Games Specialism Portfolio:

https://stulsbuac-my.sharepoint.com/:u:/g/personal/s4227106_lsbu_ac_uk/EYKQgJDFB0NNmtvVUdK4rXIBFp2cXjt42xOJ2v7zxVIZEQ

Brief 1: **Ingame FPS Counter**

Creating a basic FPS (Frames Per Second) counter in Unity helped me learn more about how games work and how to use Unity in a better way. Even though this was a small project, I learned useful skills that I can use in bigger projects later on.

The FPS counter shows how well the game is running by displaying how many frames are drawn every second. I used Time.unscaledDeltaTime in the script, which gave me the time between each frame without being affected by time scale changes. This helped the counter stay accurate even if the game speed changed. I also used the OnGUI() function to draw the FPS text on the screen, and I learned how to use GUIStyle to change the font size, color, and position of the text.

One of the most helpful things I learned was how to smooth the FPS value using a simple math formula:
deltaTime += (Time.unscaledDeltaTime - deltaTime) * 0.1f;
This stopped the number from jumping around too much and made it easier to read.

At first, I had trouble placing the text in the right spot on the screen. I also didn't understand how the Rect system worked, but after messing around with the values, I understood how to control the position and size of UI elements in Unity using screen width and height. This is a system that I know will be very useful.

This project also helped me understand how to write script way cleaner and to stay more organized. I learned how to separate logic into the Update() and OnGUI() methods and why it's important to keep the UI code simple and fast.

Overall, making this FPS counter helped me improve my Unity scripting, learn more about how games measure performance, and understand how to show helpful information to the player.

Brief 2: **Instanced Scrolling Material**

Making this scrolling script in Unity helped me understand how textures work and how to make them scroll to look like flowing water or moving surfaces. This project taught me a lot about shaders, UVs, and materials.

The goal of the script was to move the UVs of a texture over time, which makes it look like the texture is moving. I believe this may be a common effect used for the creation of water, lava and even things like conveyor belts etc. I used Time.deltaTime to make sure the scrolling speed stayed the same no matter the speed of how much frames the game was running. I also made the scroll direction and speed a variable so it could easily be changed in the Unity Editor.

A big part of this project was learning how to scroll the texture without creating memory problems. At first, I used rend.material to change the texture offset. It worked, but then I noticed the game was using more and more memory when many objects used the script. I found out that Unity makes a copy of the material every time I change it like that, and those copies stay in memory.

To fix the problem, I learned to use MaterialPropertyBlock. This lets me change the material's values on the object without creating new copies. It was a little confusing at first, and I forgot to call GetPropertyBlock() before setting the values, so nothing happened. Once I figured that out, everything worked and memory stayed clean.

This project also helped me understand how to use RequireComponent to make sure the script is only added to objects that have a renderer. It made my code better and helped me to stop errors from happening.

In the end, I felt more confident working with materials and shaders in Unity. This scrolling effect is simple, but I now understand how to use it properly and how to avoid memory leaks. Another thing I learned, was that even very small problems could cause the whole project to not work, and that I should always look carefully whilst scripting.

Brief 3: **Advanced Edge Detection Shader**

Working on the edge detection shader in Unity helped me learn more about how post-processing effects and shaders work. The goal was to make the game look more like a

drawing by showing outlines on the edges of objects. I wanted the shader to find the edges and highlight them using a color and thickness I could change, which I managed to create mostly.

I used Unity's Shader Graph to build the edge detection effect. Shader Graph was a bit tricky at first, but I liked how it let me see what I was building in real time. I used nodes like "Scene Depth" and "Saturate" to try and find the edges between objects and to differentiate them.

I checked the camera settings, render pipeline settings, and made sure I used a post-processing volume. Still, the effect didn't show in the game view. I tried changing the order of nodes and playing with the texture itself, but I still couldn't find the problem. It's something I haven't solved yet. I believe it may have something to do with the materials it self not being able to pick up the shader, or maybe there is something I need to attach the shader to.

Even though the shader didn't fully work, I still learned a lot from the process. I now know more about how post-processing works, how to use the Universal Render Pipeline, and how to use custom materials and passes with Shader Graph. I also got better at organizing my shader nodes and testing changes step by step.

This project helped me improve my problem-solving , aswell as learning more about rendering in Unity. Even though it didn't fully work yet, I'm proud of what I learned and hopefully I can figure out the problem.

Overall, I am proud of the experience I gained from this course, as I started off with almost 0 knowledge with Unity and gained alot due to the Unity Documentation tab which was shown in class previously, to help me figure out the use of many functions and when would be the most appropriate time to use them.

https://stulsbuac-my.sharepoint.com/:u:/g/personal/s4227106_lsbu_ac_uk/EYKQgJDFB0NNmtvVUdK4rXIBFp2cXjt42xOJ2v7zxVIZEQ