

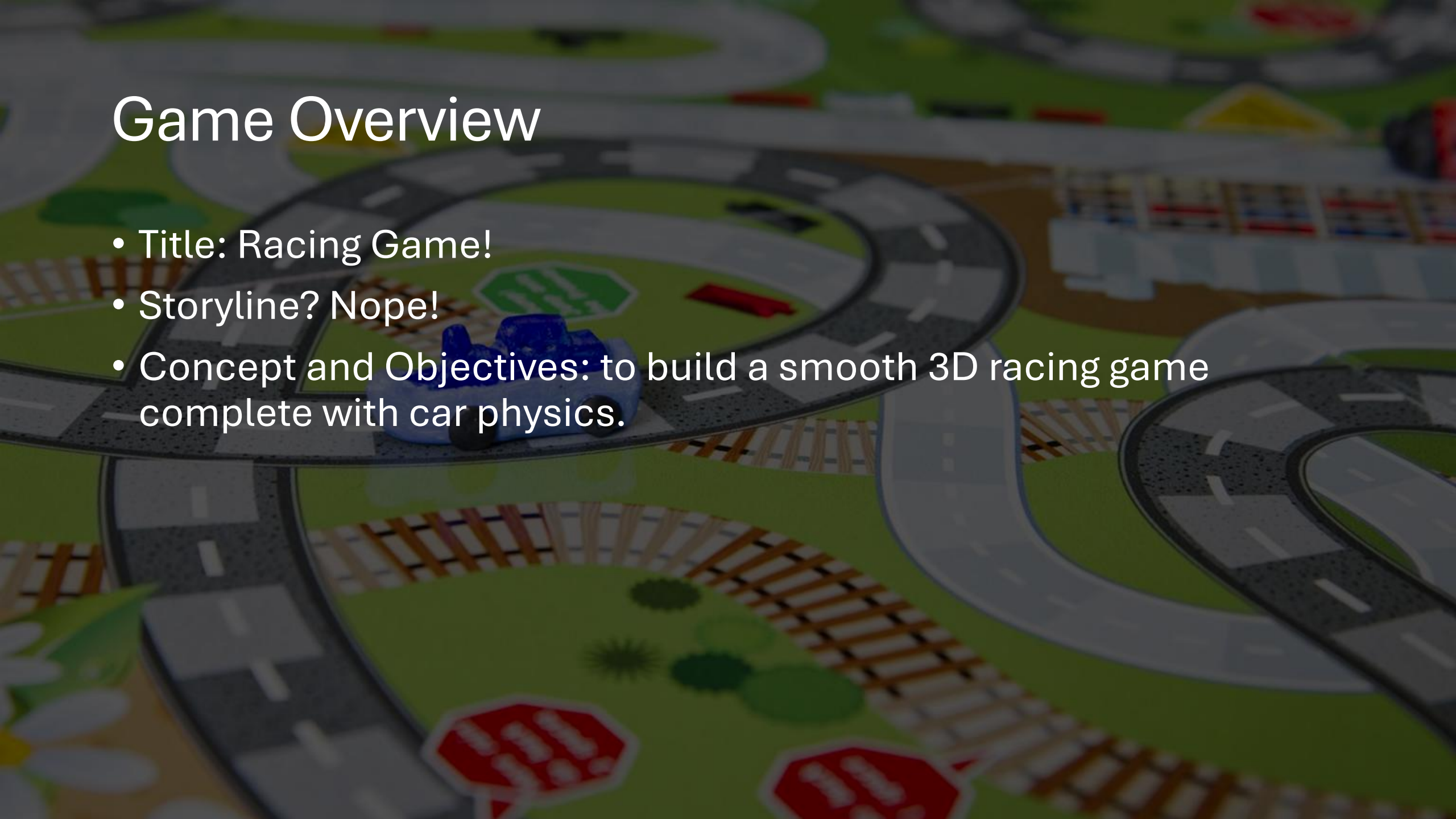


CITA 312 Final Project

By Max Valle

Game Overview

- Title: Racing Game!
- Storyline? Nope!
- Concept and Objectives: to build a smooth 3D racing game complete with car physics.





