SUMANT PATEL (6796841) and ROHAL KABIR (7105836)

COSC 3P91 – Assignment 1

1 INTRODUCTION

The design and architectural choices for the construction of a traffic simulation game are described in this document. The traffic network in the game is modeled after an urban one, enabling both player-controlled and automated cars to maneuver through constantly shifting traffic patterns. The purpose of this design paper is to defend the architectural selection and describe how it enables the necessary game behaviors.

In the interactive traffic simulation game, users take control of cars as they navigate a city's road system. The goal of the game is to skillfully maneuver through traffic while avoiding crashes and preserving the health of the car. The simulation runs on a time-stepped basis, requiring choices to be made at different points in time, such as; lane changes and crossroads.

2 PACKAGE DESCRIPTIONS

2.1 VehiclePack

Classes for vehicles in the game, including Car, Truck, Bike, and Bus, are included in this bundle. Each class has unique characteristics, such as health and speed. A more realistic simulation of vehicle durability and player reputation is made possible with the addition of the Health and VehicleReputation classes, which has an effect on game play dynamics.

2.2 PlayerPack

The PlayerPack bundle emphasizes decision-making and player engagement in the game. It comes with MyPlayer, which is the user's avatar, and Bots, which are randomly driven cars. In later versions, adding multiple player functionality may be done in a scalable manner thanks to this arrangement.

2.3 GamePack

The DecisionMaker interface, which controls how cars make decisions at crossings, when changing lanes, and at other crucial times, is a key component of the GameEngine package and is essential to the logic of the game. This abstraction makes it possible for game logic to be adaptable and change over time.

2.4 GameMapPack

To improve the realism of the simulation, the GameEngine package is the main package of this game which basically serves as the core logic and control center, orchestrating the interactions between various components of the game, such as vehicles, the map, and player decisions.

Authors' address: Sumant Patel (6796841); Rohal Kabir (7105836).

Update date: 4 February 2024

3 DECISION MAKER INTERFACE

Is the interface; which is a pivotal part of the Game Engine package, abstracting the decision-making process within the game. It outlines methods for making critical decisions at intersections, lane changes, challenges, and gambles. By implementing this interface, the game can dynamically respond to the current state of the simulation, making real-time decisions that affect the outcome of the game.

3.0.1 Game Logic and Rules. The Game Engine manages the game's rules and logic, ensuring that the simulation adheres to the defined parameters, such as vehicle movement, traffic light changes, and collision detection. It evaluates player decisions against the game's rules, determining the consequences of those actions.

3.0.2 Time Stepped Simulation Control. It controls the time-stepped nature of the simulation, updating the state of the game at regular intervals. This includes moving vehicles according to their speed and the player's inputs, updating traffic lights, and triggering events like challenges or accidents.

3.0.3 Collision Detection and Response. This part of the Game Engine's responsibility is to detect collisions between vehicles or with other objects in the environment. Upon detecting a collision, it calculates the outcomes based on the game's physics, such as damage to vehicles and impact on player reputation.

3.0.4 Reputation and Score Management; The Game Engine tracks and updates the reputation and scores of players based on their actions within the game. This includes rewarding conservative decisions, penalizing risky maneuvers that result in collisions, and determining the outcome of gambles and challenges.

3.0.5 Event Handling and the State of the Game. It handles events within the game, such as player inputs for vehicle movement, interactions at intersections, and lane changes. The Game Engine maintains the overall state of the game, including the positions of all vehicles, the status of traffic lights, and the progression of the game over time.

4 CLASSES AND RELATIONSHIP DESCRIPTIONS

4.1 Vehicles

In this Abstract class we have attributes such as speed, health and reputation. The child classes (Car, Truck, Bike, and Bus) will inherit these properties. The child classes also have some properties of their own such as their own fixed top speed and max health they can achieve.

