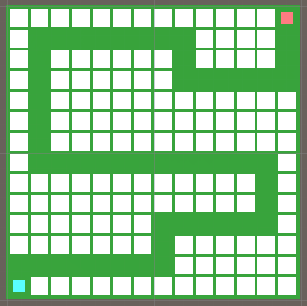
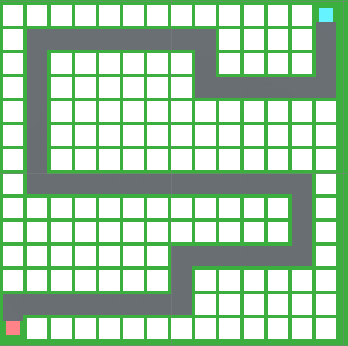
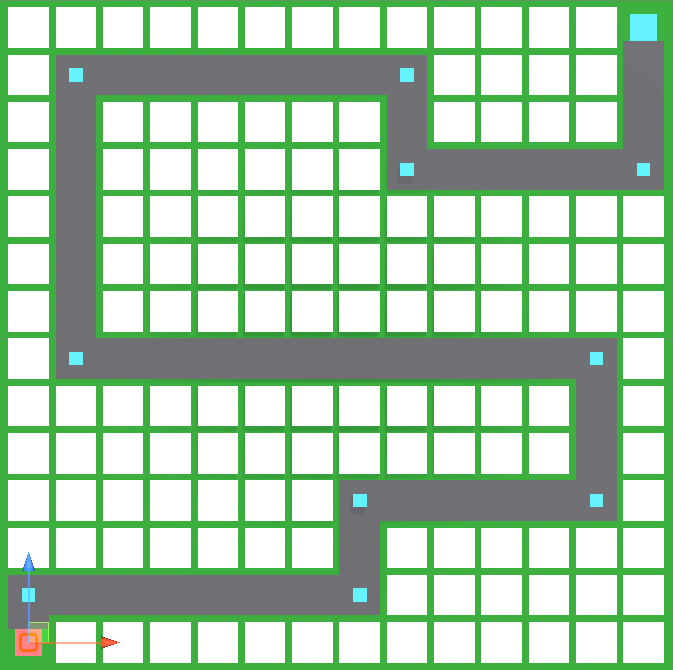
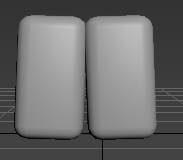
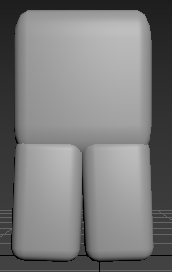
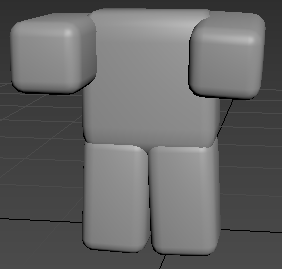
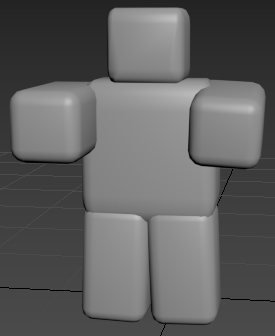
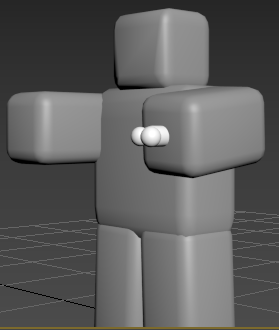
**Production Log:**

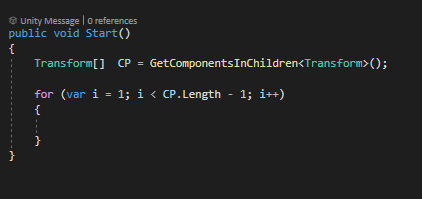
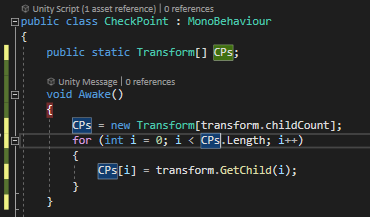
Map:

Insert a plane for the base of the map.  
Insert 2 cubes, one for the start of the path, and one for the end of the path.  
Create an empty parent, to put all the tile's assets inside (To make the workspace more organized)  
Then, insert a cube. This is to make the tiles where you will place all the towers. Then duplicate them and place them in a grid pattern. Delete some of the tiles in the shape you want the path to be. Like this:  
  
Then insert another empty parent. This is for the paths. Fill in the empty spaces with paths.  
Like this:   