Development Highlights

Scripts: Movement, CreateTrack, RaceManager, UI Manager, and Checkpoint.

Assets: I built my own racetrack pieces some of the vfx; car asset was imported.

Mechanics: Drive a randomly generated track as fast as you can without flipping, falling off, or missing a checkpoint. Driving is (sort of) more realistic than other car movement we've created.

UI: Three primary "screens": Main Menu, Fail, and Win, with additional UI timers while driving.

Special Features: Random map generation!



Project Progress and Verification

Original Timeline:

- Week 1 – Player Movement, Modular Racetrack Tile Creation (Blender?)
- Week 2 – Procedural level design, *Obstacles*, Checkpoints, Race Timer
- Week 3 – Game Polish (Visuals, Assets, UI), ***AI Implementation***

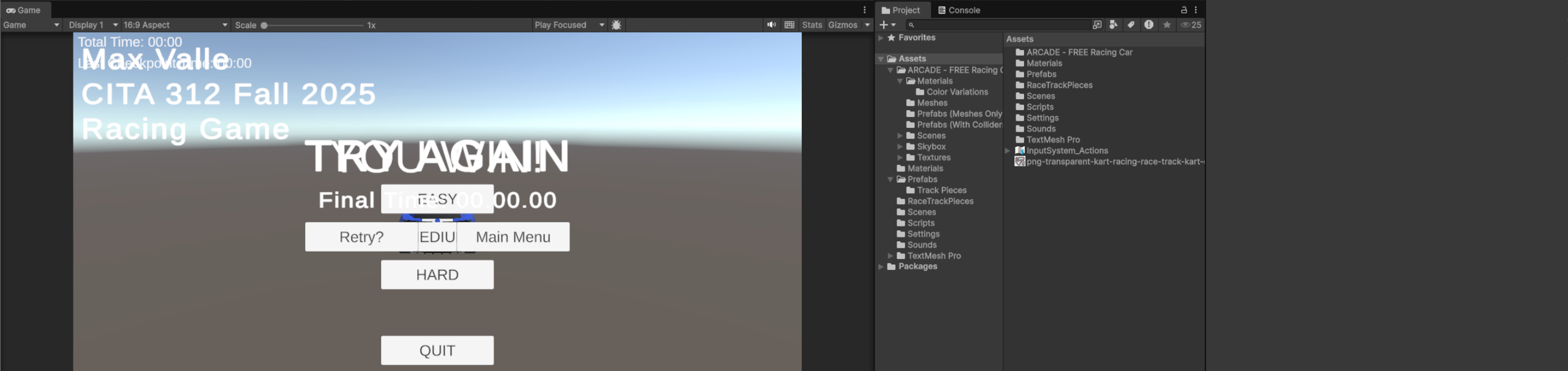
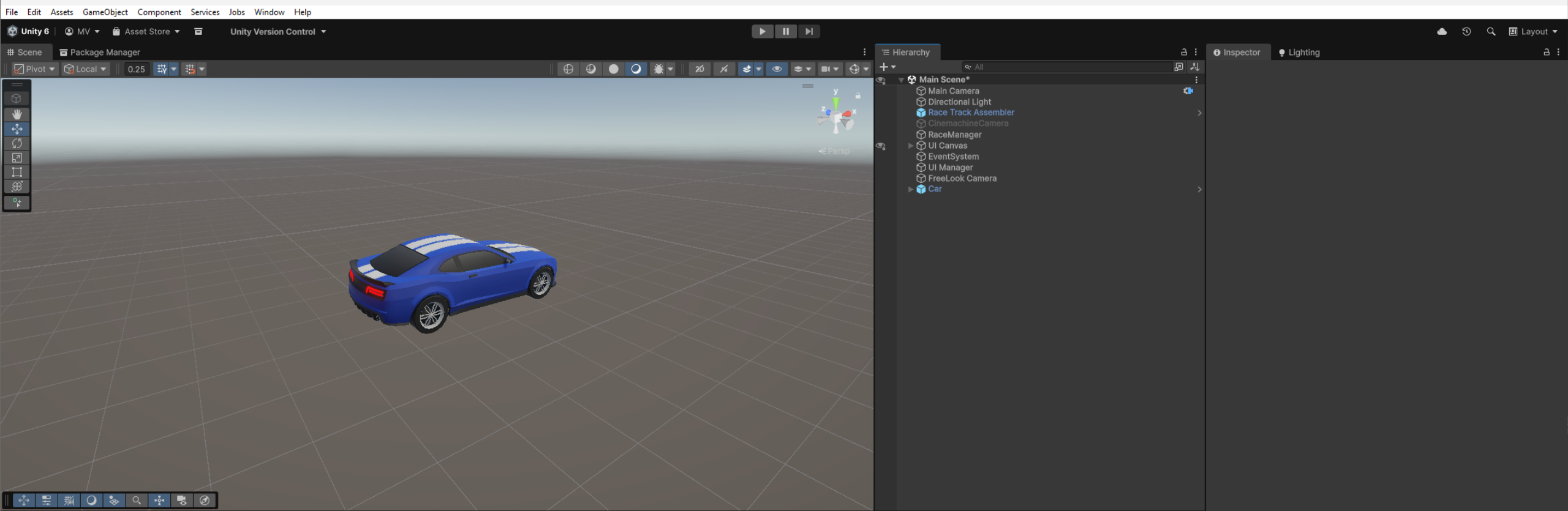
What didn't get finished? *Obstacles** and AI Implementation.



