Endless Car Driving (Unity)
Code Writing Guide

This guide uses the **exact code** provided by the teacher. Students write it part-by-part.

Date: February 05, 2026

Project goal

Make a simple endless driving car controller using Rigidbody physics:

- W/S = accelerate & brake
- A/D = steer
- R = restart the level
- Two scripts: **CarHandler** (movement) and **InputHandler** (input).

Scene setup (use your existing car model)

We will NOT change your 3D model. We only add physics + scripts.

1) Create a clean root object

- Create an empty GameObject named **CarRoot**.
- Make your model a child: **CarRoot/CarModel**.
- Rotate CarRoot so its blue axis (Z) points forward.

2) Add physics components

- Add **Rigidbody** to CarRoot.
- Add a **BoxCollider** (or multiple simple colliders) to CarRoot.
- Place the car on a ground plane with a collider.

> **Note:** CarRoot is the object that moves. CarModel is visuals only.

Script 1: CarHandler (write it in parts)

Create a C# script named **CarHandler** and follow the parts below.

Part A - Fields (variables)

Copy this first:

```
using UnityEngine;

public class CarHandler : MonoBehaviour
{
    [SerializeField]
    Rigidbody rb;
    float accelerationMultiplier = 3;
    float breaksMultiplier = 15;
    float steeringMultiplier = 5;
    Vector2 input = Vector2.zero;
```

In Inspector: drag CarRoot Rigidbody into **rb**.

Part B - Start + FixedUpdate

Add these methods next:

```
    // Start is called once before the first execution of Update after the MonoBehaviour is created
    void Start()
    {

    }

    private void FixedUpdate()
    {
        if (input.y > 0)
            Accelerate();
        else
            rb.linearDamping = 0.2f;

        if (input.y < 0)
            Brake();

        Steer();
    }
```

Tip: FixedUpdate is used for physics movement.

Part C - Update (keep as provided)

Add this exactly as provided:

```
// Update is called once per frame
void Update()
{
    Accelerate();
}
```

Important: Later (optional improvement): we can refactor this. For now, keep it exactly.

Part D - Movement functions

Add the 3 functions below:

```
    void Accelerate()
    {
        rb.linearDamping = 0;

        rb.AddForce(rb.transform.forward * accelerationMultiplier * input.y);
    }

    void Brake()
    {
        if (rb.linearVelocity.z <= 0)
            return;

        rb.AddForce(rb.transform.forward * breaksMultiplier * input.y);
    }

    void Steer()
    {
        if (Mathf.Abs(input.x) > 0)
        {
            rb.AddForce(rb.transform.right * steeringMultiplier * input.x);
        }
    }
```

Part E - SetInput + close class

Add this at the bottom, then close the class bracket:

```
    public void SetInput(Vector2 inputVector)
    {
        inputVector.Normalize();

        input = inputVector;
    }
}
```

Script 2: InputHandler (write it in parts)

Create a C# script named **InputHandler**.

Part A - Imports + field

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class InputHandler : MonoBehaviour
{
    [SerializeField]
    CarHandler carHandler;
```

In Inspector: drag the CarRoot object (CarHandler component) into **carHandler**.

Part B - Update loop (read keys and restart)

```
    void Update()
    {
        Vector2 input = Vector2.zero;

        input.x = Input.GetAxis("Horizontal");
        input.y = Input.GetAxis("Vertical");

        carHandler.SetInput(input);

        if (Input.GetKeyDown(KeyCode.R))
            SceneManager.LoadScene(SceneManager.GetActiveScene().name);
    }
}
```

Note: Axes used: Horizontal and Vertical (default: A/D and W/S).

Final check: full code (exact)

Your final scripts should match exactly (no edits).

CarHandler.cs

```
using UnityEngine;

public class CarHandler : MonoBehaviour
{
    [SerializeField]
    Rigidbody rb;
    float accelerationMultiplier = 3;
    float breaksMultiplier = 15;
    float steeringMultiplier = 5;
    Vector2 input = Vector2.zero;
    // Start is called once before the first execution of Update after the MonoBehaviour is created
    void Start()
    {

    }

    private void FixedUpdate()
    {
        if (input.y > 0)
            Accelerate();
        else
            rb.linearDamping = 0.2f;

        if (input.y < 0)
            Brake();

        Steer();
    }

    // Update is called once per frame
    void Update()
    {
        Accelerate();
    }

    void Accelerate()
    {
        rb.linearDamping = 0;

        rb.AddForce(rb.transform.forward * accelerationMultiplier * input.y);
    }

    void Brake()
    {
        if (rb.linearVelocity.z <= 0)
```

```
            return;

        rb.AddForce(rb.transform.forward * breaksMultiplier * input.y);
    }

    void Steer()
    {
        if (Mathf.Abs(input.x) > 0)
        {
            rb.AddForce(rb.transform.right * steeringMultiplier * input.x);
        }
    }

    public void SetInput(Vector2 inputVector)
    {
        inputVector.Normalize();

        input = inputVector;
    }
}
```

InputHandler.cs

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class InputHandler : MonoBehaviour
{
    [SerializeField]
    CarHandler carHandler;

    void Update()
    {
        Vector2 input = Vector2.zero;

        input.x = Input.GetAxis("Horizontal");
        input.y = Input.GetAxis("Vertical");

        carHandler.SetInput(input);

        if (Input.GetKeyDown(KeyCode.R))
            SceneManager.LoadScene(SceneManager.GetActiveScene().name);
    }
}
```