A Traffic Simulation Model Allowing For Wide-ranged Vehicle Communication

Timmy Galvin- TJHSST Computer Systems Lab

Abstract:

Traffic is an ever-growing problem as population around the world increases exponentially and with it, the number of drivers. Previously, fluid flow models have been used in an attempt to model traffic, but as has been recently discovered, only agent based models can accurately model a traffic scenario as small perturbations can have a butterfly effect and change the entire system.

Introduction:

I want to create an algorithm that responds to traffic backups by sending information across the system and altering traffic laws to help such situations. I will develop a method of calculation bad traffic jams and to what extent the speed limit on the roads leading up the jam need to be changed. For this to function well, I need to have an accurate model of driver behavior and an ability for my program to collect data and analyze the situation well. The model will be of varying road systems and the vehicles on the road will have their own properties such as location, speed, acceleration, speed limit, and aggressiveness, some of which will be user defined. While the simulation is made in an attempt to copy human behavior, like all other traffic models in use, it will be collision-free.

Developments Sections:

The World class that my environment is stored in holds an ArrayList of all the Vehicles so that it can access their information and location. Using these values it will be able to detect traffic jams and react to them by altering conditions. I have set up a basic simulation to do this, at the moment it is a two-lane system with traffic flowing in two directions moving at realistic velocity and acceleration and responding to a speed limit and vehicles around them. There is a variable in the program for traffic density that can easily be altered to change the number of cars. This program will be successful if it one, accurately depicts traffic flow by means of a traffic density versus flow graph, and two, if it can use information from the entire system to alter traffic laws.

Reaction Algorithm

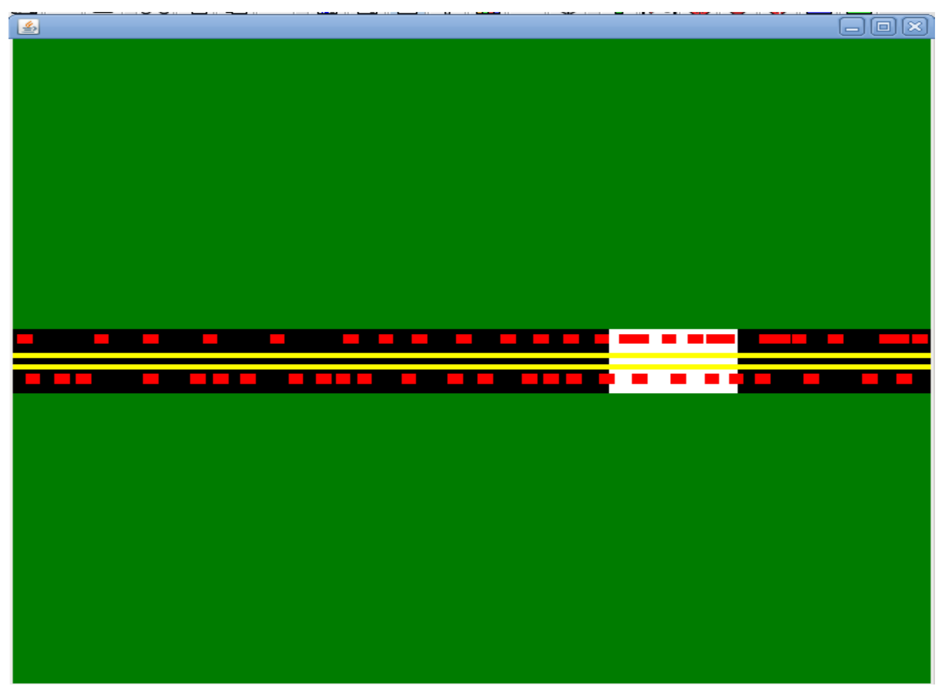
This is a main component of my program as it determines the braking speed of the vehicles in relation to the vehicle in front of them. At first I used a linear model but found it be rather inaccurate, or rather, that with my micro model, I could use more complicated functions because of small set sizes and the ability to visually observe the results. Currently it is a combination of two parabolic functions that vary the speed of the car behind as a function of its current speed and the speed of the car in front of it. This will prevent the car behind from ever running up on the car ahead of it or overrunning it.

Results and Discussion

At the moment my program works in a two-lane traffic simulation. The only test of its validity that I have used is a visual comparison to known behavior. The behavior that it depicts is a traffic jam moving backwards in traffic. With the speed trap I built in, I was able to discover that it is better to have a speed trap earlier on rather than later, as traffic will build up on the road with the speed trap later on, making a slower lane. Hopefully, when my project is fully implemented I will yield results on how information sharing and system reaction will affect traffic flow, though I am not sure if it will be marginable enough to be reasoning behind buying many sensing systems for actual road systems.

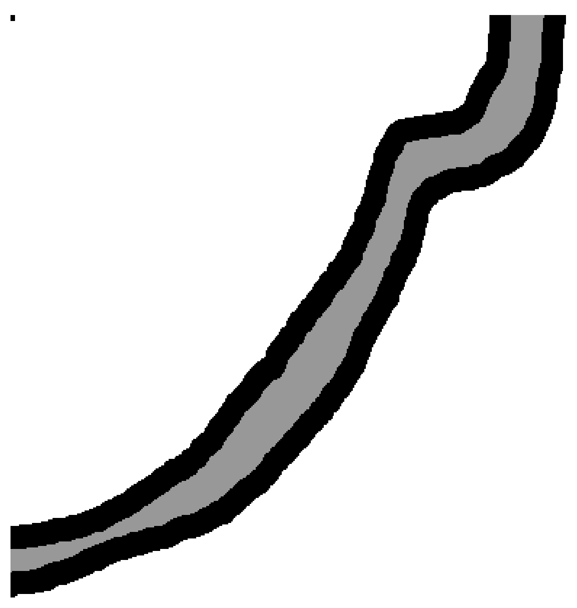
Realistically allowing the user to define environments initially provides no results to my project because it still lacks the eventual elements that will separate it from other traffic simulations such as a 2d lane changing and wide-ranged system communication. It is more steady progress that will eventually peak with a capable traffic simulation that allows for very extensive user definition.

I advise that my model be further developed and used as starting point. Current traffic simulation models are built upon micro models, a large compilation of them. If my code can be optimized and a computer network utilized, my program can be slightly altered to set up a large set of systems adjacent to one another to depict more than just a small road system, perhaps even a whole system.



User Generated Environments

In early versions of my program the environment is hardcoded into the system, to be drawn and to be reacted with by the cars. Because this severely limits the extents to which the program can be applied, I altered it to allow for users to create their own road systems which could be loaded into the program. The user would draw road systems in an imaging editing program such as Gimp and save it in pgm (ascii) format. This stores the data in a matrix-like fashion that can be easily read in and stored. The program reads this data in and scans the edges of the drawn environment to find roads. When it finds the start (or end) of a road, it traces it and stores its edge location to be used in car generation. Some samples user environment would look something like this:



Scanning Algorithms

The scanning algorithm searches the edges of the environment to find the location of the roads. The code for scanning one edge looks like this:

```
public ArrayList scantXEdges()
      {
        ArrayList temp = new ArrayList();
         int k=0;
         int ig=0;
         for(int x=50;x<650;x++)
         {
            if(worldmatrix[x][50]==0)
               if(ig==0)
               {
                  temp.add(new Integer(x+11));
                  ig=1;
                  x+=16;
               }
               else
                  ig=0;
         }
         return temp;
      }
```