Challenges and Solutions

Do you really want to know? Ok, but don't say I didn't warn you.

- Input actions not firing even though they were mapped correctly.
- Player falling into the void because the track didn't exist yet when the player spawned.
- Player needing to be frozen / made kinematic while in menus to avoid drifting into the abyss.*
- Controls not working after retrying because race state flags weren't being reset consistently.
- Procedurally generated track pieces overlapping in ways that defied the laws of geometry.
- Raycasts checking *from* the wrong direction and mis-detecting valid placements as invalid (or vice versa).
- Track pieces rotating and translating in ways that accumulated floating-point drift over time.
- False positives on overlap detection caused by colliding with the *previous* piece (which needed to be ignored).
- Track generation breaking if you made too many turns in a row, causing "impossible placement" that needed undo logic.*
- The "10 seconds since last checkpoint" fail condition triggering constantly because checkpoint indexing jumped by 10s instead of 1s.
- Total time and checkpoint time getting tangled in each other before being handed off to the UIManager.
- The Booleans `raceStarted`, `raceFinished`, and `failTriggered` ending up in contradictory states because multiple scripts were controlling them.
- Menus activating in strange combinations because panels weren't being disabled consistently.
- Timer UI starting before the race, after the race, or not at all.
- Win and Fail panels appearing correctly — but the underlying game logic not pausing correctly.
- Failing, retrying, and then losing player control because flags weren't reset or the player wasn't un-frozen.
- Procedural piece placement depending on object transforms that didn't update until later in the frame.
- Definitely a few more I've banished from my brain



Inspector

Box Collider

Edit Collider

Is Trigger

Provides Contacts

Material: None (Physics Material)

Center: X 0, Y 1.157988, Z 0

Size: X 2.806107, Y 1.669001, Z 6.716757

Layer Overrides

Movement (Script)

Script: Movement

Car RB: Car (Rigidbody)

Ray Points: 4

- Element 0: FL (Transform)
- Element 1: FR (Transform)
- Element 2: RL (Transform)
- Element 3: RR (Transform)

Acceleration Point: Acceleration Point (Transform)

Driveable: Driveable

Tires: 4

- Element 0: FrontLeftWheel
- Element 1: FrontRightWheel
- Element 2: RearLeftWheel
- Element 3: RearRightWheel

