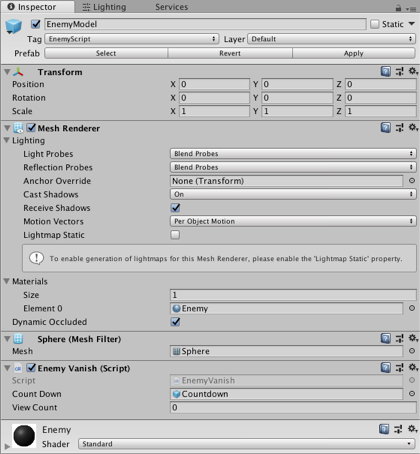
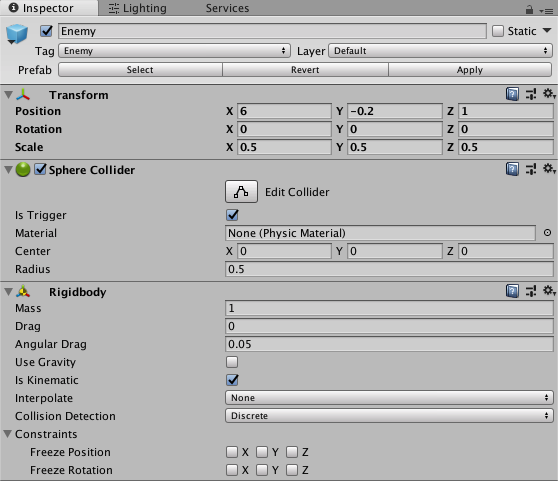
**Component 2 – Enemy Vanish**

In this component tutorial, I’ll be creating a script that will allow enemies to vanish at the beginning of each level whilst keeping their colliders in tact. This will serve as our “memory maze” for the player. I’ll also be adding the ability to reveal the enemies upon pressing the spacebar.

To begin, I separated my enemy in to two game objects – the “Enemy” parent that would hold the spheres information for collision (Rigidbody, Sphere Collider, Enemy tag) and the “EnemyModel” that would hold the script to deactivate the Mesh renderer.

**../Screen%20Shot%202018-12-05%20at%2017.17.00.png**

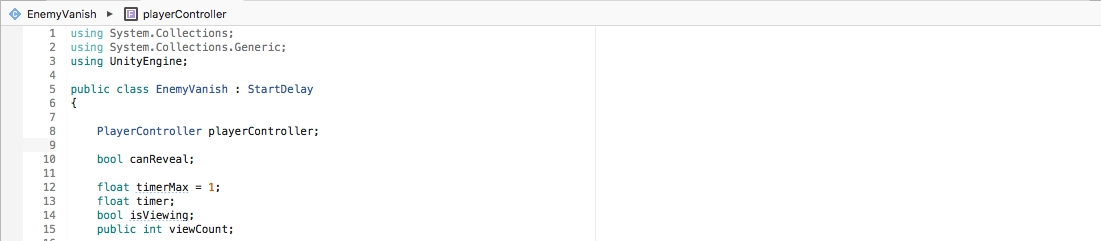
****

We want our enemy’s vanishing to coincide with the the level countdown so we need to derive the Enemy vanish script from our StartDelay script by using public class EnemyVanish : StartDelay.

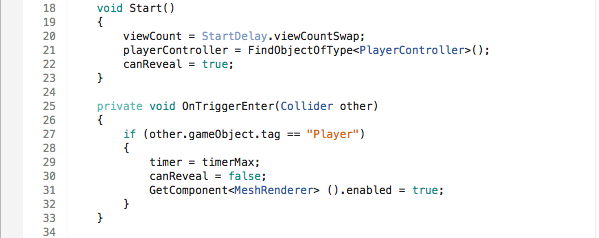
As we will need to pull variables from our player controller, we’ll type PlayerController playerController and define the game object later in the Start class. We’ll also add a bool to dictate if the enemy can or can’t be revealed called canReveal.

I want the enemies to only be revealed for a certain amount of time – to do this, I’ve created floats for timerMax = 1 and timer. This timer will coincide with how long the enemy is displayed for before vanishing again.

Finally, I’ve created a public integer for viewCount which we’ll use to dictate how many times the player can view the enemy.

****

In Start, I’ll assign the variable playerController to the PlayerController script using FindObjectOfType<PlayerController>(); and set canReveal to true.

