Practical Gaming 2022

# Name of Student Hyden Johnson

# T Number T00220182

# Name of Project Raid of the Vault

# Gameplay

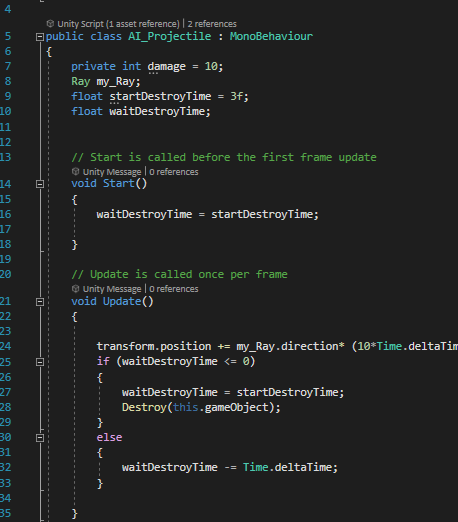
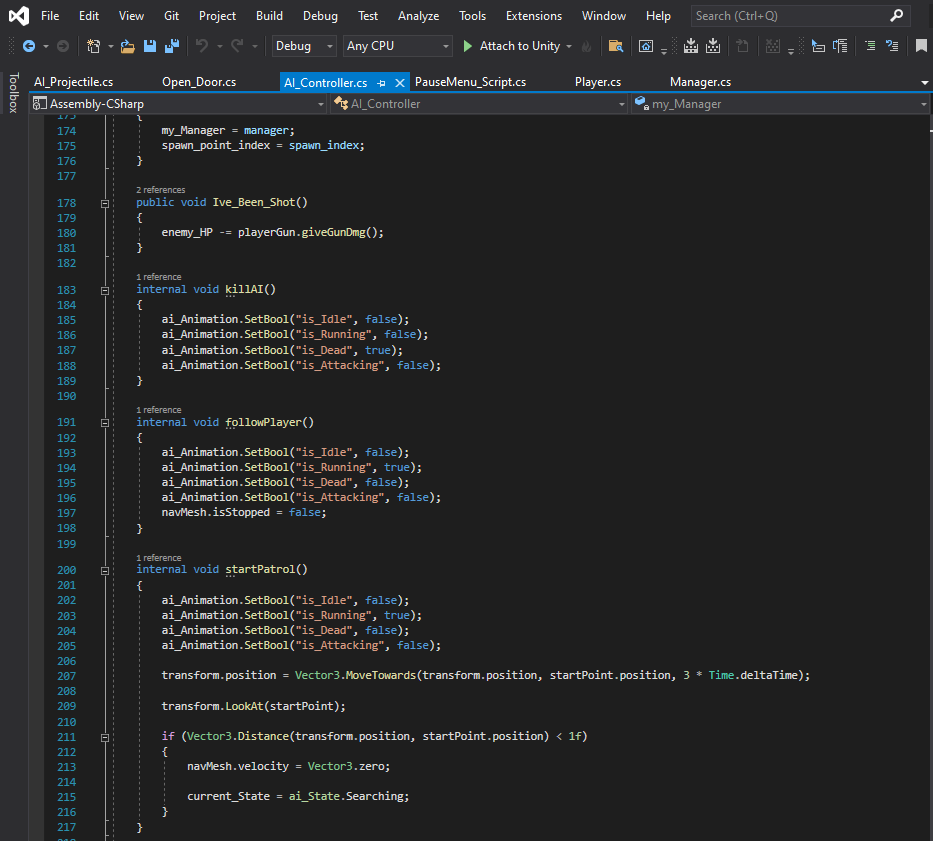
Describe how to play the game here, specify keys/mouse etc. what needs to be done to unlock further features etc.. i.e. a walkthrough which covers all of what is to be seen to be marked.

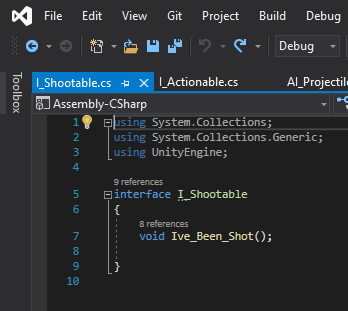
The goal of the game is to find and defeat the boss while you fight endless waves of enemies until you either die or defeat the boss. The boss has shields surrounding it which need to be unlocked by solving a puzzle. The puzzle is solved by picking up a buff which grants you the “operator buff” this allows you to shoot panels on the wall that have turned red. You must head to the lower level of the boss room and search for red panels and shoot them all. Once all panels have been shot the damage phase commences and you must output as much damage as possible. If the boss is not killed within the time his shield comes back down and you must repeat the process.

