A Study of Creating Computational Models of Traffic

Madeleine E. R. Pitsch

June 9, 2005

Abstract

The goal of my project is to make an accurate simulation of traffic in a multi-lane intersection world that will be easily mutible. I will complete this goal by creating a logical organizational structure that will easily lend itself to creating situations in which to study traffic. My main concerns will be in the studies on the effects of construction work and accidents on traffic flow. I will then proceed to advance the accuracy of the traffic simulation as much as time will allow so the study will prove to be useful in its own right.

1 Introduction

Traffic simulation is a fairly complex computational model becuase of the many interactions between the agents and the world plus the duality of the agents themselves. There has been much to study in approaching and understanding traffic simulations and why they are useful tools.My goal when researching before the creation of my project was to find the main considereations when creating a traffic simulation. I then studied actual projects to find which one type would prove the most useful to elaborate on. Traffic has become a major problem in the Washington metropolitan area and in the United States in general. In 2001, Paul Krugman, a writer for the New York Times, stated that in Atlanta during 1999, for each individual that decided to drive during rush hour, the resulting cost to the other drivers was 14 dollars per day, or 3,500 dollars per year. In the 1996 census, Atlanta was rated as the fourth worst traffic congestion area in the United States, while Washington D.C. area was rated second, later dropping to third in 2001. Drivers in Washington D.C. lose an average of 36 hours in traffic per year. The Washington D.C. area is ranked fourth in the amount of extra time needed for a trip during rush hour and fifth in the average amount of time each person wastes in traffic jams. Island & T St. Bridges over North Capitol and the 23rd St. Bridge over Virginia Avenue . Construction currently taking place on the Theodore Roosevelt Memorial Bridge has already caused major traffic problems in the city. Work on the Springfield Interchange has caused traffic difficulties in Virginia and is predicted not to be finished until 2010 or later. The government is now debating whether to shut down multiple traffic lanes even during rush hour so construction can be completed in months instead of years.

2 Background Information and Theory

2.1 Uses

Traffic Simulations are used in a variety of ways. One of the most prominent and original uses was to use traffic simulations to evaluate alternate treatments. Since engineers were in control of all the variables, they could evaluate such things as signal control strategies, and speed limit management. Traffic simulations are often used to test new designs. Because roadwork is so expensive, traffic simulations can help quantify the improvement of traffic flow with different geometric arrangements. Even more so, traffic simulations can also be an element of the design process as well. Traffic simulations are also used to test traffic center personal and can also be used to recreate a traffic accident and then design a safer environment in response to that accident.

2.2 Why simulations

Simulations describe a non-static environment in a statistical or pictorial way. They can be used whenever there is a system undergoing mathematical changes over a long period of time or there is a need to view an animation of the system to understand what is causing the final results. They are also approached when the mathematical equation can not accurately factor in all the agents of a system. Traffic simulations are used to support optimization models, and new theories in management. They are an efficient way to see understand calculated data particularly if the simulation creates an animation output.

2.3 Classification

Most traffic simulations have a dependent variable of time meaning they are dynamical systems. Discrete simulations show real-world examples by having certain changes occur at a certain time. The two discrete models are: discrete time and discrete event. Discrete event models run cycles after each change in the system while discrete models merely run at time increments were situations occur according to the time. Discrete event models are a lot harder to create and model but are useful in specific cases. A traffic simulation called NETFLO was a discrete event model that considered a single facility. Discrete event models are often more efficient for models that have lots of down time but since traffic simulation is a study of continuous flow of traffic, discrete time models are more prevalent.

Simulations are also defined by there level of detail and/or fidelity to a real life system. There are three levels: Macroscopic, Mesoscopic, and Microscopic. Macroscopic systems are the least accurate and are usually use when only a simple understanding of the interactions between the vehicles is necessary. Mesoscopic are used in situations where a model with high accuracy with the entities is needed but describes their interactions at a more simple level. Microscopic is the most detailed which keeps a high level of detail in both the entity and their reactions. For examples, a microscopic model would have lane-changing or left turning while the other two levels might not. Microscopic however only refers to the possibility of a more accurate simulation. They may not be so accurate dependent on the complexity of the system. They are also very hard to create and maintain. Lower-fidelity models are much easier to create and maintain and could be complicated for the purpose created. Often the designer must have a well-developed focus to choose the correct level of fidelity for the project.

The last form of classification refers to the processes of the model. There are two types: deterministic, and stochastic. Deterministic models are models where every decision made relies on previously gathered data from a real-life situation. Stochastic models instead use random-number generators and statistics to have the entities make decisions. While the deterministic model should be more accurate, very often situations will arise where there is not gathered data. Unless the system is very focused on one type, a deterministic model would be inefficient.

