**Gravity Boii Runner Documentation**

**Scrolling Background:**

1. Pick a sprite and save it as a **default** texture type and change the wrap mode to **repeat**
2. Create a material and set the sprite as the albedo component and then change the material type to unlit sprite default
3. Apply the material to a quad
4. Change the tilling as necessary in the material to create the background.
5. Create a script that gets the material of the game object the script is on and changes the offset every frame with delta time
   1. *Material.offset += value \* time.deltatime*
6. Apply the script to the quad and make sure that the value is public so it can be changed for different sprites to create a parallax effect.

**Touch Controls:**

1. In the Update() function, you are going to check if *Input.touches.length* is greater than 0. This means that we have had a touch on the screen.
2. Save the first touch in a local variable with *Input.GetTouch(0)*
3. Run an if that if the current phase of the touch is the beginning and then save that into a vector as the first touch pos
4. Then run another if for the end phase and save that in another vector as the second touch pos
5. The difference of the two vectors is the length of the swipe.
6. Save the length of the swipe and then check that it is not a tap by having its magnitude not be less than a certain value.
7. You can now use this swipe after normalising it to see what direction it was
   1. With room for error on the +- on the x axis, if the value is greater on y, it was an up swipe
   2. With room for error on the +- on the x axis, if the value is lesser on y, it was an down swipe
   3. With room for error on the +- on the y axis, if the value is greater on x, it was a right swipe
   4. With room for error on the +- on the y axis, if the value is lesser on x, it was an left swipe

**Player Controls:**

The player can dash a short distance vertically and shift between walls. This is handled by the same block of code.

1. There is a timer which will control the movement of the player. Essentially for how long the dash will be allowed.
2. First we will check if the player is not moving (no swipe is done). In this case we will set the velocity of the player to zero and reset the dash time.
3. Second if the player is moving (Swipe has occurred). We will first decrement the timer with delta time and then for each case of the swipe.
   1. If it is up or down. We will use vector up or vector down and multiply it with the dash speed.
   2. If it is left or right. We will use vector left or vector right and multiply it with the dash speed

**Object Pooling**

Object pooler is a technique used to pre-instantiate several game objects which can then be set to active or inactive and recycled as per use. In order set up the object pools

1. First create a dictionary with the tags as strings and the items as a queue
2. Create a public class with elements needed (tag, prefab object, size)
3. Create a list of the type class that you made above.

The point of making the list of that class type is to fill the list with the different types of prefabs and their tags and their count.

Once the list is populated. We will first initialise the dictionary in start. And then we will create a for each loop to take every single class element in the list and create a queue of game objects. And then using a for loop.

* 1. Create a gameobject
  2. Instantiate prefab in the class
  3. Set active to false
  4. Enqueue the game object

Once the for loop is complete. Your queue should be filled properly.

And now simply add the queue along with the tag of that list to the dictionary we created and initialised.

Now that the dictionary is ready. It can be used. In order to use it, according to the tutorial I followed, they used a kind of a singleton.

What they did was create an instance of the class.