select ”Extract Here”. Once the folder has been extracted, navigate to it and open it. Here, look for a file called `EndlessDelve.x86_64` and right click it. Select properties, and in the properties window, navigate to

permissions, as depicted in Figure 2. In the permissions tab, you must tick the box next to Executre, stating ”Allow executing file as program”. After this, simply double click the file `EndlessDelve.x86_64` and you can begin playing!



Owner: Me
Access: Read and write
Group: tim
Access: Read and write
Others
Access: Read and write
Execute: [✓ Allow executing file as program]

Figure 2: Allowing File To Be Executed As A Program On Linux

### 3.3 Mac Installation

The MacOS installation is done through the use of a `dmg` file, which can be downloaded from:
`https://ufile.io/8q3mhzfp`
After downloading the `dmg` file, double-click it. This will take a while (your computer will decompress the file etc.), but eventually open a folder. In this folder, there will be an Application File with the name: `EndlessDelve`. Double click this Application File, and you are now able to play Endless Delve!

## 4 Troubleshooting

### 4.1 Windows Installer and Anti-Virus Problems

Rest assured the file is safe and virus free, Your operating system simply doesn’t recognise the source so is being cautious. To fix this you can add it to the exclusions tab of windows security.
To do this Go to Start >Settings >Update  Security >Windows Security >Virus & threat protection.

Under Virus  threat protection settings, select Manage settings, and then under Exclusions, select Add or remove exclusions.

Select Add an exclusion, and then select file. From here you can navigate through file explorer and select EndlessDelveHD.exe.

You should now be all set to install the game by running EndlessDelveHD.exe. You can now enjoy countless hours delving through dungeons. Good Luck!