2.4 Approach to simulations

A certain approach should be taken towards a building a traffic simulation. The first step is to define the problem and the model objectives. A builder must now what is the primary focus of his simulation so the appropriate model choices can be made to make the most efficient model for that purpose. The builder must now what data the simulation should output.

The next step is to then, using the previously found data, to define the situation the model is evaluating. The builder must understand the major components, and what are the major interactions of those components. He must as well identify the information that needs to be acquired.

The third step is the most complicated and is where the actual modeling occurs. The complexity of the model must be established to then identify which of the three classifications (macroscopic, mesoscopic, microscopic) will be used. Next is determine what will be the major functions of all of the components of the models and how the data will flow through the components. For that to be efficient, an appropriate hierarchy must be chosen as well. The modeling language must be chosen and all the logic in the program must be documented. The final point is the hardest which then the actual coding of the program followed soon after by debugging.

After the model has been created, the model must then be tested with input data and then followed up by acquizitionof output data. The model must be validated and evaluated on pre-specified criteria relating to the actual purpose of the model.

2.5 Specified Functions

Part three of the development model refers mainly to the creation of the model in particular the creation of the functions that interact with the different components. One of the most important is the car following function. The function that determines how one car reacts to the car in front of it is probably the most important function in a traffic simulation. The Car has to decide whether to keep the speed, accelerate, or decelerate depending on the what the car does in front of it. In the real world, the driver sees how the car in the front proceeds and makes judgment calls dependent on their own attitude. This is hard to program and also can be one of the most messy methods to deal with. It is the primary method that requires outside information of other cars and often is one of the definers for approaching the hierarchy of the entire model. The most customary parameters for this method are speeds of self and leader, separation distance, projected deceleration/acceleration methods for self and leader, and the reaction time of the follower.

Another important method if the model is stochastic is the random number generators. Because all random-generators are actually no random they have to be truly referred to as pseudo-random number generators. For stochastic model to work well, the pseudo-random number generators have to be fairly random.

An often forgotten specified function when planning is the vehicle generation method. If any of the real-world data has to be found to make a model, most of it would have to be required for this method. Particularly the volume for a particular road has to be accurate or the entire model will be fairly inaccurate. Other points have to be taken into account like if the mean volume of cars should very over time (eg. Show traffic over a period of time like an entire day with two rush hour periods).

2.6 Picking a language

Picking a language is a very important decision to make when considering a simulation. There are two types of languages to consider: general purpose languages, and simulation languages. Simulation languages are exactly what there title implies: languages created entirely for the purpose of being used to make simulations. They are often easier to use and can be very efficient. They incorporate many features that compile statistics as well as other functions common to modeling. General-purpose languages fit into two categories: procedural or object oriented. Object-oriented languages support the concept of defining an object and then can be put into a world and process data as well as react with the environment. Although it creates better analysis, object-oriented programming is much harder to program . Some of the factors that should be considered when picking a language are the expected life of the simulation model, the skills of the user community, budget, and assessment of the developers skills.

2.7 Representative Model Component

In simulation, particularly object-oriented planning an important aspect is defining the agent in which the world is created for. For traffic simulation, it is important to realize that the agent has two parts: the driver and the car. The car has fairly easy to quantify attributes such as size, acceleration limit, deceleration limit, and maximum turn radius. These attributes are easy to define and pose no problem. It is the human aspect of the agent that proves to be difficult to model. The driver has an aggression factor(will take more risky turns or be a defensive driver), response time to stimuli, and a destination point. The destination can often be ignored in a simpler program but the other two pose a problem. The simplest approach is to set the mode of people in the center with average aggression and response time. The mass of the people can form a standard bell curve with the fewest people being very risky or cautious. The response time will have a much smaller range because of the sensitivity of the data to a fairly long response time. This component has to use its own attributes to accurately give logical responses to the environments characteristics like roadway geometrics, intersection configurations, nearby driver-vehicle entities, control devices, lane channelization, and confliction vehicle movements. Some of these aspects of the environment can be ignored depending on the level of fidelity of the program (micro-, meso-, or macro-) and focus of the simulation. Others could to be very vital to the model and have to seriously be taken into account when programming the entity and its reactions to vehicles and the environment.

Typically, this is the most important activity in the creation of the model because this is where the analyst decides whether the program is an accurate and reasonable traffic simulation. Because traffic simulations tend to be very complex, certain things should be taken into account such as that some of the models features do not accurately simulate a process, the input data is not valid, the results are not detailed enough for this model, the statistical analysis is incorrect, or the model just has bugs or incorrect algorithms.