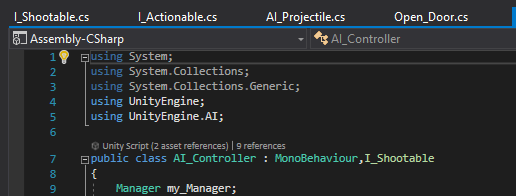
To move, the keys W A S D are used. To look around the mouse is used. To shoot you gun, left mouse button is used and to aim down sights is right mouse button. To run, the shift button is used. There is a secret room located on the right hallway which consists of a code to be entered. There are random words placed around the map. These are the words to solving the puzzle. The key to figuring out these words is located in the room where the secret panels are located on the celling. The puzzle is a Caesar Cipher. The code to solving the puzzle is 4695. This unlocks a new weapon that has increased damage and shoots full auto. To switch between these weapons the buttons 1 and 2 are used. To reload your gun once all the ammo is shot, the button R is used.

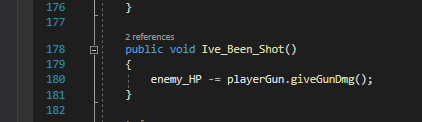
# Coding

Under each of the following headings, please describe the concept, why is it or isn’t it useful/needed, where do you implement in you project, you may provide screenshots or cut and past code segments etc..

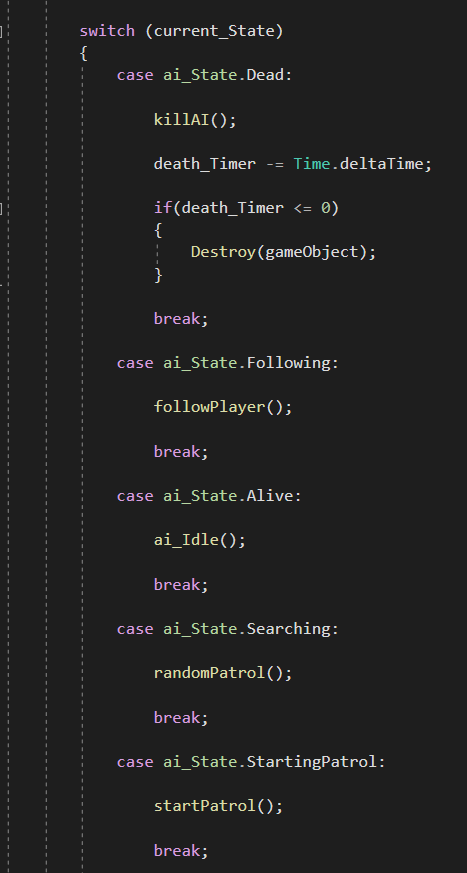
* Frame Rate Independence
  + Frame Rate Independence is where the game runs at a constant speed and is not dependant on the computers frame rate. No matter what frames the pc runs at the game will stay at the same speed
  + This is useful especially for motion within a game. If a character is moving and is not frame rate independent, if the computers frame are running at 60 frames or higher the character will move much faster compared to a computer running at lower than 60 frames per second. Frame rate independence will allow the character to move at the same speed no matter the computers frames per second
  + I implemented frame rate independence on for example my AI character movement, my main player movement, door movement and AI projectile movement
* 
* Interfaces
  + An interface is a class with an empty method and when another class implements the interface that class must implement the method from the interface and there the method can be used to do what the programmer wants and if another class implements the interface it will implement the same method but can have different code written within the method
  + Interfaces can be useful when having objects that are different but you want them to carry out the same task through code
  + I implemented interfaces in my project such as a shootable interface, and actionable interface



