```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class RobotController : MonoBehaviour
{
    // naming constraints do not change
    [SerializeField] private WheelCollider FLC;
    [SerializeField] private WheelCollider FRC;
    [SerializeField] private WheelCollider RLC;
    [SerializeField] private WheelCollider RRC;
    [SerializeField] private Transform FLT;
    [SerializeField] private Transform FRT;
    [SerializeField] private Transform RLT;
    [SerializeField] private Transform RRT;
    [SerializeField] private Transform FRS;
    [SerializeField] private Transform L1S;
    [SerializeField] private Transform L2S;
    [SerializeField] private Transform L3S;
    [SerializeField] private Transform R1S;
    [SerializeField] private Transform R2S;
    [SerializeField] private Transform R3S;
    [SerializeField] private Transform ORS;
    [SerializeField] private float angle_x;
    [SerializeField] private float angle_y;
    private void AdjustIndicator(Transform sensor, float x_angle, float y_angle,
float z_angle)
    {
        sensor.transform.Rotate(x_angle, y_angle, z_angle);
    }

    [SerializeField] private bool EndOfRoadF;
    [SerializeField] private Rigidbody RigdBody;

    private void Start()
    {
        RigdBody = GetComponent<Rigidbody>();
        s1dist = 8f;
        s2dist = 8f;
        s3dist = 8f;
        rcfrdist = 8f;
        float s1x = 0; float s1y = 23; float s1z = 0;
        float s2x = 25; float s2y = 25; float s2z = 0;
        float s3x = 14; float s3y = 55; float s3z = 0;
        AdjustIndicator(FRS, 15, 0, 0);
        AdjustIndicator(L1S, s1x, -s1y, s1z);
        AdjustIndicator(R1S, s1x, s1y, s1z);
        AdjustIndicator(L2S, s2x, -s2y, s2z);
        AdjustIndicator(R2S, s2x, s2y, s2z);
        AdjustIndicator(L3S, s3x, -s3y, s3z);
        AdjustIndicator(R3S, s3x, s3y, s3z);
        AdjustIndicator(ORS, 100, 180, 0);
        EndOfRoadF = false;
        motorForce = 450f;
        maxSteeringAngle = 25f;
        brakeForce = 0f;
    }
    [SerializeField] private float brakeForce;
    private void ApplyBrakes()
    {
        brakeForce = 2000f;
        FRC.brakeTorque = brakeForce;
        RLC.brakeTorque = brakeForce;
        FLC.brakeTorque = brakeForce;
      RRC.brakeTorque = brakeForce;
    }
    private void UpdateWheelsArea(WheelCollider wheelCollider, Transform trans)
    {
        Vector3 pos;
        Quaternion rot;
        wheelCollider.GetWorldPose(out pos, out rot);
        trans.rotation = rot;
        trans.position = pos;
    }
    [SerializeField] private float motorForce;
    private void HandleCarDrive()
    {
        FLC.motorTorque = motorForce;
      RRC.motorTorque = motorForce;
        FRC.motorTorque = motorForce;
        RLC.motorTorque = motorForce;
    }
    [SerializeField] private float steerAngle;
    [SerializeField] private float maxSteeringAngle;
    private void HandleCarSteerWheel(float direction)
    {
        steerAngle = maxSteeringAngle * direction;
        FRC.steerAngle = steerAngle;
        FLC.steerAngle = steerAngle;
    }
    private bool Sense(Transform sensor, float dist, string layerName)
    {
        int layerMask = LayerMask.GetMask(layerName);
        if (Physics.Raycast(sensor.position,
sensor.TransformDirection(Vector3.forward), dist, layerMask))
        {
            Debug.DrawRay(sensor.position,
sensor.TransformDirection(Vector3.forward) * dist, Color.green);
            return true;
        }
        else
        {
            Debug.DrawRay(sensor.position,
sensor.TransformDirection(Vector3.forward) * dist, Color.red);
            return false;
        }
    }
    private void UpdateCarTyres()
    {
        UpdateWheelsArea(FLC, FLT);
        UpdateWheelsArea(RRC, RRT);
        UpdateWheelsArea(FRC, FRT);
        UpdateWheelsArea(RLC, RLT);
    }
    [SerializeField] private float s1dist;
    [SerializeField] private float s2dist;
    private void AvoidObstacles()
    {
```

```
    if (Sense(R1S, s1dist, "Obs"))
    {
        HandleCarSteerWheel(-1);
    }
    if (Sense(L1S, s1dist, "Obs"))
    {
        HandleCarSteerWheel(1);
    }

}
private void BreakIntegrator(int integ)
{
    brakeForce = 200f * integ;
    FRC.brakeTorque = brakeForce;
  RLC.brakeTorque = brakeForce;
    FLC.brakeTorque = brakeForce;
    RRC.brakeTorque = brakeForce;
}
[SerializeField] private float velocity;
private void AdjustCarAcceleration()
{
    if (velocity > 4 && motorForce > 0)
    {
        motorForce = motorForce - 5;
    }
  if (velocity < 4 && motorForce < 500)
    {
        motorForce = motorForce + 5;
    }

}
private void EndOfTrack()
{
    if (EndOfRoadF)
    {
        ApplyBrakes();
    }
}
[SerializeField] private float s3dist;
[SerializeField] private int EndOfRoadCntr;
[SerializeField] private float rcfrdist;
private void StayOnLane()
{
    bool senseShortLeft = Sense(L2S, s2dist, "Road");

    bool senseLeft = Sense(L3S, s3dist, "Road");
    bool senseShortRight = Sense(R2S, s2dist, "Road");
    bool senseRCFR = Sense(FRS, rcfrdist, "Road");
    bool senseRight = Sense(R3S, s3dist, "Road");
    if (!senseRCFR && !senseShortLeft && !senseShortRight)
    {
        EndOfRoadCntr++;
        if (EndOfRoadCntr > 5)
            EndOfRoadF = true;
    }
    else if (senseLeft && senseRight)
    {
        HandleCarSteerWheel(0);
        EndOfRoadCntr = 0;
```

```
       }
       else if (!senseRight)
       {
           HandleCarSteerWheel(-1);
           EndOfRoadCntr = 0;
       }

       else if (!senseLeft)
       {
           HandleCarSteerWheel(1);
           EndOfRoadCntr = 0;
       }
   }
   private void FixedUpdate()
   {
       StayOnLane();
       if (EndOfRoadF)
       {
           EndOfTrack();
       }
       else
       {
           HandleCarDrive();
           AvoidObstacles();
           AdjustCarAcceleration();
       }
       UpdateCarTyres();
       angle_x = ORS.eulerAngles.x;
       angle_y = ORS.eulerAngles.y;
       velocity = RigdBody.velocity.magnitude;
   }

}
```