Martin Sardin
BSEE/MSCS

Multi-vehicle_pathfinding_MSardin_github.odt

GitHub Description:
The program simulates multiple vehicles picking up passengers and dropping them off at their final destination. Vehicle movement is computed and plotted in real-time on a graphical map display. Due to the pathfinding innovation, only non-pathfinding algorithm code will be published at this time.

The Simulation (Python & Tkinter)

The program simulates multiple vehicles picking up passengers and dropping them off at their final destination. A request is made for a pickup, then an available car is sent for the pickup. A pathfinding algorithm computes the best route. Vehicle movement is computed and plotted in real-time on a graphical map display. Due to a pathfinding innovation, only non-pathfinding algorithm code will be published at this time.

Due to time constraints, I **opted for a rapid development, proof-of-concept,** in all Python.

The bulk of the Simulation Program was conceived and accomplished in my spare time over the span of about 7-days, starting from nothing, zero. Learned and used the graphic engine Tkinter for the first-time during that 7-days.

The **pathfinding algorithm** concept, coding, testing, and bug fixing took less than 2 hours on the last day of that 7 day period. *It required a paradigm shift to conceive and the ability to look outside-the-box. It's elegantly simple.*

Features:

Pathfinding / routing algorithm

My own design. The innovation, I'm calling it **NewX** for now. I don't mean I did an implementation of a popular algorithm, I mean new. Inspired by: A-star and my willingness to look *outside-the-box*. It's about 30 lines of code

Avg computation time for one route on one core of my Intel Core i7 @ 3.50 Ghz: *100 microseconds* in Python (a scripting language)

From past experience, I expect that porting the algorithm to C++ will result in a computation drop to around *35 microseconds.*

Further work:

Need to do a comparative study between **NewX** and a popular routing algorithm, perhaps A-star, using a large set of computer generated randomly chosen pickup & dropoff test cases.

Physics

Vehicle movement and distances traveled are real-world correct.
Vehicle top speed: 40 miles/hr
Vehicle starts slowing down within 1-mile of it's final destination

Graphics

Sim time, 1 (1-sec) = 1 minute vehicle movement delta-time
The Red circle depicts area of potential safety concern.
The Blue circle depicts the service area within as 10-minute window.
The GE receives GPS coordinates of vehicles, then translates that to the correct location on the map.
Vehicle colors: black = available blue = In-use / in-transit, red = arriving at final destination

Vehicle location re-balancing

In the sim vid at the end, the re-balance was done manually, there is no algorithm behind that. Algorithm TBD