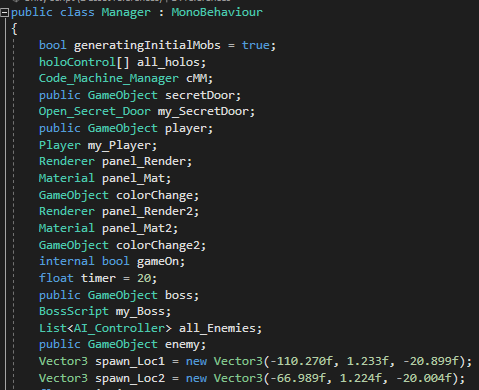




* Inheritance
  + Inheritance is when a new class is defined based off of an already existing class and will use data that already exists in the class it was defined from
  + Inheritance can be useful when having characters of the same type but carry out different functions for example, a character class exists and so does a bad guy class and good guy class. For example both of these characters share something in common. They share Health points so health points will be defined in the character class and both the good guy and bad guy class will take this data from the character class
  + I did not implement inheritance as I did not have use for it within my project
* Case pattern
  + Case pattern is when states are defined to a script. Multiple states can be made and tied to cases. A current state will be set to one of the state when a certain condition is met and within the case code will be carried out until the state is changed and the next case will take place
  + Case patterns can be useful in the terms of having a character that must change different states based on what happens in the world
  + I implemented case patterns on my AI. This allows my AI to switch between an attack state, search state, death state and start state for example. If the AI is in the search state the case will tell the AI to go to a random point within a set amount of points until the case is changed



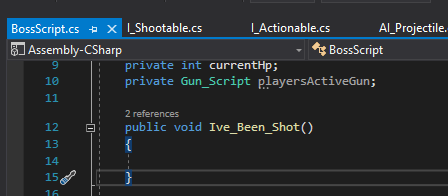
* Observer Pattern
  + An observer pattern is a class that overlooks everything within the project and keeps note of the changes that take place and can give information to other classes
  + Observer patterns are very useful for when you need one set class to overlook either the whole project or a certain set of items that are similar
  + I implemented and observer pattern in the terms of a manager for my project. I had the manager overlooking my AI in which they are stored within a list and making sure that a set amount of AI are only spawn in at one time. If they die they ae taken out of the list and then if another needs to be present another AI is spawned and added to the list



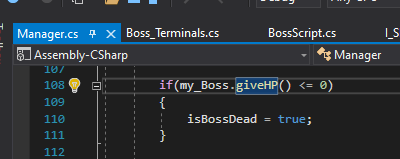
* Polymorphism
  + Polymorphism is when a single interface is implemented onto multiple classes
  + This is useful for when you want to have multiple objects that implement the same interface and methods basically acting as the sub-objects
  + I used polymorphism in the terms of my I\_Shootable interface with my boss and AI both of these are shootable objects and in term shall carry out the same function

Graphical user interface, text, application

Description automatically generated



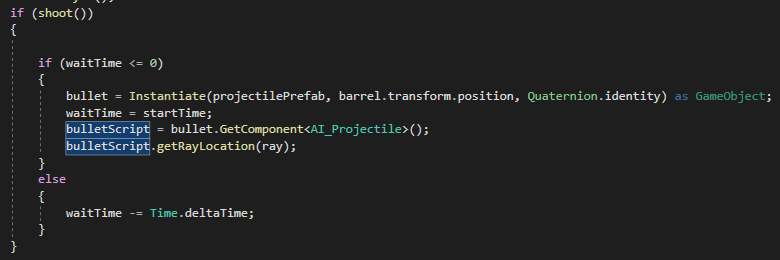
* Communication between scripts/game objects
  + Communication between scripts is when two or more scripts can talk to each other and share information and data between each other
  + This can be useful for when you need to find out how much damage is being applied to something from for example a gun or a sword
  + I implemented this on my boss and manager for example. I have a method in my boss script that returns the current hp of the boss. Within the manager I have the boss method returning the hp to the manager at all times to check if the current hp is below 0 or not



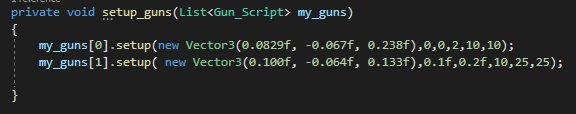
* Instantiation and Prefabs
  + A prefab is when a game object is turned into its own object or prefab, within the assets and is identical to the original and can be brought into the game as many times as needed but as its own unique object. Any changes made to the prefab will be made onto all objects created from the prefab. Instantiation is when a prefab is created or “Instantiated” through code usually
  + This can be useful for when you want to make a carbon copy of the prefab within the game
  + I implemented prefabs such as my AI and I implemented instantiation for my AI projectiles