Animation displays are one of the best ways to analyze whether or not the model is an accurate representation of a traffic simulation. The analysis has to be through, but can help identify cause-effect relationships and anomalous results. Particular cause-effect relationships to look out for are where congestion starts and is it where consistent traffic jams start or only so often. Anomalous results (when traffic congestion starts in an illogical point) should be observed until they can be pinpointed to a particular behavioral or model deficiency.

When there is no animation similar methods can be used to check the reasonableness of the model. One is to execute the model in a real-world application to see if it stands up to the real world data. Another technique is to perform sensitivity tests where certain key variables such as random-number variable seeds, maximum seeds, volume of cars/area, are changed and to analyze the data for the model responses. Plotting that data can also give a good indication of how reasonable the data is.

2.8 Statistical Analysis of the Data

More often than not, statistically analyzing the data is often allotted the least amount of time in comparison to the actual coding of the program. Many simulations are often programmed to just show a situation and therefore there is no analysis is needed. Others however, have to understand that without statistically analyzing the data confidence cannot be placed on any simulation. Simulation is a sampling experiment on the computer and the data gathered has to be appropriately and statistically analyzed. One of the most popular ways to analyze to is to pick point estimates of the measures of effectiveness. Using different arrangements of a system and these point estimates of measures of effectiveness, one arrangement can prove to be the more efficient or reasonable of the others. These points can be found from only one simulation run or from a set of runs.

2.9 Predicting and Understanding Traffic Congestion

There are two types of roadways to consider when predicting traffic. Urban traffic flow is defined by the intersections or more importantly the traffic light time arrangement. But that has been debunked by that amount of traffic flow can actually by helped by the insertion of traffic lights. Traffic has also been known to increase as well with the shutting down of certain roads. Freeway traffic is a even harder to evaluate. The theory that if there were no traffic lights there would be no traffic is obviously debunked by the mile long traffic jams seen often on highways. Although many traffic jams are started because of accidents, some are created because of one underlying principle: when a car slows down the car behind it will always slow down more. When one person slows down by just tapping the breaks, the following car will see the break lights and slow down more. This interaction can actually lead to a traffic jam. In the NETLOGO traffic simulation, it was found that when the simulation started that unless every car had the same distance from the other a traffic jam would result. It would be impossible to separate each car an equal distance from each other, but this program also ignores a very important issues. People all have their own aggression factor when it comes to driving. Although the concept of a fast left lane does help counteract that, there is very little that can be done to work with the human aspect of the traffic simulation.

3 Design Criteria and Procedure

My project was made in MASON, a specific derivation of Java that is condusive to making simulations. It requires that I have not only a World.java file but a WorldWithUI file. In MASON, simulations can be run without any type of visual representation by just running the World file. The WorldWithUI file is like a wrapper for the World filea and actually converts everything into an animation that can be seen and re-animated. I have chosen to work primarily in animation mode because it helps me see if the cars are actually doing anything. I have two other class files: Car and Street.

In the world there are two data members: Land and World. The Land is the actualsparsegrid of the world where the streets are placed. The world is a doublegrid with a layout of how the streets are actually lined up together. This was done because one of my original problems was the streets had a length but that length could not shown in the sparse grid because it is only a object which therefore can only take up one unit (like the cars). WorldWithUI uses the doublegrid to project the screen in the animation becuase world is full of 1 and 0's where 1's are areas of street and 0's are areas of grass.

The land contains two objects: the streets and the token objects for cars. Each street has a data member called myArea which is where the cars are actually located. Whenever a car moves it finds its token object in World.land and moves it the same spot. This is done again so that the streets can cycle the cars and since the cars cannot be in two spots at one time there are tokens for the cars in World.land. WorldWithUI uses the token objects to project the cars on the animation.Towards thed end of the project I created three new objects for tokens so that the cars could have different colors.

Each street runs through the cars that are located in this area starting from end of the street to beginning of the street. Each car checks certain critera -current speed, the max speed of the road, and the orientation of the cars around it- to decide whether it should change either its lane or its current speed. This is where most of the work had to be done to create a logical and accurate simulation. The cars have to keep track of their own tokens by knowing their location on the road, the road's location, and from that be able to find the location of their token in the land.The cars are the main agents in this study although Streets carry some of the load. The cars are the main determiners of what happens throghout the simulation.

4 Phases & Results

There have been many phases to the detailing of my case study. Each step was made for the advancement of my project. The case study was originally created with the organization and then from there additions could be made to bring the project where I wanted it to.

4.1 One Dimensional Movement

Just as it sounds, the original step of making the cars move down a street at a constant speed to the end of the track and then off.

