**ROLLING ROADS BRIEF**

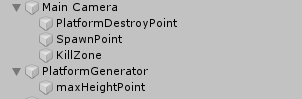
**BY SAMUEL PARSONS**

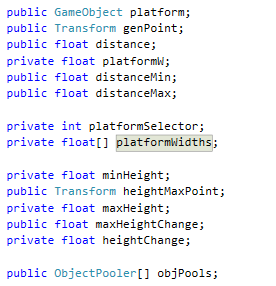
For the rolling roads brief I used a game I made for a graphics design company for christmas 2019. In this game the player would be challenged to run and jump for as long as they could in this snowy infinite runner. This project was a huge learning experience for me that helped me learn a lot about coding as well as encouraged me to go out of my depth.

In this documentation I will focus on the platform spawning section of the game.

To begin I created an empty gameobject that I named PlatformGenerator which I attached a script component to called PlatformGenerator I also attached another empty gameobject which I named maxHeightPoint which would be used as reference as the highest point platforms could spawn whilst the PlatformGenerator itself would represent the lowest point.

I also created 3 empty gameobjects on the main camera the first was the SpawPoint which I placed far ahead of the player out of the camera view, next was the PlatformDestroyPoint which to its name destroyed all platforms that reached its position and lastly was the KillZone which would kill the player if they missed a platform and fell(For the killzone I attached a rather large box collider).

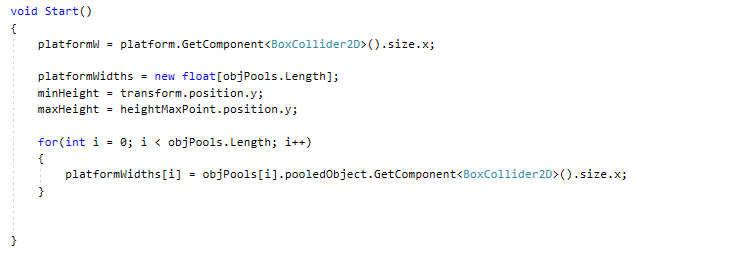




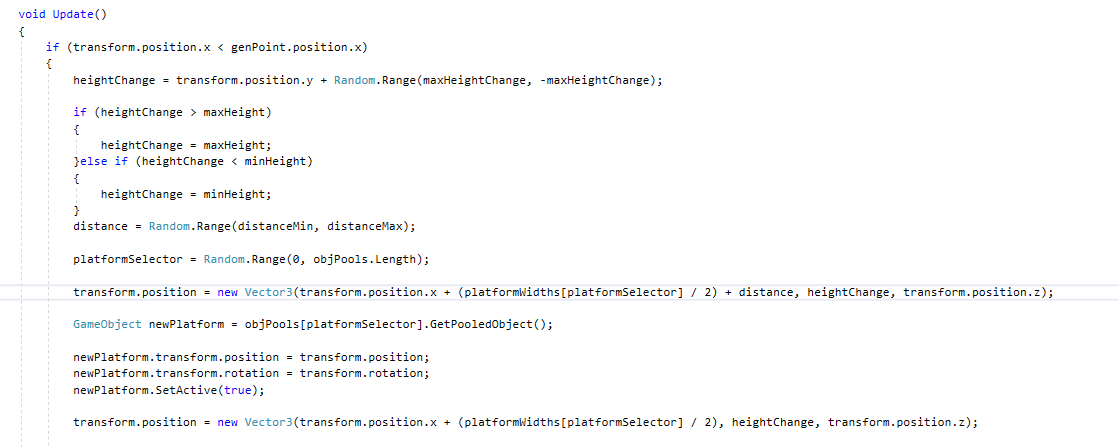
These were all the variables I used within the platform generator script. The first two variables are for the first platform that was pre-places as well as a float array which has the width of the platforms in. Next was the floats for distance, due to the range of sizes of each platform I had to make sure a reasonable gap was between each platform as well as make sure no platforms would be inside one another. To do this I would use the distance float to determine how far apart they will be, this would be by using the minimum and maximum distance they can be whilst still being fair as well as the width of the platform to compare how far it should be from the platform before it so it's not inside it or miles away.

