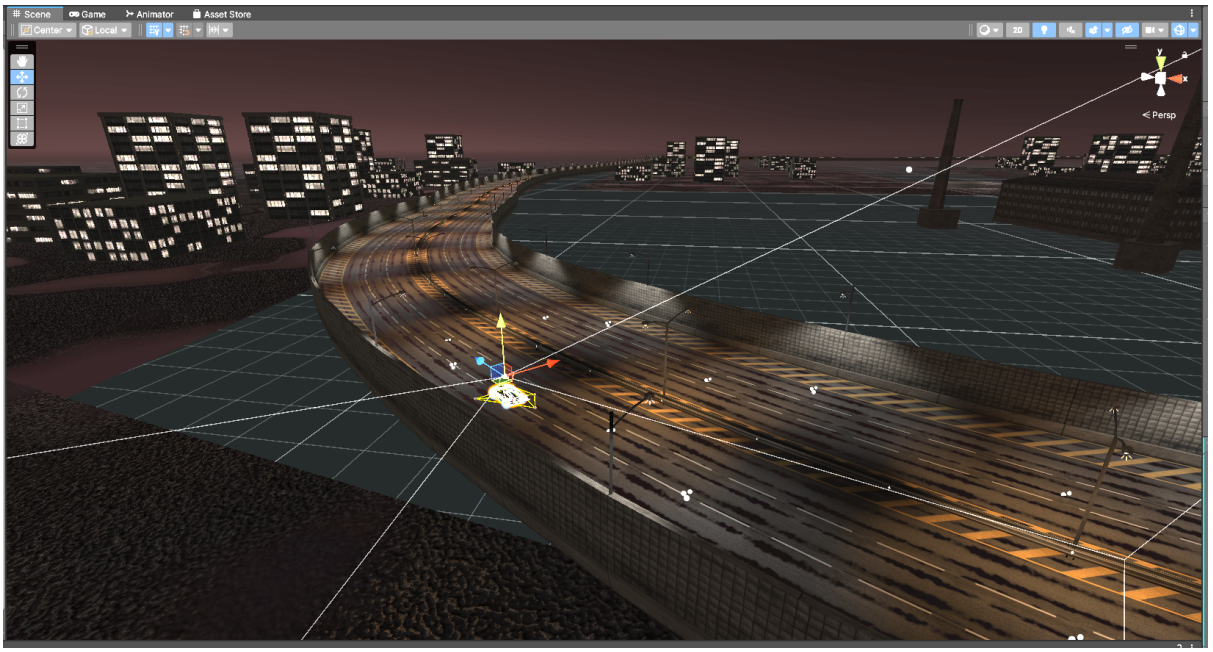
Project Structure and Code Quality Report

Project: *Drift: Immersive*
Name: Georgi Tsvetanski
Date: 11/11, 2025
Engine: Unity 2022.3 (C#)
Platform: PC / VR (OpenXR, iVRy)



1. Project Selection

This report is about my Unity project **Drift: Immersive**, a small 3D racing game where players drive and drift around short tracks.
The project was built to practice vr, physics, camera control, and clean C# scripting.

2. Project Structure Analysis

Current Project Setup

My Unity project has the following folders:

Assets

- **_Project**

    - **Scripts**
    - **Materials**
    - **Prefabs**
    - **Scenes**
    - **UI**
    - **Audio**
- **Packages**
- **ProjectSettings**

What Works Well

- Scripts are grouped by type, which keeps my things organized.
- My movement, drift, and camera logic are separated into different scripts.
- Each script has clear names and short comments that explain what they do.

What Can Be Better

- The **Highway scene** is too large. It makes the game slower and harder to control. I plan to **make it half the size** to improve testing and camera flow.
- In the **City scene**, the player can drive out of bounds. Since the buildings are modular, I can **duplicate and move them** to close open roads and make a drivable loop.
- Some script names don't match a pattern (for example, *CarMovementBehavior* vs *DriftBehavior*). I can rename them for a consistent look.
- I want to add a **Documentation folder** to keep my GDD, diagrams, and notes in one place.

New Folder Plan

Assets

- **_Project**
    - **Scripts**
    - **Scenes**
    - **Art**
    - **UI**
    - **Audio**
    - **Documentation**
- **Packages**

This makes it easier to find files and helps if more people ever work on the game.

3. Code Quality Assessment

Good Code Examples

1. **Clear variables:**
   My scripts use headers and comments to show what each part does.

   ```
   [Header("Camera Settings")]
   public float smoothTime = 0.1f;
   public float zoomOutFactor = 0.001f;
   ```

1. **Good structure:**
   Code is split into small scripts like *CarMovementBehavior*, *DriftBehavior*, and *PlayerController*.

2. **Uses Unity correctly:**
   Input is in Update() and physics is in FixedUpdate(), which should be best practice in Unity.

Needs Improvement

Error checks:

1. Some scripts don't check if a camera or component is missing. I will add a line like **if (rearCamera == null) return;** would stop the game from potentially crashing.

2. **Hardcoded numbers:**
   Example: 0.05f in *PlayerController* is used for steering dead zone. It should be a public variable so it can be changed in Unity.

1. **Repeated math:**
   Both *CarMovementBehavior* and *DriftBehavior* use rotation math. Perhaps this can be moved to a shared helper script to make the code cleaner.

Why Error Handling Matters

Adding simple checks helps prevent crashes when a script is missing a reference, especially as development progresses and more features get added.
It also makes debugging easier and keeps the game running even if small setup mistakes happen.

4. Final Recommendations

- Make the highway smaller and close the city map so players can't drive out.
- Rename scripts and folders for a more consistent style.
- Add a /Documentation folder for notes and diagrams.
- Add more null checks and public variables to make testing easier.
- Create a helper script for shared math functions like rotation and grip.

Expected Results

These changes will make my project:

- Easier to edit and test.
- More stable and optimized.
- Better organized for future updates or VR features.
- Ready to include in a professional portfolio.

References

- Unity Documentation: *C# Best Practices and Scripting Guide*
- Reddit Unity3D Thread – *Car Camera Help* (used for CarCamera.cs idea)
- Unity Asset Store: *CyberpunkSunset* and *Demo City by Versatile Studio*

End of Report
Prepared by Georgi Tsvetanski
Project: Drift: Immersive