4.2 Speed Changes as Reactions to the Environment

The phase was based on the original creation of the cars abilities to orientate themselves to their environment and use that data to decide whether to speed up or slow down. The original method for finding the location of other cars proved faulty because it also took into account the cars behind the focused car.s location. A new method had to be created but the cars still showed too responsive reactions to the nearness and farness of the cars around them. That problem was to be refined in the later phases when testing was concerned.

4.3 Changing Lanes

Changing lanes seemed to be an easy task mainly because each street had a number of lanes set its height of its data member myArea. Using the orientation methods from the Speed Changes as Reactions to the Environment, changing lanes was created as an alternative. Although cars would not change lanes onto other cars at this moment cars were still going through each other. Changing Lanes had similar problems to Speed Changes in that the cars were too responsive.

4.4 Cars Stopping at the End of Roads

This method seemed out of place but it was the creation of an algorithm that would have the cars stop at the end of the road. Although this method was not immediately used, it would become very useful after the creation of intersections.

4.5 Continuous Flow of Traffic

This was perhaps one my first real triumphs which was creating a method that enabled the program for many turns and have cars continuously flow. It was not very difficult but began the first real showings of a traffic simulation. These methods would end up being refined as the directions of traffic were changed and multiple roads not starting at the ends were created.

4.6 Flow Changing by Turns

This was another easy and quick phase but another that made the traffic simulation more concrete. This phase was based on the idea of having turns (each time the cars moved once (could be seconds or minutes .long. depending on how .big. the pixels are thought to be). The turns were kept track of and during certain periods of time the volume of traffic would increase and decrease like normal days. Because the program could only run so fast and I have only so many minutes in a class period, I counted each turn as about a a minute having .rush hours. about every 600 turns for about 200 turns. A slightly unrealistic setup, but it did incorporate the idea of changing flows of traffic. This would become very useful when I began creating situations were cars could start piling up.

4.7 Two Directions of Traffic

This was perhaps the most difficult phase so far because it involved dealing with perhaps the major weakness of my program: the difficulty of accounting for both the cars on a street and the token in the sparse grid. This was primarily difficult because when going the other direction the value of the cars location (in the x direction) was increasing while the tokens location in the sparse grid was decreasing. After a long while I was finally able to make the cars go both directions and was pleasantly surprised that the amount of changing I had to do to my continuous flow methods was very little.

4.8 Lane to Lane Interaction

Another difficult barrier was trying to account for how to change cars to different streets while accounting for where the cars were and moving their tokens accordingly. This was just the beginning and although was changed to suit the more difficult concepts of intersections be the four way bridge between eight different street objects. In its original form, the algorithm merely checked if the car was going to go off the street, if it was then find the next street and make the link between the car and the new street. The most important part of this algorithm was making sure that the former street no longer had any link to the car. The removing of the car from the former street was where I most of my difficulties. Another difficulty was making sure that if the car reached the end of the street of a street that ended on the edges of the world, the car did not look for a new street but merely move of the street.

4.9 Creating Unique Tokens

This phase was more about making the animation easier to understand then making the program more exact. Before all of my cars had been red, and although that was acceptable for the sake of focusing on the less aesthetic ideas of my project, it made seeing where the cars went after a turn very difficult to see. This phase was surprising difficult mainly because the WorldWithUI wrapper colored the objects in the animation by color so all the tokens could be made one color and one color only. My first strategy was to one by one color the objects (tokens) randomly but this proved ineffective because I could then only color the cars that were on the world when the program began and not the cars that were generated afterwards. Thus I had to create for extensions of the Object class: ObjectRed, ObjectBlue, and ObjectGreen. They completely extended the Object class and in fact the only difference was that they had different names. Thus when I put down tokens I could randomly choose a type of token and then when the Graphic was put down I could color by the three Object classes. It was slightly convoluted but it was quite effective.

4.10 Creating the Intersection Class

After much deliberation, I decided to create an intersection class that instead of being a sister class to street under an abstract road class was in fact an expansion of the street class. This proved to be useful mainly because the cars needed areas like the street and its main difference from the street is that along with having to move cars across its area, it had to keep track of the lights of 8 different joining streets. There has been little to be done to expand upon the idea of right turns or left turns but that hopefully will be incorporated later on.

4.11 Four Way Traffic

This title is misleading mainly because I merely worked on having streets in the up/down direction and not in all four directions at once incorporating the intersection class. This phase was a good idea at this time mainly because it made up update my token getting and token setting algorithms so that they were more precise. This phase was slightly difficult because like making the cars going in the left/right directions it pushed at the weakness of finding the tokens in the world from knowing the location of the street and the car on the street.