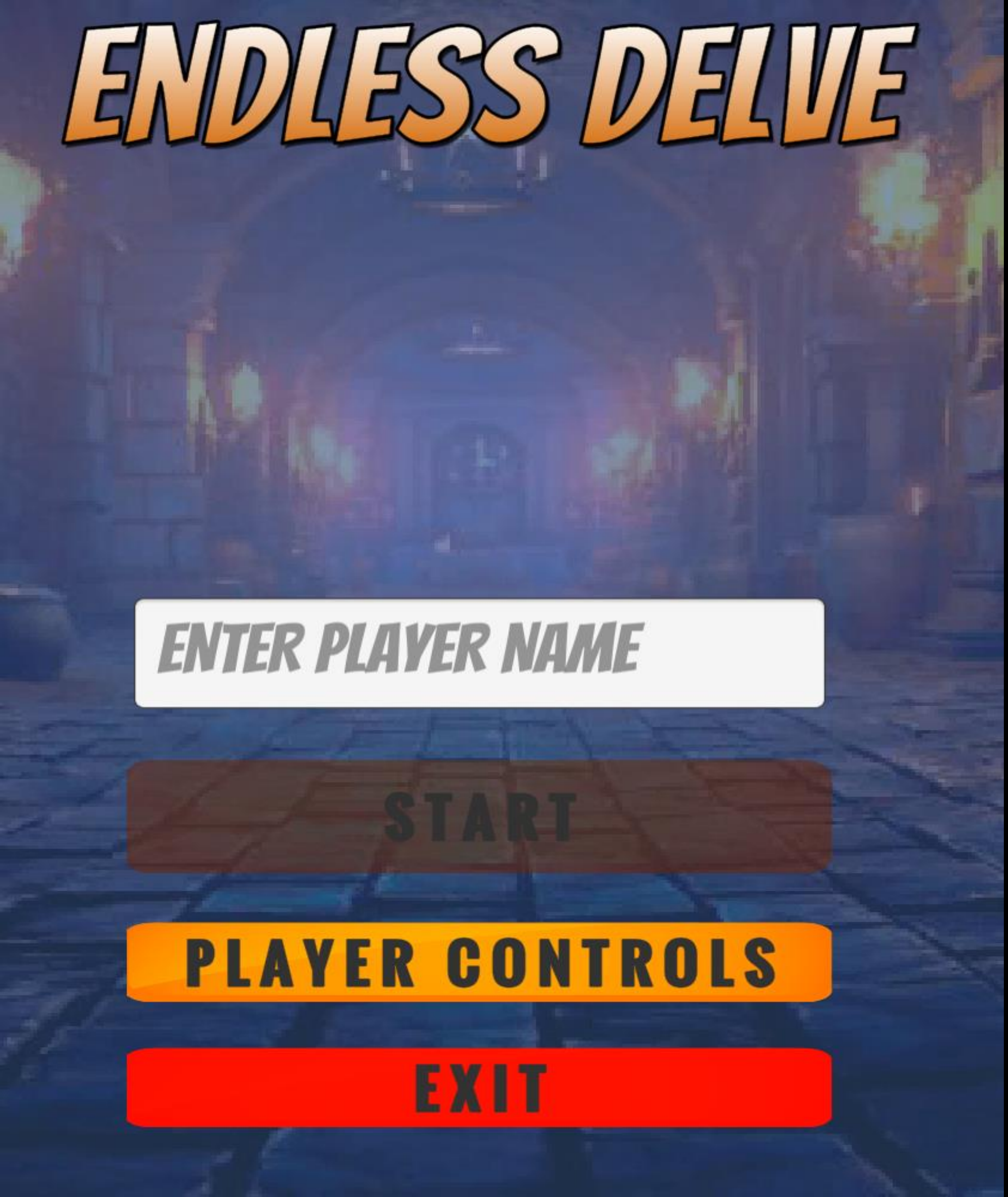
# ENDLESS DELVE

User Manual

## Object of the Game

You take on the role of a treasure hunter, at the beginning of your latest adventure. You have been exploring ancient ruins, rumoured to be the site of a long-forgotten tomb filled with gems that could make you rich. As you're about to give up, you spot a staircase descending into darkness. Excitedly you check your torches, and follow the worked stone path downwards. You reach a door of solid oak, flanked by two braziers. As you press your ear to the door, you can make out a far-away susurration, and realise you're not alone. Steadfast, you open the doors and light the braziers on the other side.

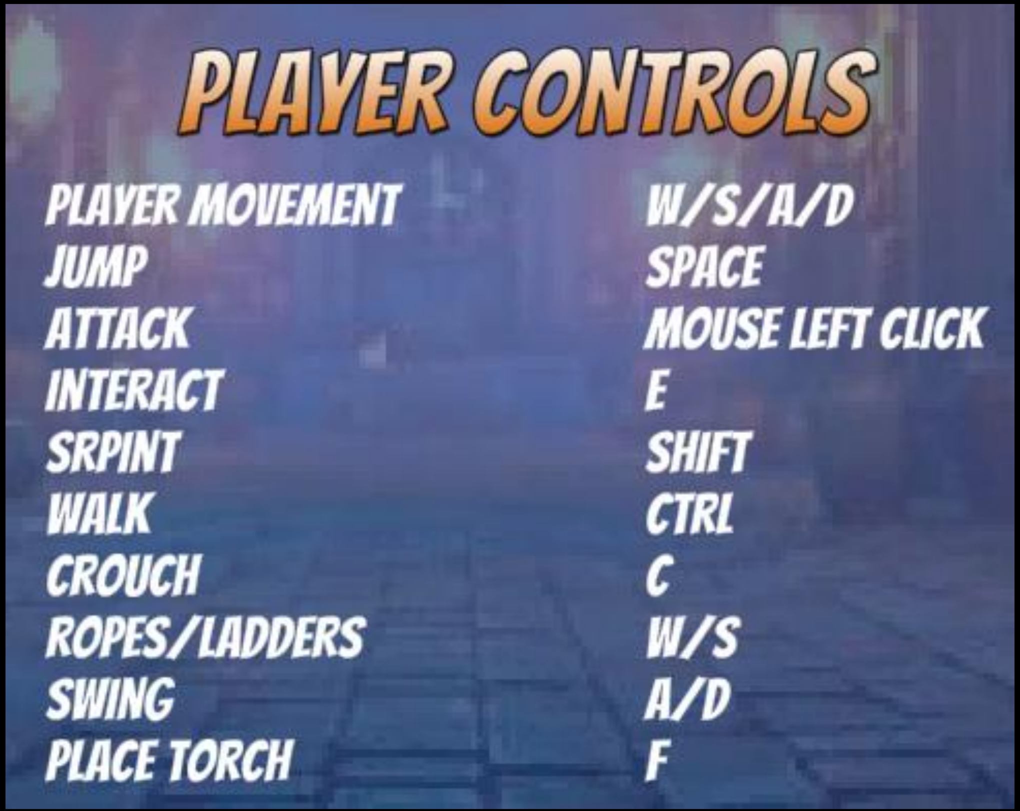
The game begins at the entrance to the dungeon. The aim is to delve as deeply as possible, filling your pockets with as much treasure as you can along the way. But be careful… the deeper you go, the more dangerous its inhabitants become and the greater the toll on your sanity. Collect as much loot as possible before you succumb to the darkness or perish at the hands of the many dangers lurking in the depths.



