**Speedometer**

The first thing I did when making the speedometer try to find an already made speedometer asset which I could use and unfortunately, I couldn’t find any and this maybe because a lack of research but most of the ones I wanted you had to buy except one free one I found online. Unfortunately, the free one I found online didn’t display the numbers in MPH, so I took the asset to photoshop where I wrote my own numbers which will display in mph.

A speedometer with a red dot

Description automatically generatedA speedometer with a red needle

Description automatically generated **OLD** **NEW**

After redesigning the speedometer in photoshop I dragged it into unity and made it into a canvas and called this “speedometer” when doing this I made sure the needle was a separate canvas because I would be making the needle rotate around the numbers.

To do this I found the angle where I wanted it to start at which was where the “0” is on the background and the angle where I wanted it to end at which is the “180” mark on the background and I named these my “startAngle” and “endAngle” when I wrote my script, and these angles will determine the range within which the needle will rotate. Then to test this I set my rotate key to “w” to see if it would stop at these angles which it did, which I deleted after testing.

After been satisfied with my speedometer pin going around, I went to find a png of a bike which I could use for free which I found. And then I dragged this into unity and made it into a sprite, which then I put on a empty game object once I did that I added a box collider and a rigid body and turned of gravity and then moved onto making the script so it could move.

In this script I referenced the Rigidbody and wrote this in the update “loat force = Input.GetAxis("Horizontal");  body.AddForce(force, 0, 0);” which enables the bike to move when I press “a” and “d”.

A screenshot of a video game

Description automatically generated

After I added cubes around the bike so it wouldn’t fly out of the screen sometimes when you crashed the bike it would float up or around the cubes so to fix that I completely covered the bike in cubes and took of the textures for some so you could still see the bike.

A screenshot of a computer program

Description automatically generated

Then after this I finished writing the unity script which was attached to the needle and made sure that it converted metres per second to mile per hour as the bike moved. I did this by referencing the bike then made the script calculate the bikes speed in mph which then converted that speed to the angle the needle needed to rotate at. And just to make sure I “[SerializeField]” the bikes speed in mph so I could see it in the inspector.

So now when you press play and press “d” the bike will move and the mph at which it is at will be shown on the speedometer, the only problem that I have ran into is that it doesn’t accurately show the mph meaning on the speedometer it could say 20 mph but it really could be 23 and I didn’t know how to fix this the only way I could make it more accurate is by setting the bikes maximin speed to 160mph.

**Edge Detection Shader**

When Making the Edge Detection Shader initially I watched the wrong YouTube video which showed me how to make a outline shader which you can attach to objects which was wrong, so after wasting my time making this outline I found the right Edge Detection Shader video which I decided I was going to learn from which used the “URP” features in unity.

A screen shot of a phone

Description automatically generated

But before this I wanted to get some ground built and something in the scene to show off the edge detection shader so it didn’t look boring and while doing this I wanted to be able to walk around and actually have a good look at the shader so I looked on YouTube and found a video which showed me what I wanted and how I can do it in 1 minute which was simple.

A screenshot of a computer

Description automatically generated

After watching that video, I copied what he said I needed to do and made the ground how I wanted to make it and I think with it like this it will make the edge detection shader look a lot better to view.

When it come to making the Edge Detection Shader, I found it a little confusing but after watching the video a few times I got to understand how it works. What I did was follow what the video told me to do with making the Fullscreen shader graph. When making this Fullscreen graph I copied what he did in the YouTube video which was making the actual outline shader graph this graph first allows us to detect normal based edges and colour edges.

A screenshot of a computer

Description automatically generated

After finishing that I made the material and named it outline and this material controls the Outline Colour, Colour Threshold, Normal Threshold and the Overlay.

Then i went to the URP project settings and found the “URP – HighFidelity-Rederer” and used the Fullscreen pass renderer feature which tells URP to run what material I pick on the screen. So, I dragged the outline material I just made onto the pass material on the Fullscreen pass renderer feature which added the outlines to my game,

Now in my game you can move around and see the edge shaders and it looks really good this is the end result.

A cartoon of a square object with columns and a ball

Description automatically generated with medium confidence

**FPS Counter**

When Making the FPS Counter I wanted to make it in a first person camera where the player could move around and see the fps go up and down instead of it been on a still screen because then it wouldn’t work or even look as good, so like before I used a YouTube video which shows me how to make a fps player in unity which can move around.

A screenshot of a computer

Description automatically generated

This is what the map I made looked like before adding the fps counter. As you can see its just a basic white map but as you move around, I can have a better reading on what the fps is at.

A white square object with columns

Description automatically generated with medium confidence

Then I found a video which could teach me and help me learn how to make a fps counter.

A screenshot of a computer

Description automatically generated

A black background with colorful text

Description automatically generatedAfter watching the video, I decided to start writing the script which would display the fps in text.

In the script I deleted the “system.Collections;” and “system.Collections.Generic” and replaced it with ”TMPro” since in this I’m using the text mesh pro library on the UI to display the fps.

The “pollingTime” is how frequently the fps display will update and unlike in the video I wanted this to be public so I could easily edit it whenever, the “time” variable measures the time and the “frameCount” Counts the frames I have in the game.

The update function calculates and displays the frames per second (FPS) based on the frame count and time elapsed since the last update and with all that I was finally finished.

But I was missing one final line of code which went in the start and that code only enables the frame counter in development builds. To find this code I had to do a bit of research but after finding it I added it to my script because in the YouTube video it was never used.

A computer screen shot of a program code

Description automatically generated

Finally, after completing the script, I made a UI canvas object in my hierarchy and then a UI Text mesh pro (TMP) to the canvas, then using the “FPSDisplay” Script I created I dragged it only my canvas and then added the Text mesh pro (TMP) to the text variable after doing all that I dragged it onto my players camera and it was done.

Now when I run my game, I can see the FPS I currently have in the top right corner of my screen, and it will only work in the development build which is good.

A group of rectangular objects in a flat area

Description automatically generated with medium confidence

**Reflection**

Overall I am happy with the work what I have done and I have learned a lot more about programming then I knew by myself which makes me happy, but at some point I want to be able to not rely so much on YouTube videos and I want to be able to know this knowledge, but I do feel like the more I watch the more I will learn as I was learning how to make these quite fast.

I encountered a few problems such as using the wrong YouTube videos when creating the outlines but by doing this I also learned how to make outlines a different way without “URP”

When doing these briefs I did get confused time to time but after doing the right research I managed to get back to been focused and learning how to make a fps player was very good because I never knew how to do that in the unity 3d mode but now I do I can use it to show off and make different games so I am also happy I learned that.

By far I found to make the edge detection shader to be the hardest to make and confused me a lot but since I was determined to get it done, I got though it and I think it come out quite well.

Although with the fps counter I wanted to make the graph and make more land to step on and parts where you might see low and high fps, but I didn’t know how to make the graph and I didn’t have time to make more land as I am still new to the 3d version of unity.

With the speedometer I am happy how it come out but I really wanted it to be in a more 3d view and maybe have it where you can actually drive a car in 3d but I don’t know how to import cars into unity yet but I did learn how to convert meters per second into miles per hour which is quite nice to know if I ever wanted to make a racing game.

So overall I’m happy with the work I have done because I have learned a lot.