5 End Matter

I have made much progress in these last few months on creating a usable traffic simulation. This program will be able to be changed fairly readily for anyone's studies. One drawback has been the inability to create a more effective decision making agent however algorithms could easily be instigated in a brief amount of time. Another drawback has been the inability to get real life data. This eliminates the ability to predict the traffic along any intersection but again with more time this could have been executed with my program.

This project involves a lot of work and understanding of the interactions. The main trial of this project was to create an organization where cars could move in a realistic fashion and be able ot interact with the streets and the world with little issues. This organization did hold up to all of the phases my program was put through and in fact this served as a perfect testing platform for my project.

If I had had more time I would have tried to expand on the graphical nature of the program along with developing a more accurate decision making agent. The simulation however was a success and could prove to be a useful tool in which future programmers could expand upon.

All in all my program has proved itself to be a incredible tool and canvas for future programmers to expand on. I may not have not solved any problems but I created a reliable tool in which problems can be solved on.

6 Appendix

6.1 Car.java

```
package sim.app.project;
import sim.engine.*;
import sim.util.*;
import sim.engine.*;
import sim.field.grid.*;
import sim.util.*;
import ec.util.*;




public class Car
    {

public double xdir;  // deterimines the amount of pixels moved after one turn
        public Street mystreet; // the connection to discover distance from cars
        public int myAttitude; // a number from 1-10, indicator of recklessness
        public World w;


public Object myC;
int myX =0;
int change;
int path;
int id = 0;
int myY=0;

public Car()
{
this.xdir = 1.4;
this.mystreet = null;
this.myAttitude = 5;
this.move=false;
}

    public Car(double xdir,Street mystreet,
    int change, int at, World w,int x,int y, int number)
        {
this.id = number;
this.change = change;
this.xdir = 10 + ((int)(at/4));
        this.mystreet = mystreet;
this.myAttitude = at;
this.w = w;
this.move = false;

}
public Int2D findToken(Int2D location, Int2D loco)
{
if(Math.abs(path)==1)
{
return new Int2D((loco.x+(path*location.x)), loco.y+location.y);
}
else return new Int2D(loco.x+location.y,loco.y + (path/2*location.x));

}
```

```
public Int2D setToken(Int2D newloc, Int2D loco)
{
if(Math.abs(path)==1)
return new Int2D(loco.x+(path*newloc.x), loco.y + (newloc.y));
else return new Int2D(loco.x + (newloc.y), loco.y + (path/2*newloc.x));

}
    public int run(int paths)
      {
path = paths;
     int change =0;
System.out.print(" My ID!: " + id);
path = paths;
        int b = checkLanes();
        Int2D loco = (w.land).getObjectLocation(mystreet);
Int2D location = (mystreet.myArea).getObjectLocation(this);
Object token = new Object();
        Int2D locax = findToken(location,loco);
       System.out.print(" CLS" + location.x + " , " + location.y +
       " SL " + loco.x + " , " + loco.y + " CLR " + locax.x + " , " + locax.y);
Int2D newloc = new Int2D();
Bag p = new Bag();
int q=0;
        int r=0;
        double z = CheckSpeed(location);
xdir = z;
int newx = ((int)(location.x +(z)));//set new location on the road
        System.out.print(" New xposition " + newx + " because speed was " + z);


     if(newx>mystreet.myArea.getWidth()-1 && mystreet.Light==0)
     {
Int2D locot;
     System.out.println("\nON NEW ROAD ");
     change = 1;
     newx = newx+1-mystreet.myArea.getWidth();
     if(Math.abs(path)==1)
     locot = new Int2D(loco.x+(path*mystreet.myArea.getWidth()), loco.y);
     else locot = new Int2D(loco.x,(path/2*mystreet.myArea.getWidth())+loco.y);
     System.out.print("Looking for Street at " + locot.x+ " , " + locot.y);
     Bag s = (w.land).getObjectsAtLocation(locot);//my roads location
     //Bag q = (mystreet.myArea).getObjectsAtLocation(location);

     for(int a = 0;a < s.numObjs;a++)
{
if((s.objs[a] instanceof Street))
{
mystreet = ((Street)s.objs[a]);
         System.out.print(" Have NEW STREET with location");
    break;
}
}

     loco = locot;
     //mystreet.myArea.setObjectLocation(this, new Int2D(newx + locot.x,
     locot.y));
    System.out.print( " Position of new street" + loco.x + " ," + loco.y);
     }
          newloc = new Int2D(newx,(location.y+b));


        p = (w.land).getObjectsAtLocation(locax); // find token like object
((mystreet).myArea).setObjectLocation(this,newloc);

 //p = (w.land).getObjectsAtLocation(locax);

 for(int a = 0;a < p.numObjs;a++)
{

if(!(p.objs[a] instanceof Street))
token = p.objs[a];

}
 Int2D L = setToken(newloc,loco);

 (w.land).setObjectLocation(token, L);

// }

  return change;

}
```

```
    public int checkLanes()
    {
   // System.out.println( "Check lanes");
     int answer =0;
    Int2D location = (mystreet.myArea).getObjectLocation(this);
   if(mystreet.myArea.getHeight()<2)
    return answer;
   if(mystreet.myArea.numObjectsAtLocation(location.x, location.y+1)>1)
  return answer;
   if(Orientation()<5 )
   {if(location.y==0)
   {
   answer = 1;
   //System.out.println("I'm moving up!  : Change " + this.change);
   }
   else if(location.y == mystreet.myArea.getHeight())
   {   answer= -1;
  // System.out.println("I'm moving down!  : Change " + this. change);
   }}
   return answer;
     }

    public int Orientation()
    { // System.out.println( " Orientation ");
    int x =0;

    Int2D location = mystreet.myArea.getObjectLocation(this);
   for( x  = 1; x<mystreet.myArea.getWidth()-location.x;x++)
     {
   if(mystreet.myArea.numObjectsAtLocation
      (location.x + x*path, location.y)>0)
     break;
    }
//        return x;
    }

    public double  CheckSpeed(Int2D loco)
{
// System.out.println( " Checkspeed");
double answer =xdir;
//System.out.println("\n \n\n\n\n\n My Speed:" + xdir);
if(xdir>0)
{
    IntBag front = new IntBag();
    IntBag nul= new IntBag();
   // Bag p= new Bag();
   int p;

   if(myAttitude>20)
   {
   if(Orientation()>12 & answer<myAttitude)
answer+=1;
    if(Orientation()<5 && xdir>1)
        answer-=1;
   }
    /*
   if(findDA(loco)==true)
   { if(mystreet.myArea.getWidth()-loco.x>0)
        answer-=1;

 }
   */
    return answer;
}
return answer;
}


   public boolean findDA(Int2D loco)
   {

  // System.out.println(" findDA");
   int sum = 0;
   for(int t =(int)xdir; t>0;t--)
   sum+=t;
   if(sum>=mystreet.myArea.getWidth()-loco.x)
   return true;
   else return false;
   }
```