## Menu

Pictured to the left is the main menu. This is the first screen you are greeted with when you enter Endless Delve. The main menu has 4 options.

- **Enter Player Name** – Here you can type in your name. This is the name that will be displayed on the leader board if you score highly enough to make it in to the top 10.
- **Start** – This is the button you click to enter into the game.
- **Player Controls** – This button displays a full list of the player controls.
- **Exit** – This button exits the game.



## PLAYER CONTROLS

| | |
|---|---|
| PLAYER MOVEMENT | W/S/A/D |
| JUMP | SPACE |
| ATTACK | MOUSE LEFT CLICK |
| INTERACT | E |
| SRPINT | SHIFT |
| WALK | CTRL |
| CROUCH | C |
| ROPES/LADDERS | W/S |
| SWING | A/D |
| PLACE TORCH | F |

**W** - Moves your character upwards. This is used to climb higher on a rope.

**A** - Moves your character to the left. Use this to navigate your way around the dungeon, and to swing to the left when on a rope.

**S** - Moves your character downwards. This is used to climb down on a rope, or speed up a wall slide descent

**D** - Moves your character to the right. Use this to navigate your way around the dungeon, and to swing to the right when on a rope.

**Space** - Allows your character to jump. Use this to jump on to ropes, avoid traps or enemies, and to start a wall ride.