A picture containing icon

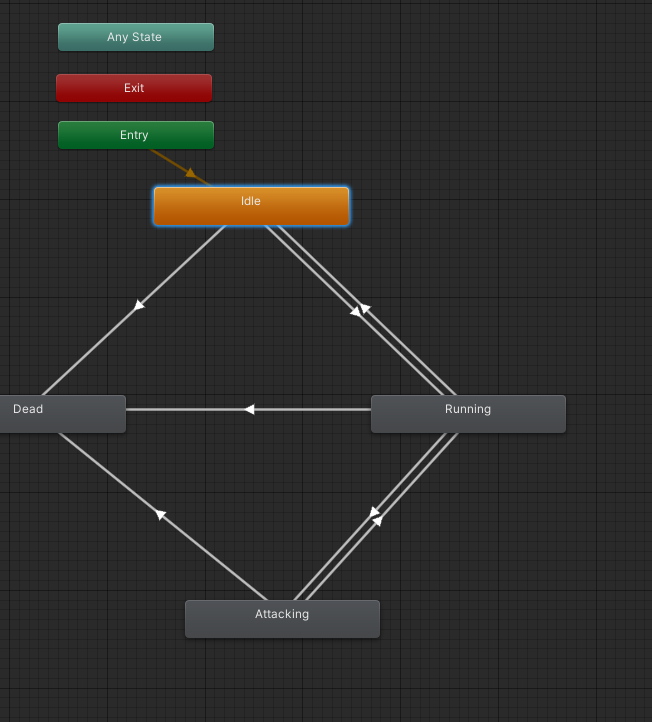
Description automatically generated



* Magic Numbers
  + Magic numbers is when a value that is chosen at random and has no real meaning is hard coded into a variable.
  + This can be useful for when you want to limit variables to a certain value that you would like and not have it be changed
  + I implemented this for my gun damage, When I initialized my guns I initialized them with a magic number which is not derived from anything



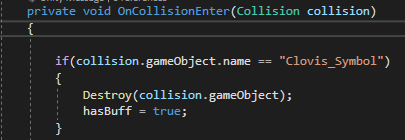
* Model Animation
  + Model animation is when movement is made to a model which is attached to the rig upon the model
  + This is useful for when you want to make the model have more life like movement and to have the game feel more like reality
  + I implemented model animation on my AI and main player models



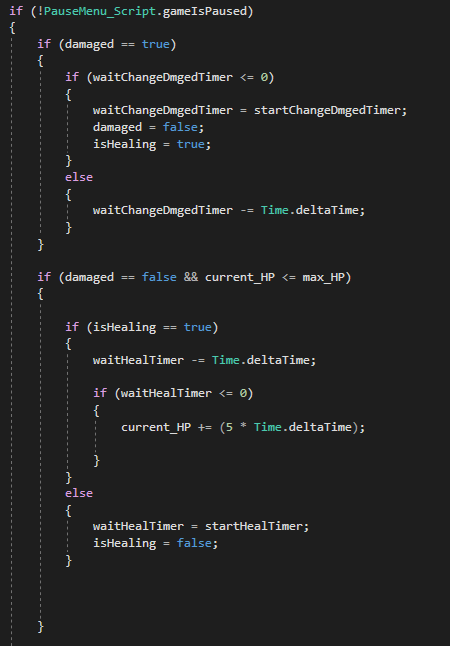
* Self made models and or animations
  + Self made models and or animations is importing a self made model through a 3D animation and modelling program such as Maya 3D and using it within the project
  + This can be useful for when you want to make your own unique models and animations
  + I implemented these through my own created model for the operator buff and model for end character and animation