6.2 Street.java

```
package sim.app.project;
import sim.engine.*;
import sim.util.*;
import sim.engine.*;
import sim.field.grid.*;
import sim.util.*;
import ec.util.*;





public class Street implements Steppable
{

public SparseGrid2D myArea; // area of the lane
public int Speed;// max speed cars should take
public int myX;
public int myY;
public int direction;
public  int myCars; // # of cars on road
public int Light;
public int Up;

public Street()
{
myX= 5;
}
    public Street( int dir, int myCars, int x, int y, int end)
        {

   this.Light =end;
   this.myX =x;
   this.myY=y;
   this.direction= dir;
   this.myCars = myCars;
   this.Speed = 10;
 myArea = new SparseGrid2D(x,y);
}
public void setCars( SparseGrid2D c)
{
this.myArea = c;
}

public void setCar(Car c, int e)
{
myArea.setObjectLocation(c, 0, e);
}

    public void step(SimState state)
{

    Bag p;
   System.out.println("\n\n Street with direction "
   + direction + " and location" + myX);

     for(int x=myArea.getWidth()-1; x>=0;x--)
     for(int y=myArea.getHeight()-1; y>=0; y--)
     {
   int  s = myArea.numObjectsAtLocation(x,y);
p=myArea.getObjectsAtLocation(x,y);

 if(s>0)
 {
 System.out.println( " The new specimen ");
int k =((Car)(p.objs[0])).run(direction);
                 if(k>0)
  myArea.remove((Car)(p.objs[0]));

 }
 }



}


}
```