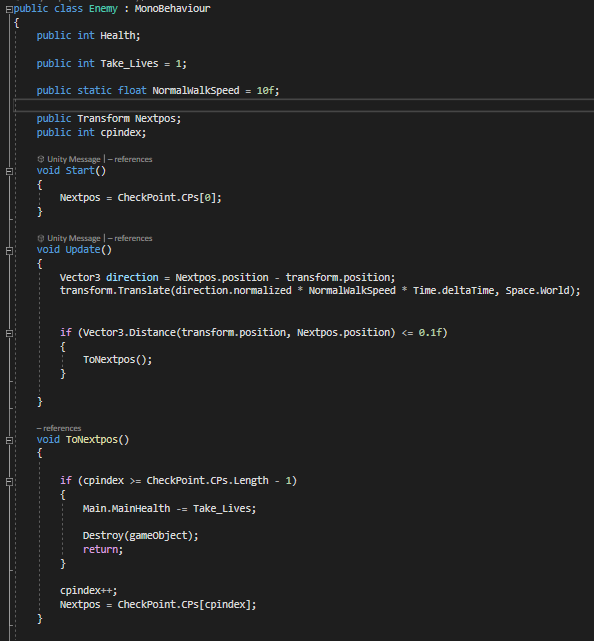
Then to make the path that the enemies will follow, I added an empty game object in order to hold all the parts. I inserted a cube at the end of the first path, so that when the enemies spawn, they know to travel there. Duplicate them and place one at the end of each path, and one inside the end cube.



Modelling:   
Insert (in extended primitives) 2 chamfered boxes. Rotate them away from each other. These will be the legs.  
  
Then insert a new chamfered box. Place it on top of the legs and shape it to be the torso.  
  
After that, insert another 2 chamfered boxes, and rotate them 90 degrees. These are the zombie’s arms.  
  
Then, to add the head, add 1 last chamfered box and scale it to the body.  
This is the ”normal zombie” model done.  
  
(To make the weak zombie model)  
  
Then scale down one of the arms, insert a cylinder, 2 spheres, and place them together to make a bone. This is the ”weak zombie” model done.   
  


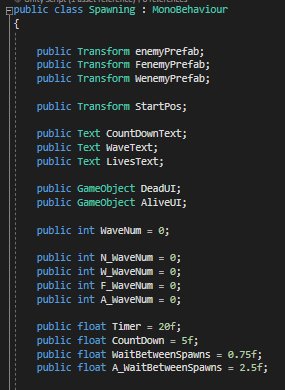
Scripting:  
  
In order to get the positions of the check points that the enemies will follow, I wrote a script to put them into an array automatically. This was my first script:  
  
Then I got stuck on the next steps, so I rewrote the script, with help, to make this:

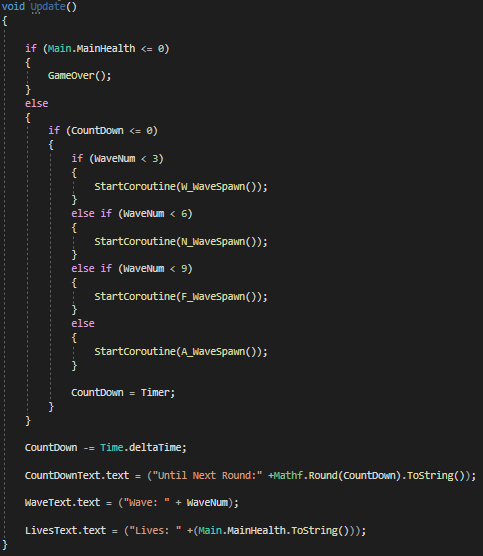
This checks how many children are in the parent object, and places them all in an array to loop through for the enemy path.

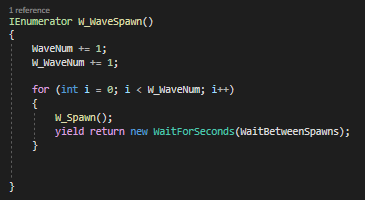
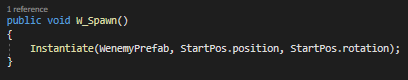


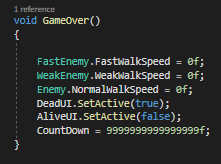
The variables here are self-explanatory. The only thing I would change from each enemy script, is the “Health” and “NormalWalkSpeed” variable, and value.  
When the enemy spawns, it will set the Checkpoint to position 0 (The first checkpoint on the map.)   
In the update function, I made a new variable for the direction of the next checkpoint. In the next line, it gets the direction, and times’ the enemy’s walk speed with Time.DeltaTime. This will make it travel at a constant speed instead of being jolty