**Left Mouse Click** - Clicking this makes your character attack. Use this to defeat enemies you encounter

**E** - Press this to use the exit. Get to the next level once you've collected everything in this one!

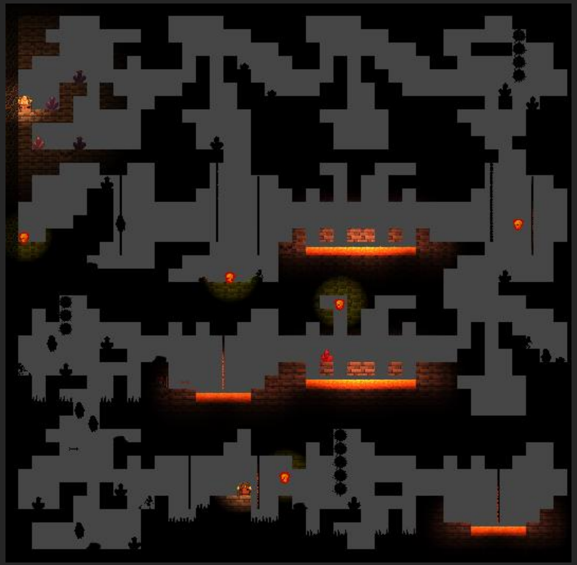
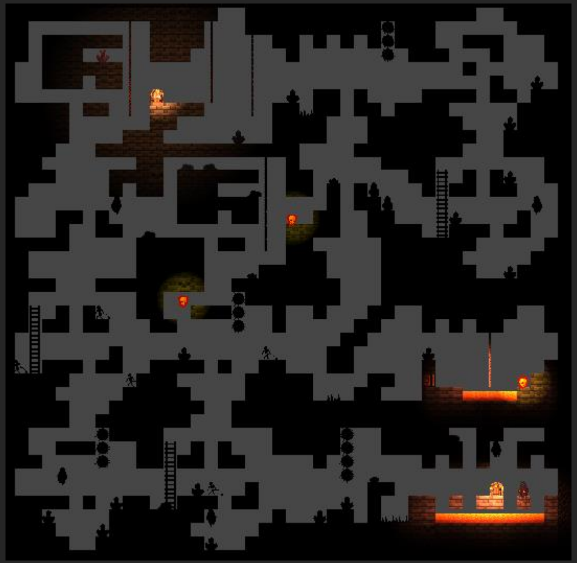
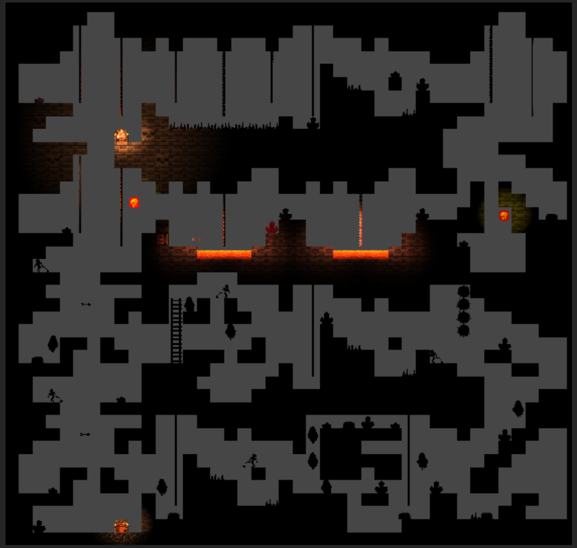
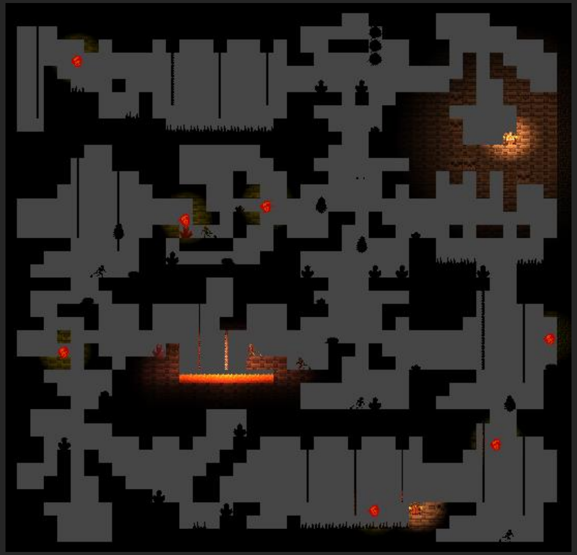
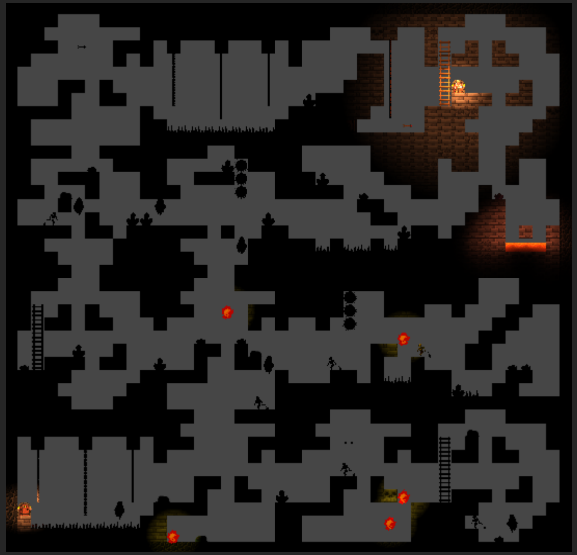
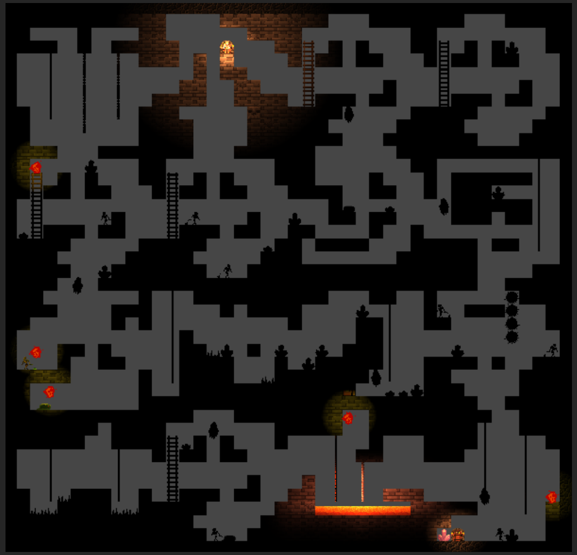
**Shift** - Hold down to sprint. Use this to evade enemies, or move quickly through the dungeon

