Unity Based Automotive Simulator

Workflow

1. Environment, packages and plugins.

2. Code, framework, algorithm and AI tools.

3. Challenges and scalability

Environment, packages and plugins.

1. **Driving environment construction:**

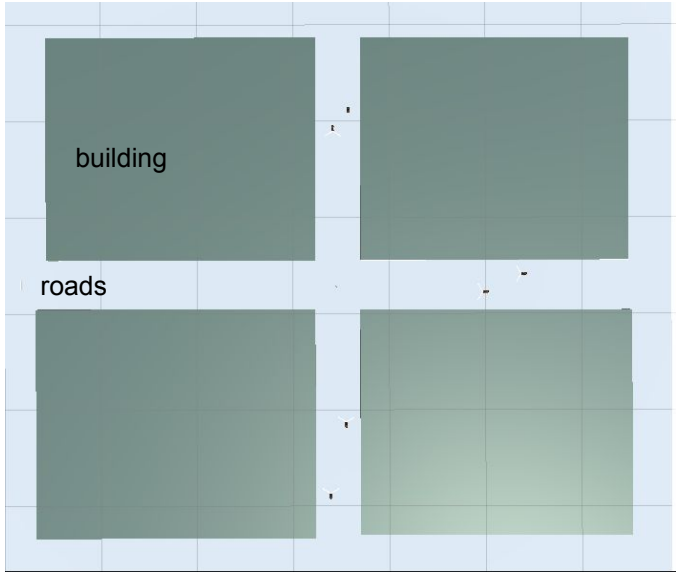
   For testing, I used unity cube 3d object to construct buildings and roads.

2. **Car Model:**

   Import from package RCK 2.3.

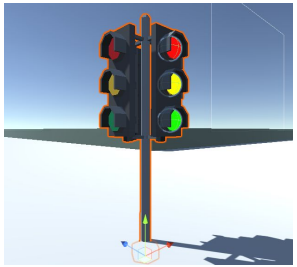
3. **Traffic light model:**

   Import from package Tarbo-CITY-TrafficLights



Pic.1: Roads and building aerial view

Pic.2: Car model

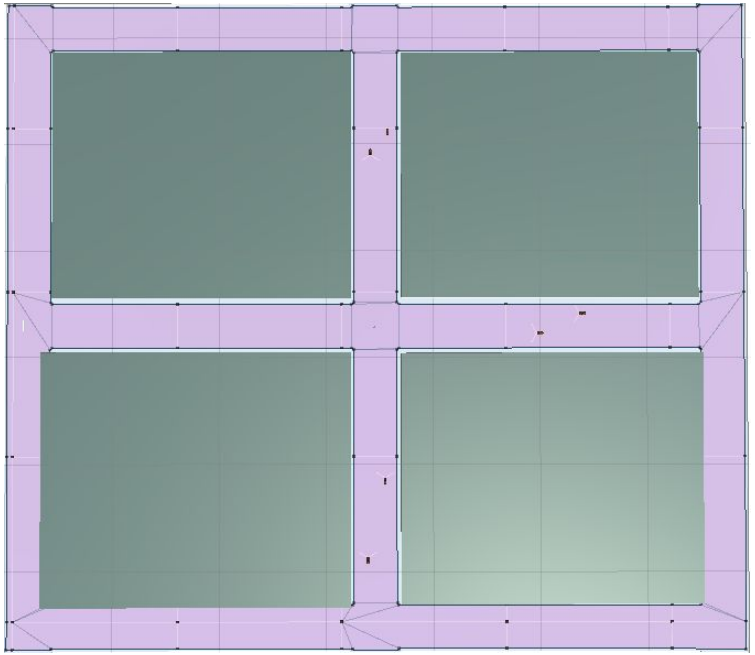


Pic.3: Traffic light

Environment, packages and plugins.

4. Car AI:

Import from CarAI-Unity package and use the NavMesh plugin in Untiy.



