**Reflection Document - George Irtelli**

**Speedometer**

Starting with the speedometer brief, I began by completing the objective in a 2D space as that was more comfortable and natural to me. In the brief I started by making a simple movement system using code from other projects and a quick play area with bounds within the unity editor.

Now with the systems in place, I had to measure the player’s current speed and represent that to UI elements. Firstly, I was using the translate method to move the player which I had to change to velocity to read speed values, the speed values taken were from the rigidbody component using velocity.magnitude . To display this as text I made a separate script to change the contents of a textbox. The brief stated to represent speed as miles per hour, while speed values are given as Meters per second due to the physics system taking values in the metric system, to convert m/s to MPH I would multiply by the constant 2.23694f. To transfer the speed value between scripts I would learn how to make references to functions in other scripts by declaring them as variables. Once the value for speed in MPH has been passed to the correct script it would be displayed in text form in the UI.

For the dials there was no need to convert the speed into MPH as the dial only needed to move in a 180 degree range, to do this I would take the current speed of the player and divide that by the defined max speed to get a value between 1 and 0, then multiply that value by 180 to tell the “Dial pivot” object what its rotation is.

For the 3D version of this application the above concepts apply identically with the only changes needed to be made was to the player’s movement system which required more research than expected resulting me to use torque to turn the car object, then using a massive plane as the ground, camera control using the LookAt() method to keep the car in frame and adding a trail renderer to give more sense of direction the above UI elements could be added.

**Hit A Moving Target**

This brief required me to make an AI agent to observe a player object and fire a projectile at the point they will intercept. To do this I made a simple area for the player to move in with areas for cover and gave the player object a way to move using velocity vectors, for the AI I gave it a way to detect where the player was constantly by giving the AI the players position vector, the first thing I did was giving the AI the ability to detect line of sight using Raycast2D, an invisible laser would be shot from the enemy AI to the player, if it intercepted a wall beforehand it would do nothing, if the Raycast hit the player the enemy would shoot a projectile directly at the player. To allow the AI to roughly predict the player, I would measure the distance between the AI and the player using Vector3.Distance() once the distance between the player and the enemy was known the AI would calculate the time it would take the bullet to reach the player as the bullet’s speed would have been defined and using speed = distance / time where time would be rearranged to be solved. Given time, the AI would multiply the player’s current speed by x seconds and add that distance vector to the player’s current position which gives the AI a predicted position.

To get the AI to shoot the bullet I had to rotate an object towards the position using Mathf.Atan2() then Instantiate an bullet object with constant forward speed, rotated towards the predicted position.

This solution only works if the player remains at constant speed maintaining constant displacement from the AI once the projectile if fired. To fix this I needed to solve the predicted location simultaneously solving both sides of the equation for position as a Vector2 although this was not possible for me in the end, instead I came up with a little bit of a shortcut by dilating the bullets speed such that they would always intercept the target which made it so multiple AI agents would not work as the bullet dilation work around would only work for one AI agent at a time and the other two tanks would miss.

**Shuffle**

Shuffle required the creation of a random number generator for songs while not generating the same number twice. To solve this I made a script generate a random integer number between 0 and 9 and made a 3 step buffer which would remember the last number generated, if the number generated was in the buffer the script would restart the function and generate a new number, if the number was not in the buffer, the contents would move down by 1 where the last number is overwritten and the new number generated is added to the buffer. This process is repeated by a defined times by the user using UI buttons and is repeatable using a reset button. Instead of playing whole songs since its purpose is non-repeating generation, I used a sprite sheet found online and learned how to cut out the numbers into its own sprites to be generated onto the screen. To be able to reset a temporary collider appears on screen and destroys all the generated numbers.