Reflective Document

Brief 1 (In Game FPS Counter)

For the FPS counter I basically had to create an FPS counter that could be used in a VR game during development to make sure that there was a consistent frame rate so it doesn't induce nausea. The actual FPS counterpart I found easy as it's just a counter that counts up every frame, adds the total to a list and then gets reset every second. So, for mine you could have a history of 100 seconds and for each of them it would store the FPS for that second. The harder part was to display this to a graph to show on the UI. But you can't do this natively. So, in order to get it to work, I had to use a line renderer which takes in the data from the FPS counter and adds it to a list of points. Then I had a separate camera which only rendered the Graph to a render texture which I then passed into an image on the canvas.

I found this quite interesting to actually get to work and function how I wanted, as I had to fiddle around with the Universal Render Pipeline to get the texture to be transparent apart from the line. Which I hadn't used render textures before apart from in the Racing game I made for 3D Level Design in the last semester, which was just showing the output of a rear viewing camera.

The FPS counter could also be used for game benchmarking as you could keep a list of all the frames over a sequence, and then once the benchmark has finished you could also calculate the percentage lows and the average over all the scenes. Which can be useful to see how stable the game is and the hardware its running on.

Brief 2 (Edge Detection)

For the Edge Detection I had to create a shader that detected the edges of game Objects and highlights them to create an illustrated art look. While doing this I learnt how to use the shader graph and a little about the Universal Render Pipeline Render Features which this relies on. It uses the game objects normal, scene depth and the angle of edges into account when calculating the outlines, so it will give the best outlines for each shape allowing more flexibility on what you can use.

The hardest part of this was probably coding the Render Feature to capture it and then pass that into the shader graph to be processed. Due to the fact that there isn't much documentation on it. I mostly found tutorials on YouTube about it and even than some of them didn’t work as expected. The tutorial I found was from a YouTuber called Robin Seilbold which went over how to use the shader graph and apply effects to it, then add it back to the screen.

This process can also be used to create pixel shaders or change how shadows and lighting works in the engine as well. Which means that the whole pipeline used in Unity can be changed depending on what effect you are going for in your game or project.

Brief 3 (Audio Frequency Detection)

For the Audio Frequency Detection at first, I tried to merge a couple briefs together, being the shuffle, and this one. So, I basically made an audio player with media controls, and the audio frequency detection. But I had to back track as this wasn't following the brief exactly. So, I went back and tweaked the audio frequency detection to follow the brief.

While doing this brief I learnt how to sample the audio playing from an audio source and split it into different bands. Once I had the bands, I found it very easy to link it up with unity events and get other things firing when different frequencies hit different thresholds. This could easily be expanded upon to have information passed through the event about how strong the frequency was or how much it went past the threshold.

In my example scene I just got it to change the hue of objects depending on what frequency got hit, so the low frequencies might be a red, where the higher ones could be a green. Thich could also be used in rhythm-based games which could cause platforms and other objects to move depending on the beat of the music.

Brief 4 (Name Generator)

In the name generator brief you were tasked to generate star names for 100 billion different names. But they must be pronounceable, and there can't be any obscenities, and there must be no duplicates. To do this I had to write a script that checks for certain conditions in the English language and apply different rules to them. Such as I before e except after c, the majority of words follow this in the English language so it’s one of the rules that needed to be implemented.

I also had to use a list of obscenities that I got from the internet to check if the generated word is in that list. If it is then it will just return and not add it to the list, but instead just print a warning to the console saying it found a word that couldn’t be added. This is also the same if a word it generates is already in the list of words generated. So, it doesn’t add duplicate words to the list.

This could be used for other word generators and not just a generator for star names. As you can easily change the parameters and get different words depending on how you use it. Meaning you could use it to generate a new language, but it would require a lot of tweaking to get right.

Summary

After doing these briefs I have learnt a bit more on using the components in Unity to generate and display things to the screen. But the thing I have learnt the most is how to make custom render features which can be used to draw shaders to the screen and get them to affect the whole scene or just certain objects. Which can be very powerful to give different feels to games as you can apply different effects to the same game.

All of these briefs can also be applied to other games and projects which could be useful to develop other tools to help with game development. But also add in features into pre-existing games to monitor performance, or change the visuals of the game or project.