Radar Project

For the radar project, I went in with no experience or knowledge with unity, so I needed to figure everything out as I went along. I started out by creating a pill shape for the character and a plane for the floor, and changing the floor to grey to create contrast between the ground and the player. I needed the player to be able to have basic movement before I started with the radar, so I created the MoveScript and did this by translating the player depending on the input pressed by the key. I also took the mouse position and found the difference between the mouse position from the current and previous frame and used this to rotate the player left and right. I also added an if statement that detects whether the left shift key is being pressed, and increases the move speed if it is being pressed. I added this sprint function so that I could see different speeds on the radar to check if it was working correctly.I did not need the ability to look up and down so I did not implement this, but in the future i would implement a more dynamic movement, as it allows for testing of the radar to be much easier.

For the Radar, I created a script called RadarContact, which was attached to all of the cubes that would appear on the radar. This script takes the object's relative position from the player, and sends that position to a second script called ping, which is attached to the UI ping on the radar. The RadarContact script also checks if the object is out of the radar's range and ensures that the sent position will remain on the edge of the radar. The ping script then receives this input and transforms the position of the icon into the correct location on the radar. For extra credit, The ping script also checks if the icon is on the border of the radar, and changes the colour of the icon to red to show that the object is out of the radar's range.

Speedometer Project

For the speedometer project, I started off by copying the movement script from the radar project, but had difficulty finding the velocity of the player, as the function that I was using to find the velocity needed a rigidbody. This meant that I had to create brand new movement scripts from the radar project, as I needed the player object to have a rigidbody, therefore I needed to change the way that the player movement worked. I started off by creating the PlayerMovement script, and in the update function it obtains the movement input from the keyboard and the mouse input and stores them in Vectors. I made a movePlayer function which is called within the Update function, and finds the direction and magnitude that the player needs to move depending on the input, speed and frame rate, and then adjusts the rigidBody velocity according to this. I also made a turnPlayer function which rotates the player depending on the mouse input and turn rate. I made a second script called CameraMovement which was attached to the camera and allowed dynamic camera movement such as looking up and down and zooming in and out.

For the speedometer I started off by creating a Text mesh pro UI element, to display the player's speed. I decided to add the speed script into the PlayerMovement script, so I created a speedometer function in which I took the magnitude of the velocity of the rigidbody and displayed it onto the text UI as an integer. This number was meters per second, so I needed to multiply the number by 2.23694 to convert it to miles per hour.
For extra credit I needed to make a rotary dial to display the speed, so I designed a speedometer UI element and needle. I found the maximum and minimum angle that the needle needed to be and put it into a euler angles function with the speed and maximum speed so that the needle will rotate depending on the speed. This function is called in the update function.

FPS project

For the FPS project I needed the same dynamic movement that I made in the Speedometer project, so I started by copying the movement for the player and the camera. The FPS display was a Text mesh pro UI and again I decided to create the function inside the player movement so as to not create too many unnecessary scripts.

Getting the frames per second was simple as it was 1/Time.deltaTime, but displaying the new frames per second every frame would be too fast and hard to read, so I needed to average the FPS over the last second. I struggled with this at first trying to find a function to do this, but then realised I could simply add the frame rate up over a second, and then divide it by the amount of frames counted in that second. This allowed me to display an average frame per second over the second, which made it more readable and useful within the game.
As the frames per second should only be displayed in build versions of the game, I made a build mode toggle when you press the B key. Now the frames per second counter will only show when the Build mode is enabled.