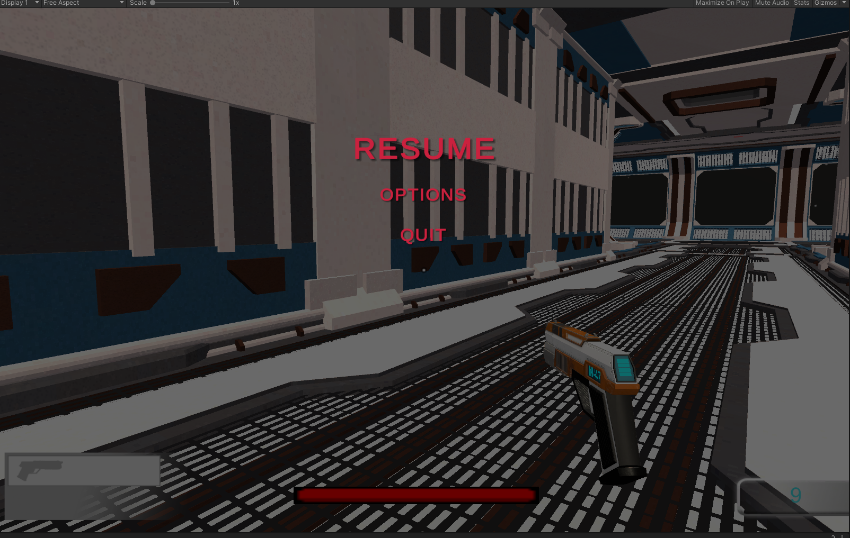


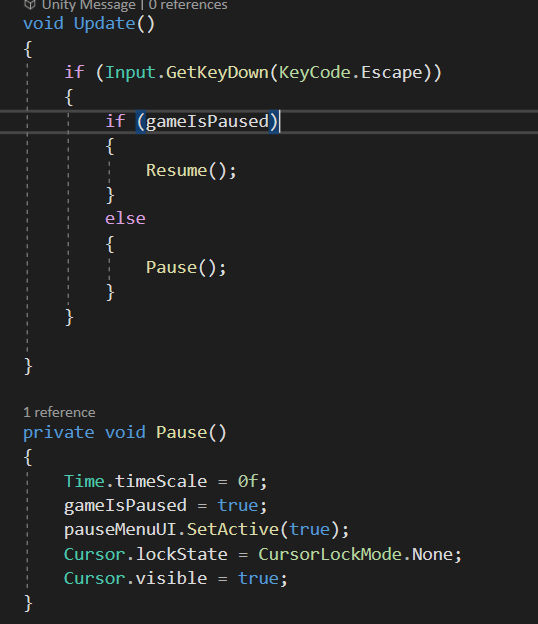
* Interactions between objects/scripts
  + Interactions between objects and scripts is when two or more game objects or scripts interact with each other be it based on a collision or at all times
  + This can be useful for when you need to detect if objects are what they say they are through collisions
  + I implemented this through the use of collisions with the operator buff and main player. If the main player enters the collider of the operator buff game object it will destroy the game object and say that the player has the buff