Next was the platform selector int which randomly chose the next platform using a random.range number to choose from the platform widths objectPooler, the object pooler is what I used to hold a list of various platform prefabs that I could pick and choose from to spawn platforms.

I then had to use a series of floats to determine how high and low each platform can spawn from its previous ones to also allow fair gameplay. To do this I used a minimum and maximum height as well as the minimum and maximum height change. The normal heights where the absolute maximum that each platform can spawn without it being above or below the camera whilst the height changes were how high and low they can spawn whilst still being able to reach.



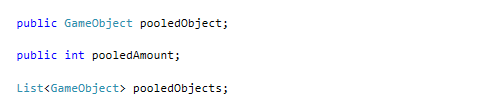
In the start function I then set the value of platformW to equal the width of the first placed platform. I then also set platformWidths to equal a fresh new float that was equal to the length of items within the objPools. I also set the minimum and maximum height to their respective anchor gameobjects being the spawner itself and the maxHeightPoint object. I then also set each of the pooled objects within platformWidths to be equal to their respective collider width.



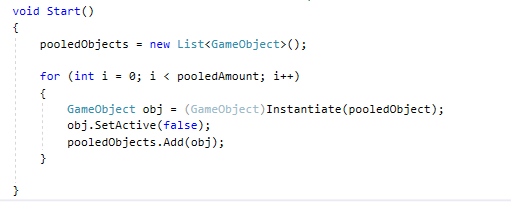
I then proceeded to use a void update function to check whether the platformGenerators position was behind the genPoint of the platforms and if so the height of the next platform will be randomly assigned by the transform value of the spawner plus a random range between the highest the platform change can be both negative and positive. The heightChange value is then checked if it's higher than the max height or lower than the minimum height if it's either the height change is set to be equal to the max height instead.

Next the distance value is determined with another random range between the furthest and closest values the platforms can be. The next platform is also determined by selecting a random number between 0 and the length of the pools list, then the position of the spawner is moved to be equal to its current position plus half the width of the next platform on top of the distance and height change variables calculator earlier then the z position of the platform which is always 0.I then call the GetpooledObject() function which is part of the next script. I then take the next platform and set it to be the value of newPlatform, set its rotation and position to be correct and then make it active.

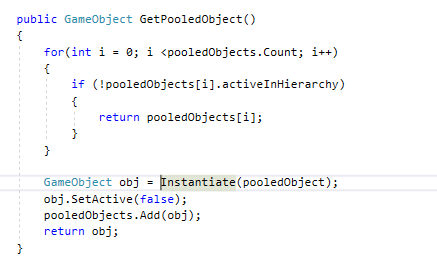
Now that I have gotten the platforms spawning and moving I needed to create the object pooling system. To do this I created a script that I attached to a gameobject that could then hold prefabs for calling later. In the script I began by making a reference to the object that will be pooled in this script, and integer that holds how many of that object there will be and then a list that would hold all the objects for spawning.

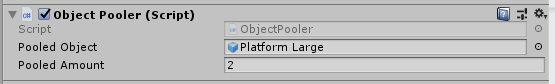
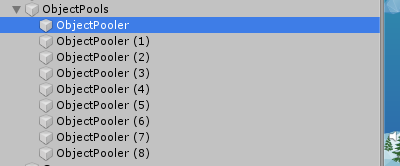


I then used the void start function to make the pooledObject variable equal to a new list. Then I created a for function that would loop for each of the objects inside the pool. Each object is then instantiated at the start of the round and set to inactive so its not visible and then the objects are added to the pooledObjects list.



I then created a function called GetPooledObject which takes the pooledObject and checks to see which ones are active, it then returns the value of those active platforms inside that pool and sets them to inactive. The pooled objects are then returned to where it was called for use in spawning more without removing any platforms below the player.



Now I made an empty game object in the hierarchy called object pools and then created a series of empty gameobjects I named ObjectPoolers with the script attached that held multiple copies of 1 platform each.

Finally I made a script that would deactivate platforms once they reach past the player. This was quite simple, I just made a reference to the point in which platforms should be removed. Once a platform reached the same x value as the destruction point set that platform to inactive so that it may be reused later.



All together along with a simple player movement script the player would move to the right of the screen with the camera following him whilst platforms would be grabbed from the pool and set active until they reach the PlatformDestroyPoint in which they would be deactivated until they are used again.