**Ctrl** - Pressing this makes your character attack. Use this to defeat enemies you encounter

**C** - Hold down to crouch. Use this to move through tight spaces, or duck under attacks

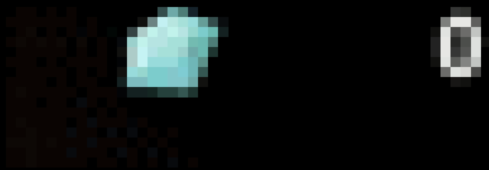
**F** - Press this to place a torch. Use this to light up the area around you.

# The Randomly Generated World

# The Game Mechanics

After clicking start, you spawn in at the entranceway to the dungeon in a random room on the top row of the map. You must navigate downwards through the maze of deadly rooms to reach the exit door placed in a random room on the bottom row. Once you exit, you will move on to the next level. Keep going, until you eventually die!



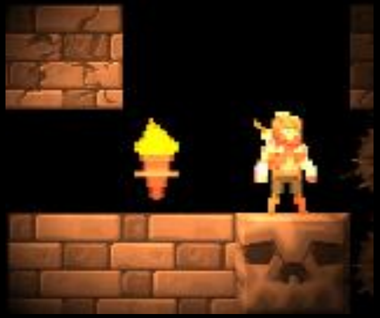
Along the way you will be gathering gems, which determine your total score. Your score is displayed in the top right-hand corner of the screen, next to the blue gem symbol.

But remember to be quick! Your sanity will drain as you spend time in the dungeon. This is displayed in the top left hand corner. As your sanity dwindles, so too does the light surrounding you. Once the bar is empty, you will start to take damage.

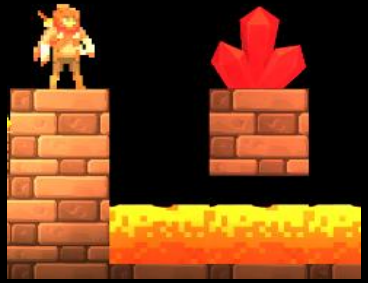
The health bar is also displayed in the top left-hand corner of the screen. This will decrease as you take damage from enemies, traps, and an empty sanity bar. Once this reaches 0, the game is over.

You will encounter enemies during your adventure. If you can't evade them, use punch and kick attacks to defeat them.

# In Game Objects



As your sanity drops and the light fades, you may find yourself wishing that you were able to see a little better. Place a torch down to illuminate the area around you. But use them wisely, you only had space to bring 3 along with you.