Front Tire Parents: 2

- Element 0: FL
- Element 1: FR

Skid Marks: 2

- Element 0: RL (Trail Renderer)
- Element 1: RR (Trail Renderer)

Skid Smokes: 2

- Element 0: RL (Particle System)
- Element 1: RR (Particle System)

Engine Sound: Car (Audio Source)

Skid Sound: Car (Audio Source)

Spring Stiffness: 30000

Damper Stiffness: 2000

Rest Length: 0.45

Spring Travel: 0.25

Wheel Radius: 0.5

Acceleration: 30

Max Speed: 140

Deceleration: 10

Steer Strength: 30

Turning Curve:

Drag Coefficient: 1.2

Tire Rotation Speed: 3400

Max Steering Angle: 30

Min Side Skid Velocity: 8

Min Pitch: 0.258

Maxpitch: 3.54

Player Input

Actions: InputSystem_Actions (Input Action Asset)

Default Scheme: <Any>

Project-wide actions asset is not recommended to be used with Player Input because it is a singleton reference and all actions maps are enabled by default. You should manually disable all action maps on Start() and manually enable the default action map.

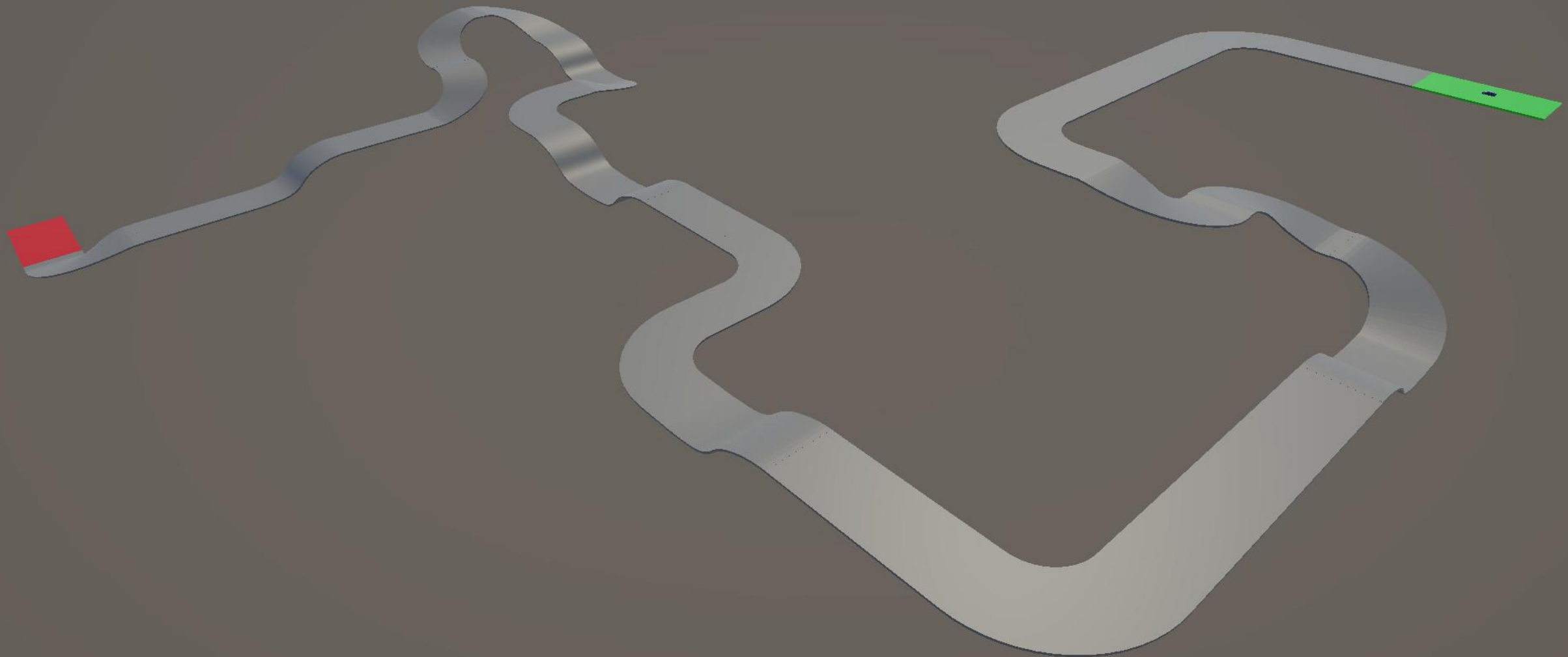
```

Assets > Scripts > Movement > ...
4 public class Movement : MonoBehaviour
137 private void Turn()
140 }
141
142 // Add sideways drag to determine how "slippy" the tires are.
143 private void SidewaysDrag()
144 {
145     float currentSidewaysSpeed = currentCarLocalVelocity.x;
146
147     float dragMagnitude = -currentSidewaysSpeed * dragCoefficient;
148
149     Vector3 dragForce = transform.right * dragMagnitude;
150
151     carRB.AddForceAtPosition(dragForce, carRB.worldCenterOfMass, ForceMode.Acceleration);
152 }
153
154 // Physics based suspension
155 private void Suspension()
156 {
157     // For each raypoint that represents a tire suspension spring
158     for (int i = 0; i < rayPoints.Length; i++)
159     {
160         RaycastHit hit;
161         float maxLength = restLength + springTravel;
162
163         // Calculate ray hit
164         bool rayDidHit = Physics.Raycast(rayPoints[i].position, -rayPoints[i].up, out hit, maxLength);
165
166         // If the ray didn't hit
167         if (!rayDidHit)
168         {
169             // Set that wheel to be ungrounded
170             wheelsIsGrounded[i] = 0;