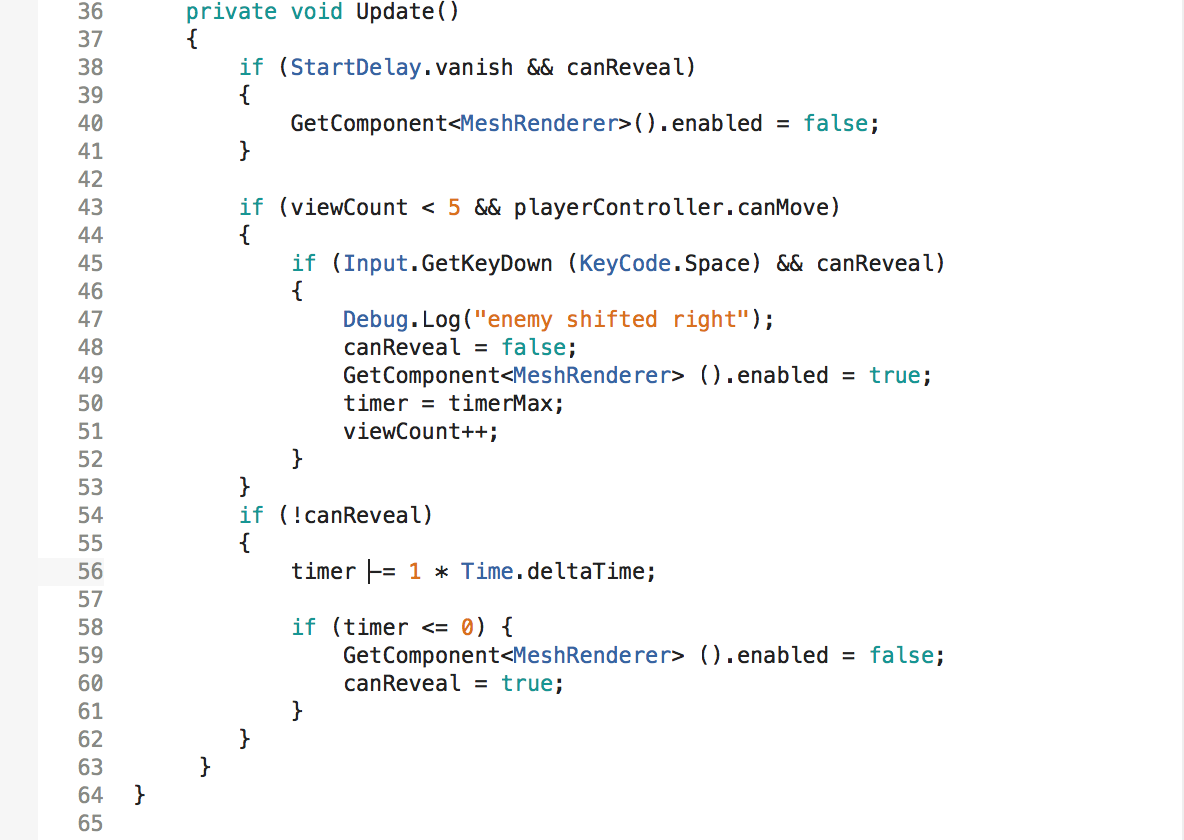
****

In update I’ll need to use multiple if statements to determine when the enemy is invisible, when the enemy can be revealed depending on a key input and the canReveal bool and if the enemy is still visible depending on our timer.

For my first if statement, I’ll deactivate the enemy’s mesh renderer by referencing our StartDelay script - if the canReveal variable is also true then the mesh renderer is set to false.

For my second if statement I’ll check if the viewcount is below 5 and if the canMove variable from our playerController script is set to true. If both of these apply, then another if statement is executed – if the space key is pressed and canReveal is set to true then the enemy’s mesh render is activated, the timer is set to 1 and the viewCount is incremented by 1. By doing this, this means that the viewCount can’t be exceeded 5 times and timer is set to be decreased in our final if statement.

In our final if statement, we’ll check if the canReveal bool is set to false. If it is (in our instance of pressing the space key), then the timer will decrease be 1 \* Time.deltaTime. If this timer reaches below zero, then the MeshRenderer is set back to false and canReveal is set to true. Since our timer is set to 1 upon the spacebar input, this means that our enemy is only revealed for one second in total.

****

In turn, this combined with the start delay script defines the basic enemy vanish mechanic. We’ll be re-using this script in our last two components to create enemies that can move upon the space bar being pressed whilst remaining invisible.