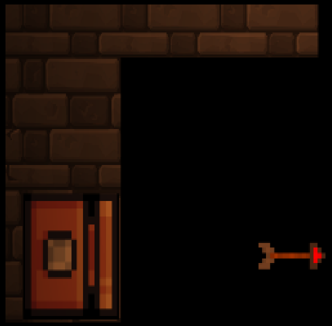
Gems like this one can be found throughout the level. Simply run in to them to collect them and increase your score! Small Purple gems are worth 100, large purple gems are worth 500, and large red gems are worth 1000 points. Pay particular attention for the illusive gold gem….

Rope swings can also be found throughout the dungeon. You can jump on to them, and use them to swing over lava pits, enemies, or to reach different areas of the map. You can also climb up and down the ropes.

# Traps

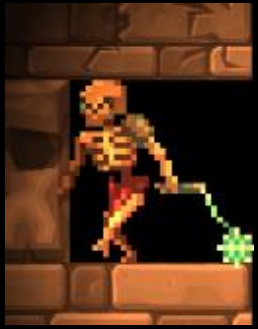
There are 3 varieties of traps that you'll have to avoid in the dungeon. The deadliest of these is lava. If you fall into this, there are no second chances. Instant death awaits you, so avoid it at all costs.

Spikes will continue to damage you for as long as you're in contact with them. If you do find yourself falling towards one, jump off as quickly as possible!



Arrow traps fire arrows out horizontally at regular intervals, and damage you on impact. Time your move well, and you'll have no trouble avoiding this devilish contraption.

# Enemies



Skeletons patrol the pathways, wielding a deadly Morningstar. If they see you, they'll swing at you and deal significant damage, knocking you back.

Flaming skulls float around the dungeon, and will pursue you if you get too close. If they manage to touch you, they will explode on impact. This will deal damage, and knock you back. Better hope there's no lava behind you!

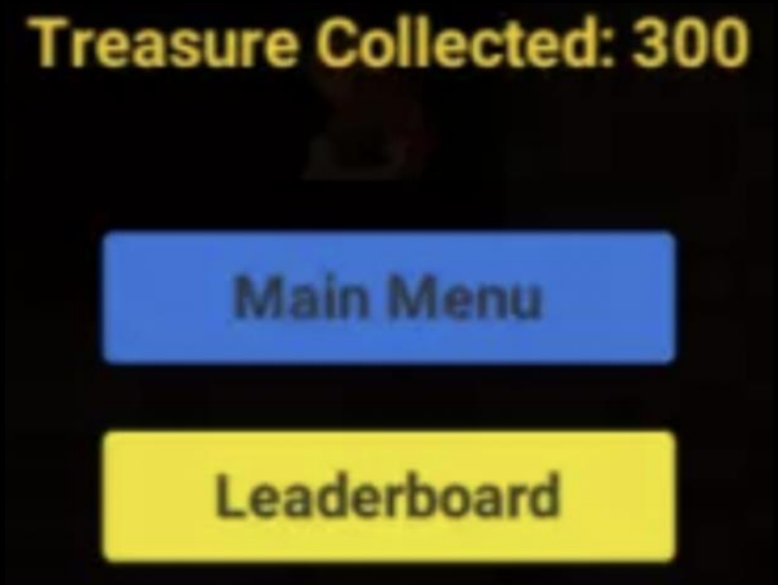
Witches may appear passive when you first see them, but don't let this fool you. If you turn your back on them they'll glow green and chase you down. You can prevent this by turning to face them again. Just don't let them reach you!

