I started on the Second Brief called Speedometer. It allows you to know how fast the car is driving by showing the Kilometres by hour on the left side of the screen with a minimum of 0 and a maximum of “260”. It goes over that range when you fall off the pathway. I followed a tutorial so I could have a Rigidbody to find out if the speedometer works. This can be completed using the scripts below.

Create a script allowing the player to move down the road and interact with things, such as obstacles. “PlayerController” allows you to move forward, horizontally and vertically down the road, and it also allows the game to know the player has moved. The speedometer should add force to a vehicle so that the speedometer knows you’re moving.

The second Script will be the “FollowPlayer”, which allows the Main Camera to follow the player a certain distance away from the Vehicle so you can know what happens to the car you are always driving.

The last Script will be the “Speedometer”. Scripts that allow me to make the Speedometer on the left side work by giving a Rigid body to latch on to and a minimum and maximum speed so the character can’t go beyond it and go too fast. I also add the text and image into the script, so they follow along with the car when it’s driving.

To make a speedometer happen, you must first have a working vehicle and its Rigidbody working, then a picture of a Speedometer and an arrow that goes up and down when speeding with the car, which then you go on the hierarchy and right click and find UI and click the raw image that allows you paste your pictures then you adjust however you like. After that, you make a speedometer script, and in there, go to the very top where it says using system or engine, go on a new line and write (using UnityEngine.UI) to tell the game that you are referring to it in your project. Then, in the public class with the parentheses at the bottom, press enters and write public Rigidbody target (Car), to let the engine know that the speed you want to track is the car.

Adds the public float and calls it max speed, which will be zero, so you see the speed go up the on your speedometer. Then you add the two lines underneath, which will also be a public float called min speed arrow angle and Max speed arrow angle so that the car doesn’t exceed the speed limit on the speedometer and doesn’t go below your speedometer either. You add a [Header (UI)] just like this so you can refer to UI in the game, and below that, you add a public float that can refer you to the text where you have the speed label that tells you how many kilometres you’re going at. Public RectTransform arrow so it can refer to the hand in the UI, an indicator that follows along the numbers in the speedometer, so you see how your speed is in the game. You then add a private float to the speed game with the numbers and f to let it know that you are using a float. Then in the private void update, which might need to be typed in. In the private void (speed = target.velocity.magnitude \* 3.6f;), which will convert that into kilometres, tell you about the momentum in the Rigidbody and control how the car goes.

Add an if(){}, which looks like this. Which means if I do this, then do that. In the first bracket, you add speed Label != null; you do not refer to any object, in the parentheses. You add a speed Label. text((int) (integer) + “km/h; which means you refer to the text in the speed label image and another if() inside the brackets, you add an (arrow != null) underneath that you add arrow.localEulerAngles (allows you coordinate with the parents object rotations) = new vector(0, 0, mathf.Lerp(minspeedArrowAngle, maxspeedArrowAngle, speed/ maxSpeed)­) allows you to move or change the values over some time. If your game doesn’t show the arrow’s movement on the speedometer, go to the player controller, add rb. add force in front of the C# that allows your player to move forward.