6.3 World.java

```
package sim.app.project;
import sim.engine.*;
import sim.field.grid.*;
import sim.util.*;
import ec.util.*;



public class World extends SimState
    {
        public DoubleGrid2D world;
public SparseGrid2D land;// area of the actual world
    public int gridWidth = 100;
        public int gridHeight = 100;
        public int numCars =240;
        public int numStreet=5;
        public int turns =0;
public World(long seed)
      {
       super(new MersenneTwisterFast(seed), new Schedule(3));
      }

public Street CreateStreet(int x, int y, int direction,
int length, int number, int light )
{
Street temp = new Street( direction, 1,0,0,light);
SparseGrid2D helper  = new SparseGrid2D(length,number);
int hat = 0;

for(int k = 0; k<length; k++)
{
for(int z = 0; z<number;z++)
if(Math.abs(direction)==1)
{
world.set(x+(direction*k),y+z,1.0);
}
else world.set(x+z,y+(k*direction/2),1.0);
}
if((x==0&& direction==1)|| (y==0 && direction ==2)||
( x==gridWidth-1 && direction ==-1)||(y==gridHeight-1
&&direction==-2))
{
for(int e =0;e<number;e++)
{
       int r = 0;
int u = 0; //
int i = random.nextInt(4);
int h = random.nextInt(3);
Car n = new Car(21,temp, i, random.nextInt(40)+1, this, u,e,e);
        helper.setObjectLocation(n, new Int2D(0,i));
        temp.setCars(helper);
if(Math.abs(direction)==1)
        land.setObjectLocation(Tokenize(random.nextInt(3)), new Int2D(x,i+y));
  else land.setObjectLocation(Tokenize(random.nextInt(3)), new Int2D(x+i,y));
}
}
schedule.scheduleRepeating(temp);
land.setObjectLocation(temp, new Int2D(x,y));

return temp;



}
    public void start()
        {
turns = 0;
super.start();
land = new SparseGrid2D(gridWidth, gridHeight);
        world = new DoubleGrid2D(gridWidth, gridHeight);
int x = 0;
int y = 0;

Street k = CreateStreet(0,0,1,100,5,1);

Street l = CreateStreet(99,6,-1,100,5,1);

Steppable decreaser = new Steppable()
               {
           public void step(SimState state)
```

```
                    {
Bag p = new Bag();
turns++;
System.out.print("TURNS:  " + turns);

  for(int y=0;y<gridHeight; y++)
  {
  for(int x = 0; x<gridWidth;x++)
  {
 p=land.getObjectsAtLocation(x,y);
      if(land.numObjectsAtLocation(x,y)>0)
        if((p.objs[0] instanceof Street))
{
int direction = ((Street)p.objs[0]).direction;
  if((x==0&& direction==1)|| (y==0 && direction ==2)||
  ( x==gridWidth-1 && direction ==-1)||(y==gridHeight-1
  &&direction==-2))
newCars((Street)p.objs[0]);

}

}

  }


                    }
                        };
    schedule.scheduleRepeating(Schedule.EPOCH,2,decreaser,1);
         }


    public void nextLane(Street s, Car c)
{
/*
  Bag b;
  Int2D loca = s.myArea.getObjectLocation(c);// Car's location on Street
  Int2D loco = land.getObjectLocation(s); // Street's Location on World
  Int2D locax = land.getObjectLocation(c);// Car's location on world
  int t = (int)(c.xdir -((s.myArea.getWidth()-1)-loca.x));
Object token = new Object();
 System.out.print(" CAR's Movemetn : " + t);
```

```
  b = land.getObjectsAtLocation(loco.x + s.myArea.getWidth(), loco.y);
c.mystreet = (Street)(b.objs[0]);
 (((Street)(b.objs[0])).myArea).setObjectLocation(c, t,0);
 Bag q = land.getObjectsAtLocation(loco.x+loca.x, loco.y+loca.y);

 for(int a = 0;a<q.numObjs;a++)
{

if(!(q.objs[a] instanceof Street))
token = q.objs[a];

}
 land.setObjectLocation((token),(int)(loco.x+(newloc.y), loco.y + (newloc.x));
*/

}

public Object Tokenize(int number)
{
number =1;
if(number==1)
 return new ObjectRed();
else if (number ==2)
 return new ObjectGreen();
else return new ObjectBlue();
}

public void newCars( Street s)
{
//working on volume
int rate =0;
int color = random.nextInt(3);
if(turns%(600)>300)
   rate = 3;
else rate = 5;
for(int e = 0; e<s.myArea.getHeight(); e++)
{
Int2D loco = land.getObjectLocation(s);
        int u = random.nextInt(rate);
int i = random.nextInt(4);
     if( u>1)
{

Car p = new Car(20.5, s, i,
random.nextInt(40)+1, this, u,i,random.nextInt(10000)+12);
      s.setCar(p,i);
      if(Math.abs(s.direction)==2)
      land.setObjectLocation(Tokenize(color), new Int2D(loco.x+i,loco.y));
      else  land.setObjectLocation(Tokenize(color), new Int2D(loco.x,loco.y+i));

}
}
}
```