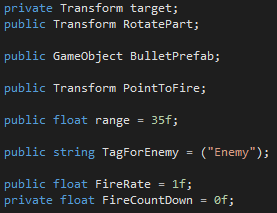
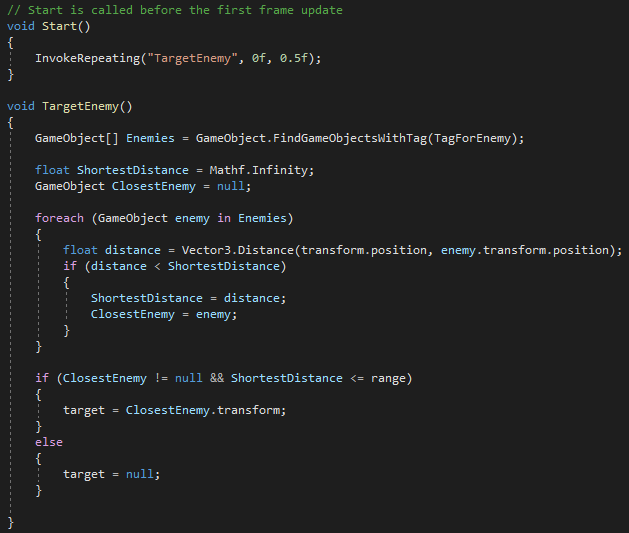
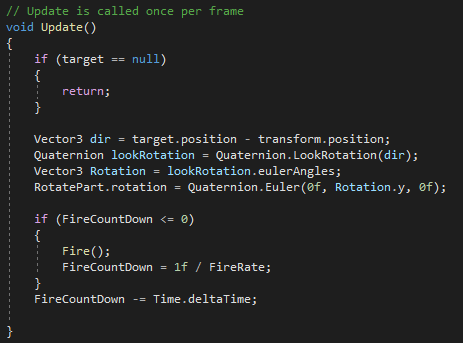
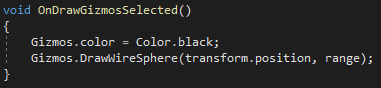
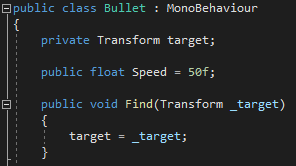
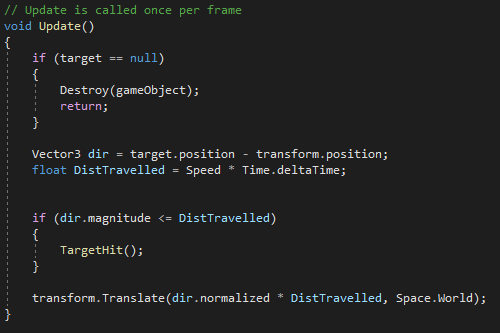
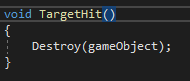
If the enemy is within 0.1 units away from the next checkpoint, it will call the function to set the next checkpoint. This is done by each time the function is called, add 1 value to the cpindex variable, and set the next checkpoint position to that variables value.  
If the next checkpoint position doesn’t exist, the enemy gameObject will be destroyed, and a life will be taken away.

  
These are the variables I set for the spawning script.

  
In the update function, it checks if the lives are above 0. If they are not, it calls the GameOver function. However, if the lives are bigger than 0, it carries on with the rounds. The first 9 rounds are sort of tutorial rounds, it gives the player a taste of the 3 enemies. Grey being weak, Purple being normal, Orange being fast. When it reaches round 10, it goes to the ”A\_WaveSpawn” function, ”A” meaning ”All”.

  
In the W(eak)\_WaveSpawn function, it adds 1 value to both the WaveNum and the W\_WaveNum variable. Then, in the for loop, it reads how many weak enemies it needs to spawn, using the W\_WaveNum value, and then calls the W\_Spawn function.  
  
In this function, all that is happening is it spawns the enemy.  
  
This is the same process that’s happening in the ”N\_WaveSpawn”, ”F\_WaveSpawn”, and ”A\_WaveSpawn” function.

If the GameOver function is called;  
  
The enemies walkspeed is set to 0, the AliveUI is disabled and the DeadUI is enabled, and the countdown is set to ”9999999999999999” so that no more rounds are spawned in the time that the DeadUI is enabled.

The next script is “TurretScript”  
  
  
These are the variables for this script.  
  
In the start function, this line calls the ”TargetEnemy” function every 0.5 seconds.  
  
In the TargetEnemy function, it creates an array for the enemies, a float to check what the shortest distance from the turrets is, and a gameObject to store the closest enemy that the turrets will target.   
The Foreach loop checks if the next enemy is closer to the turret than the current target. Then if the target exits the turret’s range, it loses the ability to track it.  
  
  
If there are no targets, then it will exit the Update script. Otherwise, it will read the direction of the target, and rotates the turret in direction of the closest enemy. If the FireCountDown reaches 0, it will call the Fire function, which instantiates a bully prefab, and follows the script to the target.   
  
  
This function basically shows that if the turret is selected, it will show the range in a sphere in the colour black.  
  
  
  
  
  
These are the variables for the bullet script.  
  
  
If the target is ”null” the bullet will be destroyed.   
Otherwise, it will make it a variable for the direction of the target, and a variable for the distance travelled in that 1 frame. In the if loop, if the direction’s magnitude is less than distance travelled variable, it means it has collided with the target so it will call the TargetHit function. Then the last line, it will move the bullet in the direction of the target.  
  
   
At the moment, the only thing in this function is the bullet gets destroyed. In the future, I will change it so the target will lose health, and eventually die but for now, the time limit restricted me from doing so.