AV Drive

By Amira Johnson

Project Description

AV Drive is a self-driving car simulator that uses Breadth-First and Depth-First Search algorithms to navigate a busy parking lot! Users can place pedestrians, bicycles, and other cars into the parking lot, and when the space key is pressed, the AV navigates the path.

Competitive Analysis

Most self-driving car simulation games, such as CARLA, are open-source python APIs that require previous programming knowledge to execute the program (i.e. programming pedestrians to walk along the roads). Additionally, they showcase 3D graphics and visualizations in place of 2D maps, and their user ‘sees’ via first-person point of view. AV Drive is different because it showcases a 2D, birds-eye view of the parking lot that the car drives on, although incorporating 3D elements is part of the plan for a post-MVP prototype.

Structural plan

The code will be organized all within one file, separated by classes, and based on MVC – so there will be a model, section, and view section of the code. Each section will be separated by a series of comments that hold the title of the section. Also, each major function will have a short comment explaining

Algorithmic Plan

The AV will navigate the course using the breadth-first search algorithm, and the bicyclists, pedestrians, and other cars will drive via depth-first-search or backtracking. The walls and parked car obstacles are randomly generated each time the course is reset. The trickiest part of the project is implementing the breadth-first search and depth first search algorithms along with getting the car to prioritize moving forward before moving backwards. The depth-first search algorithm, based off what we learned in class, will work by adding a step to the solution, seeing if it works, and then proceeding from there. The breadth-first search algorithm will work by keeping track of nodes on the queue, and nodes that have been visited already. Once the next node on the queue is the AV’s goal node, it will return all the nodes that were visited to get there.l≥÷

Timeline Plan

Most of the major features of the application – placing obstacles, randomly generated lot, Breadth/Depth-first search have already been implemented by TP1. Additional features such as improved user interface will be implemented by TP2, and possible 3D graphics by TP3.

Version Control Plan

Graphical user interface, application

Description automatically generated

I plan to use Google Drive for version control, each TP submission, in addition to safety files will be kept in a folder. Since Google Drive records the date of upload, I will be able to keep track of the most recent version of AV Drive.

Module List

As a potential post MVP feature, I hope to use the Argoverse dataset as a part of the main AV cars pathfinding algorithm. This would mostly be used in the case of a pedestrian, bike, or car moving in the direction of the car, potentially crossing paths with the main AV. With Argoverse, the car would be equipped with predictions information that would let the car know what the most common response to obstacles would be.

TP2 Updates (4/22/22)

* Improved bfs/dfs algorithms
* Can click buttons
* Instructions Screen
* Can place a limited number of bikes and pedestrians, both go to their designated goals
* Trying to get Pedestrians to be sprites instead – a little walking person

TP3 Updates (4/27/22)

* Improved user interface (cute theme)
* Can place an unlimited number of obstacles on the board.
* Finalized bfs/dfs algorithms
* Highlight bike goals