Traffic Simulation Documentation

Traffic Simulation

A simple system to simulate traffic at an intersection in the context of computer security.

Authors/Contributions

Huseyn Gambarov (hxg410@case.edu):

- Found pygame library
- Implemented majority of code for moving cars and pedestrians and part of markdown file

William Kanieski (wck26@case.edu):

- Designed background and part of sprites
- Wrote majority of markdown file and helped debug code

Both:

- Put together the logic design for the application (traffic light sequence, pedestrian/car syncing, et cetera)

Specification

The purpose of this program is to visualize road and pedestrian traffic at an intersection in the context of secure versus insecure states. It allows the user to see how traffic can be seen as a sequence of secure states that, when followed in practice, lead to a safe road where cars and pedestrians do not collide and are safely able to reach their destinations. This simulation includes left and right turns as well as straight-moving traffic and displays the traffic lights shown to the pedestrians and drivers.

Design

Our program depicts a cartoon of a four-way intersection complete with multi-colored cars traversing the streets and miniature pedestrian sprites clustered along the sidewalks. Each of the four intersecting lanes of traffic has a corresponding red/yellow/green stoplight on the upper right part of its sidewalk which controls the traffic for that lane. The stoplights transition from green to yellow to red and back, with at least one light being red at any given time. Cars in the center lane can make left turns, and cars in the regular lane can make right turns. Pedestrians always move parallel to whatever lane of cars currently has a green light.

Implementation

Our program is written in Python 3 and uses the Python GUI from the library "pygame" version 2.5.2. The background is a four-way intersection drawn using Microsoft Paint to have certain dimensions to fit the car lanes and pedestrian sidewalks. A constant number of car sprites are used, and when they reach the opposite ends of the canvas they go off screen and respawn back where they started. The code utilizes Python's random library to generate new colors for incoming cars in all lanes.

Pedestrian and car traffic are both synced in such a way that they do not collide with one another, and cars will wait for pedestrians and other cars while turning. This is done by measuring the distance between the vehicles and pedestrians in a certain direction and ensuring that one stops and waits while the other one proceeds forward and moves out of range.

To successfully run the game you need to:

1. Create the virtual environment:
    - *python -m venv venv*
2. Activate the environment
    - *./venv/Scripts/activate* - Windows
    - *source ./venv/bin/activate* - Linux/Mac OS
3. Install dependencies
    - *pip install -r requirements.txt*
4. Run the game:
    - *python game.py* - Windows

- *python3 game.py* - Linux/Mac OS

Results/Conclusion

We can see from our program that, even when the rules of the road are followed, our traffic intersection still requires a lot of human input to remain safe and secure. For instance, when a pedestrian is crossing the street parallel to vehicle traffic, it is still possible for a car to make a right turn and intersect the path of the crosswalk. In order to prevent a collision between a car and a pedestrian, either the car or the pedestrian must wait for the other one to pass before they themselves can move on their way. In real life, this is mitigated with the concept of right of way, which based on the mindsets of the pedestrians and drivers involved at a given time, and the order in which drivers and pedestrians cross may not always be deterministic.

An important concept to learn from this simulation is that many security systems rely on safe, rational human behavior in order to remain secure. If people do not follow traffic laws, there are bound to accidents; similarly, if programmers do not follow security protocols, there may be insecure states. An important goal is to predict these states before they happen, and hopefully prevent them in practice.