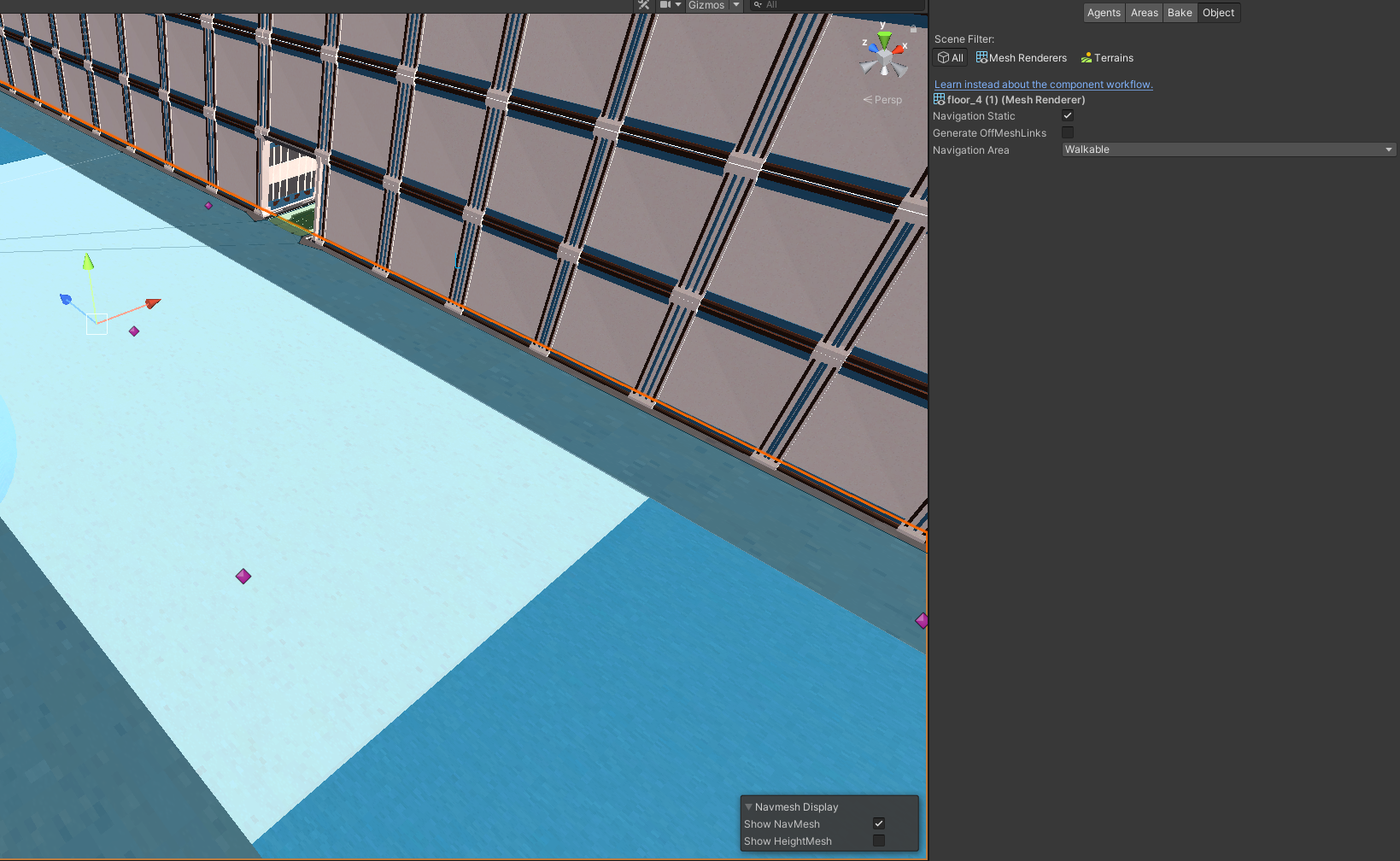
* Propper code placement
  + Propper code placement is when code is written within the correct format and positioned correctly within the right script
  + This can be useful for when working on a project as part of a team. If proper code placement is used then code can be read clearly and followed the way that it is meant to
  + I implemented this all throughout my script so that actions and decisions can be followed almost linearly



* Code repetition
  + Code repetition is where the exact same lines of code is used in multiple places throughout a script or multiple scripts
  + This is not very useful in almost all cases as it can be seen as bad coding practise but it can be useful when code repetition is seen as the only possibly way to carry out code
  + I did not have a need for code repetition
* Pause/Pause Menu
  + This is where the game pauses so that the player can take a break or for some reason has to stop playing the game and also make changes to their sensitivity within the options
  + This is useful for when the sensitivity is too high or low for the player and the pausing feature is useful for when the player wants to exit the game or stop the game from running until they are ready to resume
  +  I implemented this throughout the main game



* NavMesh
  + A NavMesh is when a mesh is generated around the chosen play area of the map and then baked into the map and can be used to indicate navigation for AI and players
  + This can be useful for when you need to have a an AI navigate around a map by themselves or to create boundaries around a map to disallow clipping through walls and floors
  + I implemented this for my AI and main player. For my AI this allows them to walk within a certain area to points and stops them from entering areas where they shouldn’t. For my main player this allows the main player to walk around the map with ease without clipping through walls and the floor



* UI Creation and Elements
  + This is where I created my own User Interface to allow visual representation of the games elements to be displayed to the player
  + This is useful for when you want to show the player visual representation and to interact with the game and to allow the user to customize their own experience of the game
  + I implemented this for on the main scene in the form of a Heads Up Display that displays the current weapon, ammo, health and boss health. I also implemented this through a pause menu which allows the player to pause the game and select to resume, quit or go to the options to change the sensitivity, resolution and to set the game to full screen mode or not