171             SetTirePosition(tires[i], rayPoints[i].position - rayPoints[i].up * maxLength);
172             Debug.DrawLine(rayPoints[i].position, rayPoints[i].position + (wheelRadius + maxLength) * -rayPoints[i].up, Color.red);
173             continue;
174         }
175
176         // Otherwise, set that wheel to grounded
177         wheelsIsGrounded[i] = 1;
178
179         Vector3 springDir = rayPoints[i].up; // direction of the spring
180
181         // Effective suspension length (pivot to wheel contact)
182         float currentLength = hit.distance - wheelRadius;
183
184         // Extension relative to rest
185         float displacement = restLength - currentLength;
186
187         // Velocity along the spring direction
188         float vel = Vector3.Dot(carRB.GetPointVelocity(rayPoints[i].position), springDir);
189
190         // Hooke's law + damping
191         float force = (springStiffness * displacement) - (damperStiffness * vel);
192
193         // Clamp so suspension doesn't "pull" downward
194         if (force < 0) force = 0;
195
196         // Add the calculated spring force
197         carRB.AddForceAtPosition(springDir * force, rayPoints[i].position);
198
199         // Adjust the visual tire position
200         SetTirePosition(tires[i], hit.point + rayPoints[i].up * wheelRadius);
201
202         Debug.DrawLine(rayPoints[i].position, hit.point, Color.red);
203     }
204
205     // Ground check requires two wheels to be "touching" the ground to be grounded
206     private void GroundCheck()
207

```

Look At All Those Serialized Fields!

They generally fall under 1 of four categories: References to other things, suspension physics, movement physics, and everything else.





Links

My Links:

- GitHub Link: <https://github.com/Valle94/FinalProject>
- Game Link: <https://valle94.github.io/FinalProject/>
- E-Portfolio: <https://sites.google.com/view/mvalle-eportfolio/games/unity-3d-final-project>

Car Physics Tutorials:

- <https://www.youtube.com/playlist?list=PLtYhPiKW6dMUdJPfA1HH2HbPjp45PU9Tf>
- <https://www.youtube.com/watch?v=CdPYlj5uZel>

Assets:

- <https://assetstore.unity.com/packages/3d/vehicles/land/arcade-free-racing-car-161085>