4.2 Health

The health class is designed to manage the durability and damage status of the vehicles. It has attributes such as; current health as an integer value representing the current status of the vehicle. It will eventually change when the vehicle gets damaged and eventually restore to the maximum limit if the player drives well and gains points. Moreover, the max Health is another attribute as an integer as well which indicates the maximum health capacity of the vehicle, this would be the highest value the current Health could reach signifying the vehicle is in good condition. There is a composite relationship between the health class and the vehicle class as the vehicle class owns the object health class, which implies that the health object is a part of the vehicle and cannot exist independently. This relationship allows for better encapsulating vehicle-related objects and operations. The health class can manage all aspects related to the vehicle condition (damages, health) which will be more focused on the primary aspect.

4.3 VehicleReputation

This class simulates the consequences of player decisions and interactions within the game environment. It describes the behavior of the vehicles in the traffic network. It has a score attribute that is based on the vehicle's reputation, which is influenced by the player's actions such as obeying the regulations, avoiding collisions, or maybe winning challenges. It has a composite relationship which is similar to the health class, where the relationship indicates that each vehicle instance owns a vehicle reputation instance as a core part of its state which is tied to the vehicle.

4.4 MyPlayer

The myPlayer class represents the human's avatar in the game. It is the primary interface between the player and the actual game, allowing the player to make decisions interact within the game environment, and monitor the progress in the game. Attributes such as; name is a string that holds the chosen name within the game whereas the score tracks the player's score which is influenced by several objectives or in-game achievements. On the other hand, the bots class represents the computer-controlled players that populate the gaming environment with the simulation of other vehicles and drivers. These entities can also make decisions based on the behaviors that are predefined and contribute to the game mechanism. They also have an attribute bot id to keep track of their records in the system and interactions that the player will make against them.

4.5 GameEngine

The game engine class serves as the central processing unit in the game, summarizing the overall game logic including initiation, progression, and the termination of the game, and some key decisions.

The attributes like Player refer to the instance of the myPlayer class where the game engine can track player status, decisions, and interactions made by them. Lanes represent the game's road lanes which are crucial for managing the traffic network. It is the collection that holds each state inside the game, like vehicles present in them, traffic conditions, and any other possible events within them. Furthermore, it implements the decision-maker interface where the engine is responsible for the in game key decisions.

There is a start method that initiates the game like loading the game map, inserting vehicles on the map for both bots and the player, traffic conditions, and setting some objectives and missions for the player. It triggers the game time, which will advance the game play events to start at the beginning. Where as the stop method is responsible for terminating the game which can lead to the player achieving his objectives, the vehicle getting destroyed, or the player choosing to exit the game reluctantly. This method will make sure that the state is concluded properly with the necessary progress required in the game.

There is a relationship between the game engine and the myPlayer as aggregation which implies that the game engine has a Player, which gives the ability to track and interact with players managing their decisions and the impacts caused. Secondly, there is a similar relationship of aggregation between game engines and vehicles as well where the game engine 'has' or utilizes the vehicles, however, vehicles can exist independently of the game engine implying that they are strictly bound to the game engine.

4.6 Map

We can say that the map class is the game's blueprint with the roads and intersections (using a graph). Attributes like a list of roads in the map are a collection that represents all the road segments within the gaming environment, each segment of the road can be a piece of a map where vehicles can travel. The purpose of this is to define the paths that the vehicles can take, including player and bot-controlled vehicles. On the other hand, intersections represent the points where two or more roads meet or cross. They serve as a critical decision for the vehicle's navigation where characters respond to the traffic signals and interact with other vehicles. The design detects the behavior of the intersections significantly and impacts the flow of traffic and the complexities of the game's navigation challenges.

4.7 Road, Lanes

In the lanes class, we have methods such as checkVehicle to check if a vehicle is positioned in a lane as well as add a vehicle to a lane. If a Road does not exist then a lane cannot and because of this we have a composition relationship between lanes and roads. Road class holds an array list of lanes which we can insert vehicles into.

4.8 Intersection

The intersection class will be able to detect if a vehicle that is controlled by myplayer has entered an intersection, with the vehicleArrived() method. An intersection has a traffic light, this is why we are using composition to describe the relationship between the intersection and traffic light.

4.9 TrafficLight

Traffic Light class and Light Color enumeration allow for accurate simulation of traffic light behavior, impacting vehicle movement across the map.