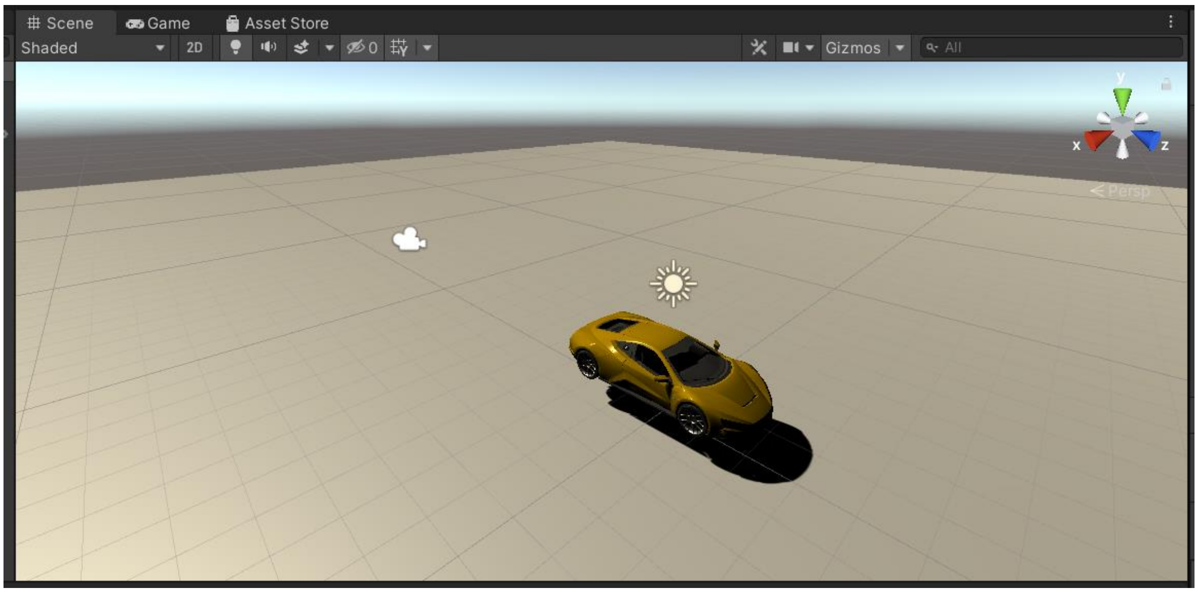
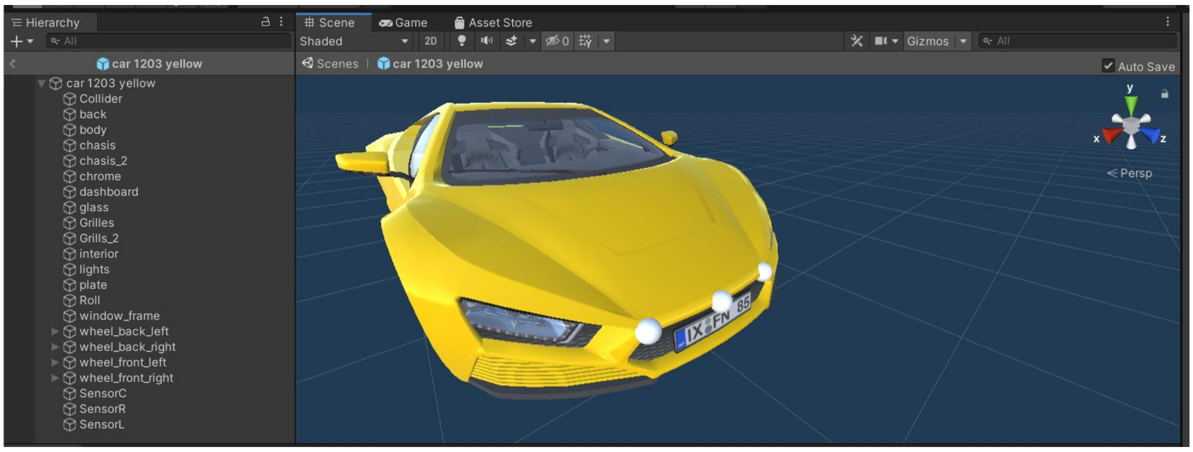
Basic perception and reaction in Unity

Step 1 – Download, unzip and open the “Unity car - User input and basic physics” file from canvas as this will be your starting point.



The majority of features that we will require are already implements we will simply build on this.

Step 2 – Open the prefab asset (the background will turn blue indicating this) and add 3 spheres in the hierarchy calling them SensorC, SensorL and SensorR then place them along the front bumper of the car as seen below. Note: ensure the rotation values are set to 0 and the scale values to 0.1.

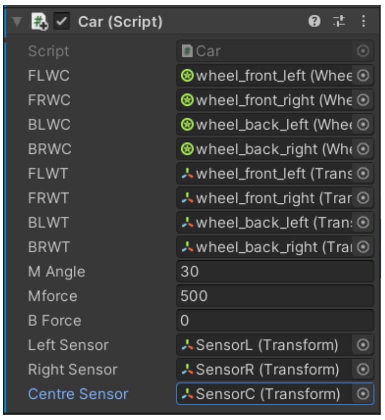


```
car 1203 yellow
  car 1203 yellow
    Collider
    back
    body
    chasis
    chasis_2
    chrome
    dashboard
    glass
    Grilles
    Grills_2
    interior
    lights
    plate
    Roll
    window_frame
    wheel_back_left
    wheel_back_right
    wheel_front_left
    wheel_front_right
    SensorC
    SensorR
    SensorL
```

Step 3 – Open the car script in the asset manager then add the following lines for the sensors and link these to the sensors on the car in the inspector interface (you will need to return to the scene to do this).

```
    [SerializeField] private Transform LeftSensor;
    [SerializeField] private Transform RightSensor;
    [SerializeField] private Transform CentreSensor;
```

We now have the sensors linked to our code to start using them.