Pic.4: NavMesh baked AI roads aerial view

Conclusion for Part One:

For testing purposes, I utilized simple meshes to construct the driving environment, supplemented with models and plugins from readily available packages. These resources proved to be quite effective.

Leveraging these resources allowed me to focus on coding and developing algorithms for the vehicles' behavior, providing a solid foundation for the initial phase of the project.

Code, framework, algorithm and AI tools.

1. **CarController.cs**

   Script for controller the egovehicle, 3 drive mode(frontWheelDrive, rearWheelDrive, allWheelDrive), has addDownForce() function and getFriction() function , and used ackermann formula to simulate steer, all of these make real-like drive experience.

2. **CameraFollowFirstPerson/CameraFollowThirdPerson.cs**

   Egovehicle(Named RallyCar) prefab has 2 cameras for different view.

3. **CameraSwitcher.cs**

   Press V to change camera view.

Code, framework, algorithm and AI tools.

4. InputManager.cs

Unity input manager, for vertical, horizontal and handbrake input

5. SpeedDisplay.cs

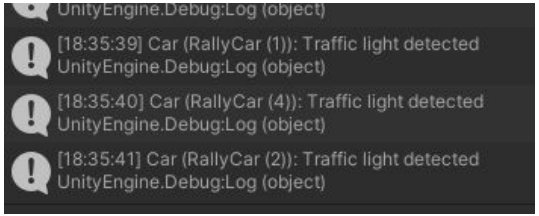
AR display, could show ego vehicle's speed, warning information of too close to AI vehicle, recommendation engine.

6. AI vehicle:

For AI vehicle driving, I used CarAI packages and NavMesh plugin. CarAI has a CarAI.cs script for vehicle's movement and steerRotate, I add a steerRotate animation code block in it.

I also add trafficlight detection function and AI vehicle detection function.

Pic.5: AR display



Pic.6: AI vehicle debug.log

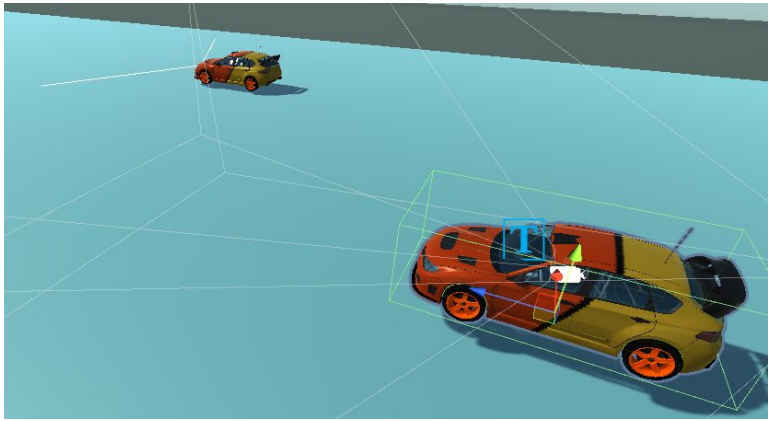
Video.1: Vehicle Controll showcase

Code, framework, algorithm and AI tools.

Conclusion for Part Two:

In this section, I have successfully implemented the movement and rotation behavior for the vehicles. The ego vehicle is equipped with an AR display providing essential information. AI vehicles are capable of autonomous driving, either towards a fixed destination or without any predefined route. They also have the capability to log debugging information.

The constructed framework consists of one ego vehicle and five AI vehicles.



Pic.7: AI vehicle(front) has a FOV Gizmos, egovehicle(back) has a AR display.

Challenges and scalability

1. The ego-vehicle can be enhanced to deliver a more immersive driving experience. This could include voice integration, improved driving code, and a more comprehensive AR display.

2. AI vehicles could benefit from a more robust AI code implementation. In this regard, they could be integrated with the ML-Agents plugin for training neural networks, further enhancing their intelligence and responsiveness.

3. This prototype was developed in a testing environment. Future iterations could incorporate a realistic city map to provide a more authentic and engaging user experience.