Programming Briefs Reflective Statement

Instanced Scrolling Material:

The instanced scrolling material was the first brief that I did. I was still getting used to using Unity and programming in C# so it took me a while to get the hang of it. I used the MeshRenderer component to make the material move. At first I had some issues controlling the speed of the material but I solved that by making the texture offset increment instead of using an equals sign. In the inspector I had to change the wrap mode of the texture to repeat instead of clamp, which is what it was by default, because the material was getting stretched out when it tried to scroll.

Instead of directly applying the material to the gameobject I made a script to create instances of the material that would be deleted when they weren't needed to prevent memory leaks. I did this by creating an instance of the material then assigning it to the renderer. I also included the OnDestroy() function that can be used to destroy the instances when they aren't needed anymore.

In Game FPS Counter & Graph:

For this brief I used the TextMeshPro library and a Canvas gameobject to create an fps counter ui element. I used Time.deltaTime and some math to calculate the fps. I made the fps counter update once a second instead of every frame so that it is displayed more smoothly. I rounded the fps to an int because if it had a decimal it wouldn’t look good. Then I converted the fps to a string and attached the string “fps” to the end of it so it would be displayed as “00 fps”.

For the graph I made an array to store the past 5 seconds of fps data. I used LineRenderer to draw the graph on the canvas gameobject so it would be part of the ui like the fps counter, however I later found out LineRenderer only draws onto the game world and I couldn't figure out how to do it differently unfortunately. I adjusted the line width in the script because the default was too thick.

I also couldn't figure out how to move the graph's location so it gets displayed in the middle of the game space but I wanted it to display next to the counter in the top right of the screen.

Speedometer:

I made the ObjectSpeed script to find the speed of the object using the RigidBody component that accesses the physics of the gameobject. I attached this script to a square gameobject which will be used to test the script. The script calculates the speed in mph by multiplying meters per second by 2.237. The mph is stored as a static variable so it can be accessed by the ui elements of the speedometer.

I made an acceleration script to test the speedometer and the ObjectSpeed script to see if it properly tracked the mph. I did this by using the default unity controls to set keybinds for moving the square gameobject to wasd. I used RigidBody to control the physics of the square and find the current speed of the object. Then I used Mathf.MoveTowards to make the speed variable increment towards the target (accelerate). I converted the velocity into a value that the physics system can accept by multiplying the velocity by the direction you want the object, then I applied the velocity to the gameobject making it accelerate. The script has a bug that a lot of retro first person shooters had where when you move diagonally you move faster due to how velocity is calculated however for some reason in my script the speed increases exponentially causing the Square to move extremely fast if you move diagonally.

For the speedometer UI element I found clipart of a car speedometer on google images.Then using photoshop i removed the pointer from the dial so they were two separate textures so i could make the pointer rotate around the dial. I found -135,135 to be a good range of angles for the pointer to rotate. Using Mathf.InverseLerp I find mph as a value defined within the range 0,100 then use Mathf.Lerp to interpolate between -135,135 using that value. This finds an angle that can be applied to rotate the pointer UI element in a way that simulates a speedometer because -135,135 is the range that the pointer can be rotated for it to still appear within the bounds of the dial with -135 being the pointer at the starting position of 0mph. By applying the mph value to that rotation the angle increases according to the mph which is what creates the speedometer effect.

I also included a text element in the speedometer by rounding the mph to an int then converting it to a string so it can be displayed nicely. I also added ‘mph’ on the end so it would be output as ‘00 mph’.

Conclusion:

Overall this project showed me that I need better time management and that I need to learn to move on when I have limited time, not everything can be perfect. Taking notes as I went through each brief helped a lot when it came to writing the reflective statements which was good because writing is what I normally struggle with the most. I found myself wasting a lot of time on the fps graph and making not much progress on it which was very frustrating but overall there were a lot of online resources available that helped with each brief, the main ones being the Unity Scripting API document and forum posts on forums such as the Unity forums and Stack Overflow.