**Game Specialism: Programming Reflective Form By Charlie Cross**

**Speedometer Brief**

The first brief that I chose to do was the speedometer brief. To begin I made two assets, pixel art of a car and pixel art of a road as I wanted to use my own assets, this allowed me to further develop on artistic skills within games.

The first script that I went about making was the rigidMovement script where I programmed the car to only move vertically and used the rigidbody velocity for the speed of the car. To create a speedometer in miles per hour I had to convert the units from units per second (the units used within the unity engine to measure speed). Having had help from Paul I found out that units per second were equivalent to meters per second, so it was a matter of converting from meters per second to units per second.

To do this conversion I multiplied the rb.velocity.magnitude (the velocity of the car in unity) by 60 twice to get the units per hour and divided the whole thing by how many meters were in a mile to get miles per hour. I then used debug.log to check if the conversion worked in unity, which it did.

I used two triggers that when driven over, teleport the player back to the far back trigger to give the illusion that the player is driving on an infinite road.

The next step was getting the speed to appear on screen in real time. I made a script for the speedometer and used an image game asset so that there would be a text box on screen. I declared a rigidbody variable and assigned the Player to it by dragging and dropping my player in to the inspector. I struggled with this as first as I had missed out the syntax of saying that it was rigidbody2D whereas I was using the 3D version at first without realising. I then used the equation for the conversion of the speed and displayed it in the box to track the speed in real time.

I had never made anything like this before, it has improved my knowledge of how rigidbody physics work in unity and displaying the speed of an object on screen. Developing my understanding of unity greatly.

**Radar Brief**

The next brief that I chose to take on was the radar brief. I began about making the radar and started to learn Euler angles, which I have had difficulty using in the past. At first, I thought of using OnCollisionEnter2D, so that when the rotating radar line collided with an invisible enemy, it would turn them red so that it was visible in the radar. However, I could not figure out how to get it to work correctly and started to look for alternative methods.

After researching unity documents and watching multiple videos, I wanted to try and use a ray cast. I managed to have the ray cast follow the radar line and I programmed it to write in the console if the ray cast collider was not null (if the ray cast collided with an enemy). When programming this, I was having great difficulty when trying to get the ray cast to rotate against the radar line. I had to find a mathematical conversion that I found on the internet and call it as a method, after which the ray cast worked perfectly.

To change the enemy’s colour, I had it so the enemies starting colour would have an alpha of 0 so they were invisible. I called the hit method in another script so that when an enemy was hit by the ray cast, their colour would turn to red for 1 second and then the alpha would go back to 0.

Instead of having the enemies be static I decided that I would code an enemyBehaviour script so that they would move in random directions. I used Random.insideunitsphere so that their direction would change randomly whilst setting a timer using Random.Range that would randomise the timer value so the enemies would move in random directions and random times between 1 and 3 seconds.

I had run out of time on this brief and had spent most of the time trying to get the radar to work correctly using a ray cast. If I had more time, I would have liked to have made the radar a minimap that would sit in the top right of the screen while having a player that can move, trying to navigate through invisible enemies. Having the potential to be used in a game in the future.

**Instanced Scrolling Material Brief**

For my final brief I decided to do the instanced scrolling material brief, as I thought it could be useful to develop for future use in any games that I want to develop.

To start, I created water pixel art in photoshop to use as my texture. Developing my artistic skills also.

Originally, I planned to do this brief in a 2D environment, but I wanted to try and use a 3D environment instead, something that I have less experience in.

To get the scrolling material to work, I had to program a script that scrolls through the uv coordinates of the texture vertically, to give the illusion of water flowing. This could potentially be used in the background design of 3D games.

I first declared a public material variable and assigned the water texture to it in the inspector so that the cube that I spawned into unity would have the texture wrapped around it.

I declared the speed of the flow of water as well as a vector 2 of the flow direction and of the uv coordinates.

I set the uv coordinates variable as the water texture offset in the start function, so that scrolling through the coordinates was possible.

Then in the update function I set the uv coordinates variable to scroll through the uv coordinates by using +=.

Finally, by using SetTextureOffset, to set the offset of the uv coordinates of the texture, and the variable that I had just declared, the material scrolled nicely at a moderate speed. Giving an illusion of water flowing.