Car (Script)

Script	Car
FLWC	wheel_front_left (Whee
FRWC	wheel_front_right (Whe
BLWC	wheel_back_left (Whee
BRWC	wheel_back_right (Whe
FLWT	wheel_front_left (Trans
FRWT	wheel_front_right (Trar
BLWT	wheel_back_left (Tran:
BRWT	wheel_back_right (Trai
M Angle	30
Mforce	500
B Force	0
Left Sensor	SensorL (Transform)
Right Sensor	SensorR (Transform)
Centre Sensor	SensorC (Transform)

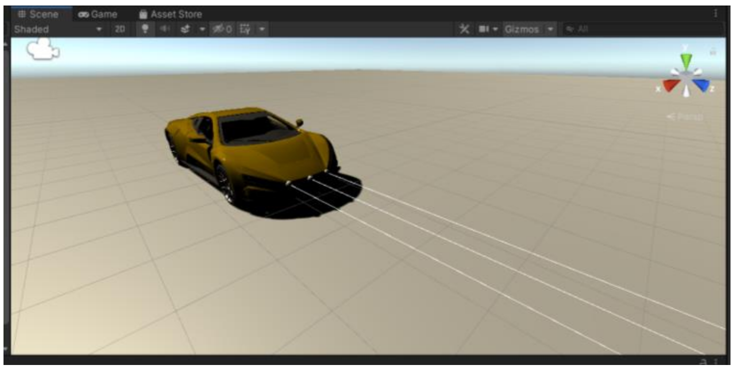
Step 4 – Write the following function to facilitate the creation of raycast sensors for each of the sensors we have just put on the car. We will soon call this function for each of the sensors.

```
private bool sense(Transform sensor, float dist)
{
    RaycastHit hit;
    if (Physics.Raycast(sensor.position, sensor.TransformDirection(Vector3.forward), out hit, dist))
    {
        Debug.DrawRay(sensor.position, sensor.TransformDirection(Vector3.forward) * hit.distance, Color.yellow);
        return true;
    }
    else
    {
        Debug.DrawRay(sensor.position, sensor.TransformDirection(Vector3.forward) * dist, Color.white);
        return false;
    }
}
```

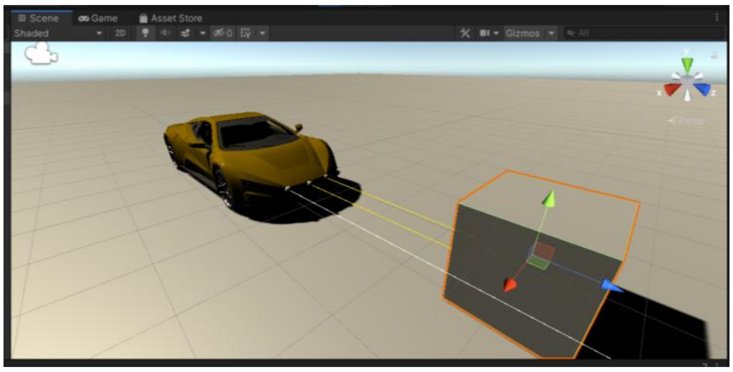
Step 5 – add the following calls to the FixedUpdate function.

```
        sense(CentreSensor, 5);
        sense(RightSensor, 5);
        sense(LeftSensor, 5);
```

Now save and run the game with the gizmod option on you should see something similar to the screenshot below.

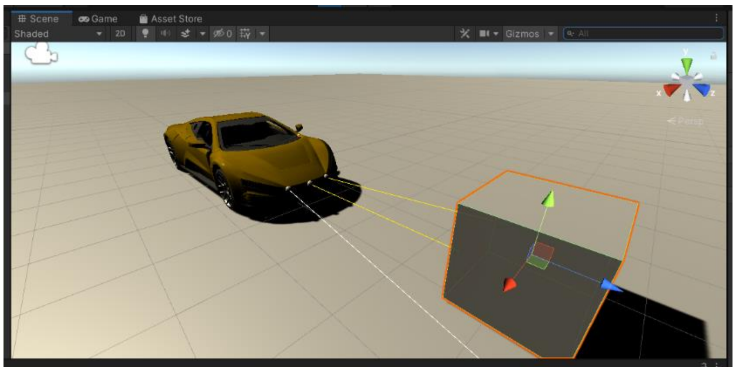


Step 6 – place a cube in the scene within view of the sensors and see what happens, the sensors should turn yellow to indicate an object has been detected.



Step 7 - Our sensors are working but they are not terribly well angled so we need to change the angle of them which we will do using the following code in the start function.

```
void Start()
{
    LeftSensor.transform.Rotate(0, -10, 0);
    RightSensor.transform.Rotate(0, 10, 0);
}
```



Step 8 – edit the code so that the car automatically steers to avoid the object based on the senor values. Since the sensors return a bool this a relatively simple task. i.e. if the centre sensor and the left sensor are triggered turn right. The obscured code below provides a hint of what is required but this will require some work to get right. Feel free to add more sensors if this helps.

```
private void HandleSteering()
{
    steerAngle = MAngle * horizontalInput;
    [illegible]
    {
        [illegible]
        {
            [illegible]
        }
        [illegible]
        {
            [illegible]
        }
        [illegible]
    }

    FLWC.steerAngle = steerAngle;
    FRWC.steerAngle = steerAngle;
}
```