# Route Reroute

## Project Idea

A simple clone of the game Does Not Commute, built using Three.js for 3D rendering. The goal is to create a minimalistic driving game where players control vehicles sequentially in a small town. Each vehicle's path is recorded and replayed in subsequent turns, creating increasingly complex traffic scenarios that the player must navigate.

## Core features

### Gameplay Loop

- Each vehicle’s path is recorded as the player drives. In subsequent turns, all prior routes replay simultaneously, turning the player's past actions into dynamic obstacles.
- As the level progresses, the town becomes increasingly congested with the replayed vehicles. The objective is to reach the destination without collisions.
- Player controls acceleration, braking, and steering for each vehicle.
- Unique vehicle, starting point, and destination for each turn within a level.
- **Planned:** A timer tracks the overall level completion time, potentially increased with each completed turn.
- **Planned:** A health/damage system where collisions impact the vehicle. If health is depleted, the current turn might be rewindable at the cost of a time penalty (specific mechanic TBD). The current implementation features a manual rewind ('R' key) for debugging and testing.
- **Planned:** Gradually introduce more complex routes and vehicles with varying handling characteristics.

### Visual Style

- Third-person 3D camera following the active vehicle.
- Environment features a small town with roads, intersections, and potentially simple buildings.
- Minimalist and cartoonish 3D graphics using Kenney assets.
- Focus on clear visuals, simple shapes, and a clean aesthetic.

### Physics and Collisions

- Custom physics model for vehicle movement (acceleration, braking, steering friction).
- Collision detection between vehicles using Oriented Bounding Boxes (OBB) and the Separating Axis Theorem (SAT).

## Project Status

### Planned Features

- [x] - Car models loading and management.
- [x] - Driving physics (acceleration, braking, steering).
- [x] - OBB-based collision detection (SAT) and basic response.
- [x] - Basic level structure (defining sequences of cars, start/end points).
- [x] - Path recording and replay system for previously driven cars.
- [x] - Rewind mechanic for the currently active car.
- [ ] - Proper scene lighting setup (beyond basic ambient/directional).
- [ ] - Design and implementation of specific game levels/maps.
- [ ] - Map model loading and integration (roads, buildings, environment).
- [ ] - Goal point detection and turn completion logic.
- [ ] - User Interface (Menu, HUD elements like timer, score, health).
- [ ] - Game score keeping mechanics.
- [ ] - Vehicle health/damage system implementation.
- [ ] - Power-Ups (e.g., speed boost, better grip, temporary invulnerability).

### Stretch Goals

- [ ] - Game state saving/loading mechanics.
- [ ] - Dynamic Day/Night Cycle.
- [ ] - Sandbox mode for free driving and experimentation.
- [ ] - Cheat codes for testing or fun (e.g., toggle collisions, unlimited time).