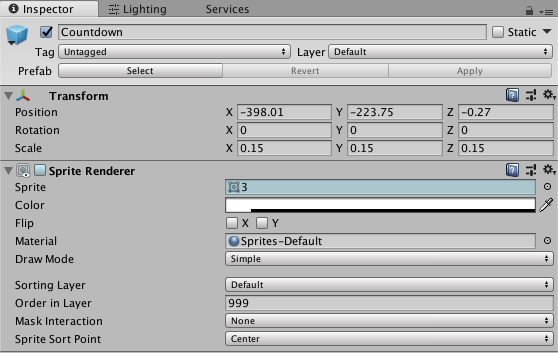
**Component 1 – Level countdown**

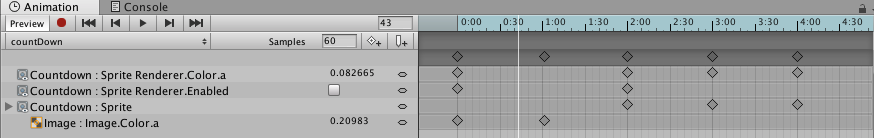
For my individual components, I’ll be using my previous tutorials and scripts as a basis to create a functioning mini game that’ll rely on the player’s ability to memorise the movement pattern of invisible enemies. To do this, I’ll be creating 4 main components – A countdown, vanishing enemies, enemies that can move in a straight line and enemies that can move in a square pattern.

As I would like my player to be able to see the enemies for a short period of time before they vanish, I’ll first be creating a countdown component that will freeze the game prior to starting.

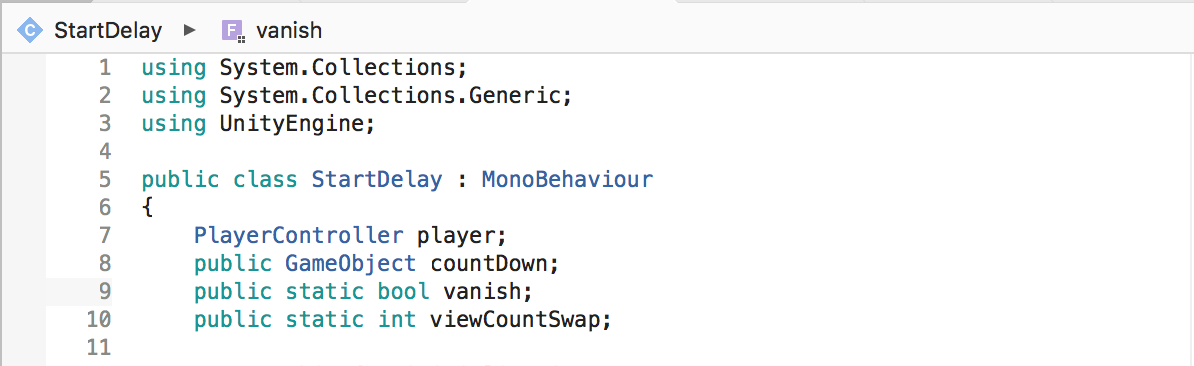
I started by creating an empty game object called “Countdown” and another within this called “CountdownDelay” to hold our script. I then created 4 PNG images for my countdown with the numbers 3,2,1 and the word “GO!” and converted these to sprites. I then dropped the 3.png in to the scene and put this image in the Countdown parent using the sprite renderer. I selected the image, opened the animations pane and cycled the number from “3” through to “GO!” and positioned the countdown within my scene to fit the stage. As an extra touch, I also increased the alpha value so the countdown becomes gradually more visible until it reaches “GO!”.

../Screen%20Shot%202018-12-05%20at%2015.56.11.png





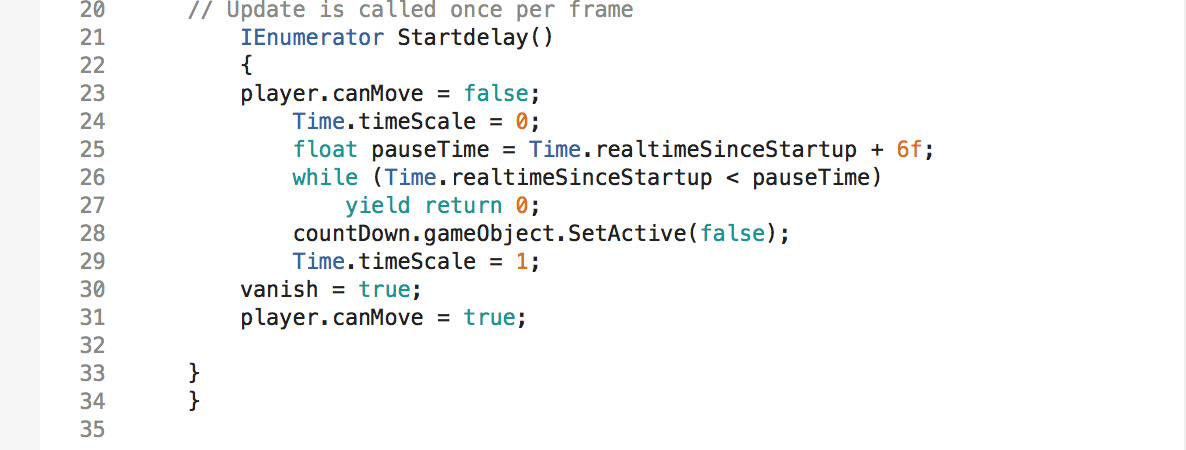
On the countdownDelay game object, I created a new script to control the countdown and when the player is able to move. As I need to access variables from my player controller script, I first added PlayerController player to my scripts public class – we’ll come back to this in Start. I then added a public gameobject called countDown to access my images and animations. As this script will be integrating a lot with the vanish script, I’ve created a static bool for vanish.



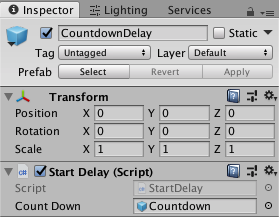
In start, I assigned the player variable to the player controller by using player = FindObjectOfType<PlayerController>();, set vanish to = false and initiated a coroutine called (“Startdelay”).



I know that I’ll need the effects of this script to run for a certain time whilst pulling variables from other scripts. To do this, I reference my Startdelay coroutine using IEnumerator Startdelay() and set out the foundations for when my player can and can’t move as well as when the enemies will vanish. First, I don’t want my player to be able to move so I’ll disable the player’s movement by access the playermovement script and setting the canMove variable to false using player.canMove = false. I then create a float called pauseTime that counts for 6 seconds from the scene’s startup using Time.realtimeSinceStartup + 6f. I then use a while statement to reverse these effects by stating that while Time.realtimeSinceStartup < pauseTime (which is the start up time of 6), it’ll set the countdown gameobject to false, set the timescale to 1, vanish to true and player canMove to true. In turn, this means that once 6 seconds of realtime has passed since the scene has started, the player will be able to move and the enemies will vanish.



In my countdownDelay object, I then assign my script and drag my Countdown object in to the Count Down field.



To complete this component, I’ll then need to create an enemy vanish component to utilise the enemies mesh renderers and numerous variables.