Reflective statement for Game Specialism 1

I feel I did well for this module, however I think I could have done a lot better.

Some of the issues I feel I could have improved on are:

Around the start of the year, I got a prescription for ADHD medication. I think this helped me keep focused but it didn't stop my bad habits from inhibiting my work. I should work harder to get rid of these bad habits and stop procrastinating.

When completing the shader brief, I found it very hard to demonstrate I understood the code, I struggled to find anything to edit to make it unique for me. So instead, I decided to write my own shader from scratch however that meant deviating from the brief somewhat.

And I procrastinated too long on completing the final brief, I could have (and should have) completed it weeks ago and done a fourth as well.

Writing the explanations of each project was also difficult for me, I struggled to write it in a way that felt chronological and easy to understand, a lot of it ended up as detached sentences explaining each little bit of it.

However I felt I did very well with my code quality and research, I used good practices and optimised everything well.

If I were to do it again, I would focus on procrastinating less and getting more prompts done. I feel that the quality of the prompts I did complete is very good though.

**Tracking turret:** [link to repo](https://gitfront.io/r/a-person5660/uQfEuWtTvNN9/PP-AB-Tracking/), I am using GitFront to link these because I use my GitHub account to contribute anonymously to a handful of open source projects and the repo includes my real name.

The turret starts by collecting a list of every entity with the target script attached to it, then it goes through the list to find the closest entity in the list which it saves as the target. The turret then gets the current velocity and direction of the target it wants to shoot, divides the distance between the turret and the target by the speed of the bullet to get the approximate impact position. However, this isn't fully accurate since it assumes the target will stay the same distance away, if it were to move closer than the turret would overshoot and miss. To fix this, I run the calculation multiple times but with the target position as the previously predicted position. This increases its accuracy with no performance impact.

There are better ways to do this with more complicated math that doesn't iterate and produces a fully accurate result, however I am bad at math and this solution is still good enough and won’t affect performance, so I did it this way.

The turret decides to shoot based on how close the turret's angle is to the angle it wants to be aiming at, if its close enough then it will fire.

The arraySearch bool exists because FindObjectsByType can be performance intensive with a lot of objects on larger games where performance is a concern (a similar function is causing a lot of lag on an open source game I have contributed to, hence I included it in the script even though I didn't make use of it) you may want to use a timer to only run it every couple of frames or run the search every time a target is destroyed or spawned and then have the turret check the array every frame. For this small number of entities, it's perfectly fine to leave running all the time.

**Edge Shader:** [link to repo](https://gitfront.io/r/a-person5660/9vXykCr9Mqwt/PP-AB-EdgeDetection/)

In the shader project I use two shaders, one for outlines and one for applying a benday dot filter over shadows, the brief was just to do an outline however after completing it by going through a tutorial I realised there wasn't much to improve on, so i decided to write my own shader from scratch with no tutorials which is what the shadow shader is. To make it i only used unity’s documentation and a couple forum posts to help fix some bugs and come up with the generated benday dots themselves (i could have just used an image/texture but i wanted to have more variables so i looked into doing this within the shader graph).

The Sobel edge detection shader uses a shader graph to do the bulk of the work but has some scripts to assist it.

The NormalsRenderFeature.cs script makes a copy of the depth normal texture which I can access in the shader graph.

The DecodeDepthNormals.hlsl script is called by EdgeDetectionOutlinesInclude.hlsl to decode and return the depth and normal from the texture it is provided.

EdgeDetectionOutlinesInclude.hlsl applies the Sobel filter itself, collecting the data for each pixel and its surroundings and doing the calculations to decide whether it's an edge or not.

ShadowRenderFeature.cs is a ScriptableRendererFeature with a ScriptableRenderPass contained within it, it extracts the \_ScreenSpaceShadowmapTexture (a built-in URP shadow texture) and assign it to a globally accessible shader property called \_ShadowScreenSpace. This makes it usable in shader graphs/other custom shaders.

I then pass it through a custom shader graph which turns it into a black/white image to use as a mask over a set of benday dots created using the Voronoi noise function, then I apply a colour and fade to it then slap that over the top of the image.

**Speedometer:** [link to repo](https://gitfront.io/r/a-person5660/SFmWqL1Z5baQ/PP-AB-Spedometer/)

The speedometer script can be attached to any object as it gets everything it needs from serialized fields, for the UI speedometer i have attached it to the “Dial” object, for the physical speedometer it is attached to the main cube that everything is parented to.

The dial works by getting the maximum and minimum speed marked on the speedometer, then it gets the angle that the dial should be facing when its at the minimum and maximum speeds. Once it has those it uses the MathF.Lerp function with the player's current speed divided by the absolute difference between the minimum and maximum speed marked on the speedometer. I additionally clamp this number between 1 and 0 because otherwise the dial would continue to rotate beyond the limits of the speedometer itself which would look off. I used local rotation to make sure that the dial would always rotate correctly, even if the object it was parented to was rotated.