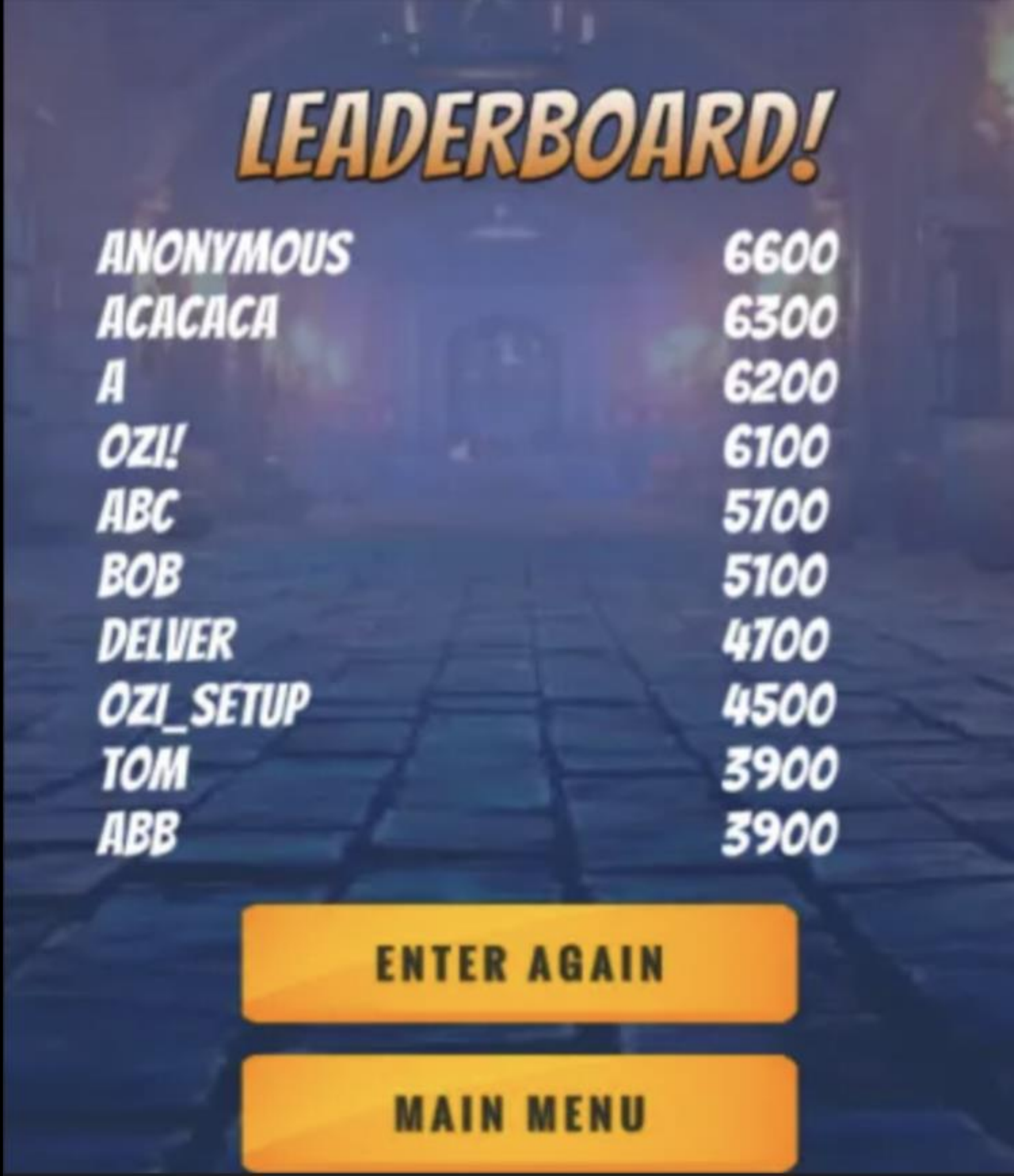
The slime residing down here doesn't often get to eat adventurers, and you'll find they have quite the appetite… don't get too close, or they'll try to devour you!

## Ending the Game

Try as you might, you will eventually die in the tomb, unable to escape with your gems. When this happens, a "GAME OVER" screen will appear, and reveal your final score. From here, you can go to the main menu, or view the leader board. This screen is pictured below.

The leader board is pictured to the right. If you were in the top 10, your name will appear in the list! Congratulations!


Treasure Collected: 300


LEADERBOARD!

| | |
|---|---|
| ANONYMOUS | 6600 |
| ACACACA | 6300 |
| A | 6200 |
| OZI! | 6100 |
| ABC | 5700 |
| BOB | 5100 |
| DELVER | 4700 |
| OZI_SETUP | 4500 |
| TOM | 3900 |
| ABB | 3900 |