City Traffic Simulator

Software Design Document

Team Name
Chavez, Bryan
Majid, Mohammad
Nguyen, Brian
Vuong Nguyen, Huy

Introduction

Purpose

This purpose of this document is to provide detailed description of the design of a system capable enough to allow for software development to continue with an understanding of what is to be built and how it is expected to built.

Design Goals

The system will make use of Object Oriented Design Paradigm. The system should be able to stop cars at traffic lights for two minutes. Cars will also stop at stop signs for a second. After the cars start accelerating it should only accelerate up to 15 mile per hour for the first 1/8th of a mile then after that it should continue up to 30 mph and maintain the speed from there.

Design Trade-Offs

The system platform will be relatively generic which helps readability and reusability.

1. Datas are imported and exported in CSV instead of database:
    - Easy to use, easy to read, user friendly
    - Hard to maintain when we have arbitrarily large amount of csv files. Cost a lot of storages.
2. Software is developed using OOP paradigm:
    - Reusability, we basically create our own API.
    - Easy to implement, deploy, test and debug.

Definitions, Acronyms, and Abbreviations

OOP: Object-oriented programming

GUI: Graphical user interface.

UML : Unified Modelling Language

Python : General programming language

Django : Backend python platform for web development

HTML : Hypertext Markup Language

XML : Extensible Markup Language

References

No references at this time.

Overview

The overview of this project is mainly organizing enough traffic lights and stop signs to be processed correctly and set in the city of Pacopolis. The city of Pacopolis' goal is achieving the necessary help in setting the traffic lights and stop signs to help minimize the amount of time needed for each vehicle to get to their specific destination. This system will eventually minimize accidents and damages done in the city too eventually becoming a functional city that will help portray a correct way of driving getting to the needed destination in less time.

Current Software Architecture

Proposed Software Architecture

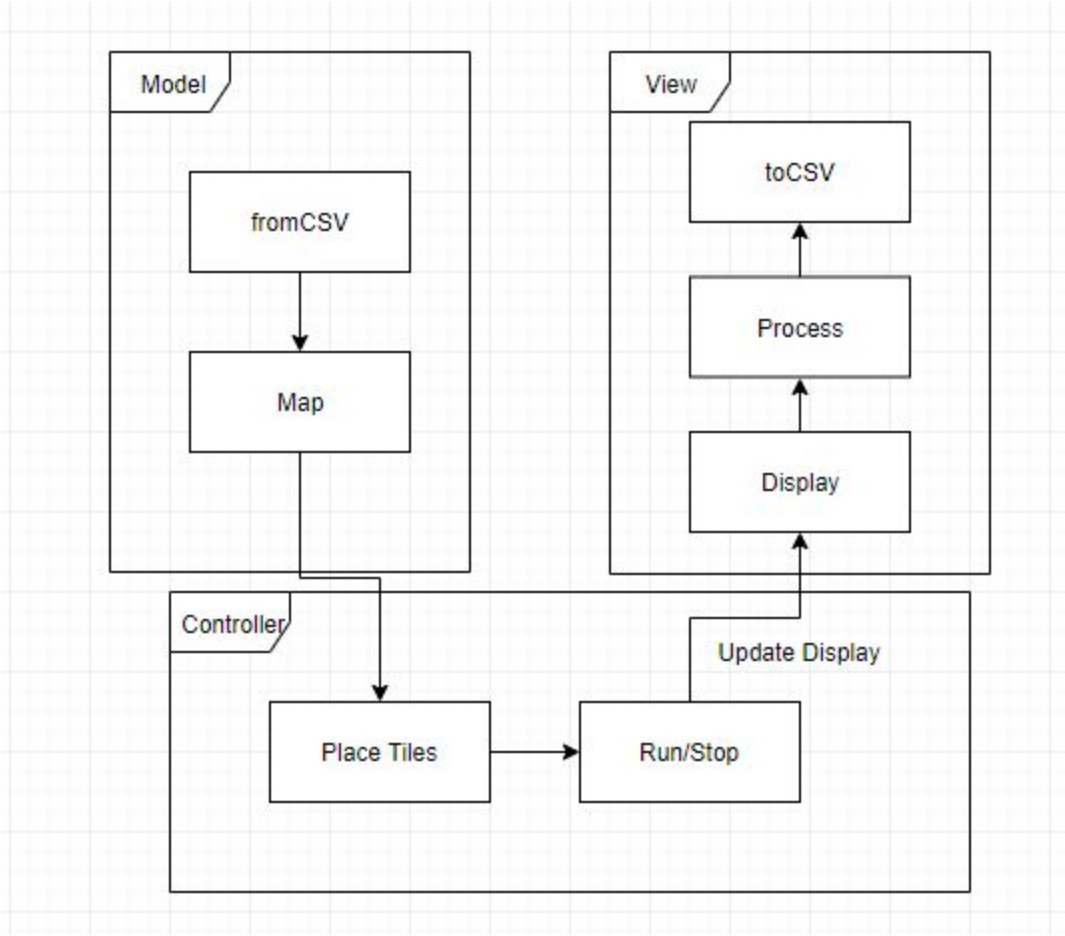
Overview

We will be using the Model View Controller architecture for our system. We will be explaining the software design of our project. These software designs will include Subsystem Decomposition where it explains all the subsystems that are part of the simulation and their jobs. It also explains where the data is being stored with the explanation of Access Control and Security. Global Software Control will be explained as well which will explain how the system is being created and how it works. It will also show how the subsystems interact to achieve the goals. Subsystem services will give you the summary of the subsystems and their jobs.

Subsystem decomposition

We have multiple subsystems doing many different jobs. The simulation class will be the main frame. CommandPallete class is where it includes methods such as setting the simulations map, start simulation, stop simulation, and quit simulation. The map class will include functions that will add, get, and set the cars. This system will handle accidents with accident_handlr and other set the approratte path using the find_path method. Another function is also included in the map called place titles which will help set the structure of the map. Finding a path is also a function that will be used to find the appropriate path for the vehicles. And lastly for the map class a accident handler function is provided to handle accidents. The car class includes functions such as moving, velocity, and setting the state of the car. Tiles is another class that included setting the coordinates needed for the tiles. The tiles class includes multiple subclasses such as road, building, stop sign, intersection, and traffic

lights. These are all subclasses of tiles and will be presented in the simulation when ran. Finally, there is a coordinate class that calculates/sets coordinates and manhattan distance.



Hardware/software mapping

No hardware implementation done in this simulator.

Persistent data management

Different model of cities, maps, cars, tiles,.. will be stored inside CSV file(s).

Access control and security

For our control and security we decided to have a login where the user can login before starting the simulation. This will help protect the simulation that the mayor will be using and testing on.

Global software control

The system will be ran through gui. We will use python to code and start the mapping and controls of system through tkinter that communicates with the interface. Requests are given by the user and the user has the ability

to start, stop, or reset the system. When the user starts the simulation the CommandPallete class will operate the method start_button() to start the simulation. When stopping the simulation, the CommandPallete class will use the pause_button() method where it will pause the game exactly where it is. When ever the user wants to reset the simulation per request iIt will use the quit() method that will quit the simulation and get back to its start phase. The CommandPallete interacts with the user interface and is associated with the map class where the simulation occurs. Map is than aggregated with the simulation, so is the Tiles class. The Tiles class include multiple child classes that will interact with the Tiles class when used. The child classes are Car, Road, Building, Stop Sign, Intersection, Traffic Lights. These subsystems will help provide the needed objects when communicating with Tiles because the Tiles class will use these subsystems, that it interacts with to portray the objects appropriately.

Boundary conditions

-The start-up of our design includes the start of the simulation itself. Vehicles will start entering the map while its count is being calculated. The shutdown simply stops the simulation exactly where its current state. Simple error behaviors that occur in the system are accounted for. For example, when accidents occur the system will take care of this by simply resetting the system.

Subsystem Services

Subsystems such as map, tiles, and coordinate of the simulation will help set up the maps, its coordinates and the tiles for a nice frame and architectural look. The other subsystems such as stop sign, traffic lights, roads, buildings, and intersections also will provide the necessary objects needed for the other subsystems in order for the simulation to properly function. First of all, the CommanPallete subsystem will properly calculate the time when the simulation starts with the start_timing method. The Coordinates subsystem will help set the right coordinates and calculate the manhattan distance of the vehicles. The start_button method will help by starting the simulation. pause_button will stop the simulation exactly where it is. The quit() method will then shut the system and log out of it. The Map subsystem will use methods such as add_car, get_car, and set_car. These methods will help add get and set the car appropriately. The find_path method will help find the most successful path set by the system. The accident_handlr method will handle any scenario where accidents might occur. The Tiles subsystem will help set and setup the atcheture look for the map. Other subsystems are connected to it such as Road, Car, Stop sign, Intersection Traffic Lights, Building. These objects will play a big role when creating the map. These subsystems are all connected to the Tile class as subclasses and the Tile subsystem is associated with the Map subsystem which will set the appropriate objects and set the architecture of the system. Map and Tile are aggregated with the simulation subsystem.

Class Interfaces

FOR OPTION 1.

Detailed Design

FOR OPTION 1.

Glossary

OOP : Object-oriented programming

GUI : Graphical user interface.

UML : Unified Modelling Language

Python : General programming language

Django : Backend python platform for web development

HTML : Hypertext Markup Language

XML : Extensible Markup Language