6.4 WorldWithUI.java

```
package sim.app.project;
import sim.engine.*;
import sim.display.*;
import sim.portrayal.grid.*;
import java.awt.*;
import javax.swing.*;
import sim.util.*;
import ec.util.*;
import sim.field.grid.*;


public class WorldWithUI extends GUIState
    {
    public Display2D display;
    public JFrame displayFrame;

    SparseGridPortrayal2D carsPortrayal = new SparseGridPortrayal2D();
    FastValueGridPortrayal2D streetPortrayal = new
    FastValueGridPortrayal2D("Trail");
    SparseGridPortrayal2D streetsPortrayal = new SparseGridPortrayal2D();

public WorldWithUI(){super(new World(System.currentTimeMillis()));}

    public static void main(String[] args)
        {
```

```
       System.out.print( " & " );
WorldWithUI t = new WorldWithUI();
System.out.print( " * ");
Console c = new Console(t);
       System.out.print( " ^");
c.setVisible(true);
        }

  //  public WorldWithUI() { super(new World(System.currentTimeMillis())); }

    public WorldWithUI(SimState state) { super(state); }

    public String getName() { return "Traffic Simulation practice"; }

    public String getInfo()
        {
        return "<H2>Traffic Simulation Demo</H2><p> Simple Car movement";
        }

    public void quit()
        {
        super.quit();

        if (displayFrame!=null) displayFrame.dispose();
        displayFrame = null;  // let gc
        display = null;       // let gc
        }

    public void start()
        {
        super.start();
        // set up our portrayals
        setupPortrayals();
        }



    public void load(SimState state)
        {
        super.load(state);
        // we now have new grids.  Set up the portrayals to reflect that
        setupPortrayals();
        }

    // This is called by start() and by load() because they both had this code
    // so I didn't have to type it twice :-)
    public void setupPortrayals()
        {
        // tell the portrayals what to
        // portray and how to portray the
System.out.print(" ! ");

    streetPortrayal.setField(
        ((World)state).world);
System.out.print( " @ ");
       streetPortrayal.setMap(
              new sim.util.gui.SimpleColorMap(
              0.0, .5,Color.green,Color.gray));
     System.out.print( " # ");
   carsPortrayal.setField(((World)state).land);
   streetsPortrayal.setField(((World)state).land);
   System.out.print(" % ");
//Car t = new Car();
int xx = ((World)state).gridWidth;
int yy = ((World)state).gridHeight;

carsPortrayal.setPortrayalForClass
(ObjectRed.class, new sim.portrayal.simple.OvalPortrayal2D(Color.red));
carsPortrayal.setPortrayalForClass
(ObjectBlue.class, new sim.portrayal.simple.OvalPortrayal2D(Color.blue));
carsPortrayal.setPortrayalForClass
(ObjectGreen.class, new sim.portrayal.simple.OvalPortrayal2D(Color.yellow));



        // reschedule the displayer

display.reset();


        // redraw the display
        display.repaint();
        }

    public void init(Controller c)
        {
        super.init(c);

        // Make the Display2D.  We'll have it display stuff later.
        display = new Display2D(400,400,this,1);
        displayFrame = display.createFrame();
        c.registerFrame(displayFrame);
        displayFrame.setVisible(true);

        // specify the backdrop color  -- what gets painted behind the displays
        display.setBackdrop(Color.gray);

        // attach the portrayals
        display.attach(streetPortrayal,"trails");
       display.attach(carsPortrayal,"Cars");
        }
    }
```

6.5 Intersection.java

```
package sim.app.project;
import sim.engine.*;
import sim.util.*;
import sim.engine.*;
import sim.field.grid.*;
import sim.util.*;
import ec.util.*;


public class Intersection extends Street implements Steppable
{
public Street[] mystreets;
public int Length;
public World myW;

        public Intersection (int type, World w)
{
myW = w;
Length = type;
 mystreets = new Street[type];
 }

public void setStreets(Street[] helpers)
{
for(int x = 0; x<Length; x++)
{
 mystreets[x] = helpers[x];
}
}

    public void step(SimState state)
    {
Bag p;
   // if(mystreet[0].Light==1)// as in the light there is GREEN yo!



     for(int x=myArea.getWidth()-1; x>=0;x--)
     for(int y=myArea.getHeight()-1; y>=0; y--)
     {
        int paths;
        int  s = myArea.numObjectsAtLocation(x,y);
p=myArea.getObjectsAtLocation(x,y);

      if(mystreets[0].Light==1)
      {
     if(x<(myArea.getWidth()/2))
      paths=-1;
      else paths=1;
     }
      else if( y<(myArea.getHeight()))
paths=-2;
else paths = 2;
```

```
     if(s>0)
 {
 System.out.println( " The new specimen ");
 System.out.println( "dir " + paths);
 int k =((Car)(p.objs[0])).run(paths); // will put variable for up down.
                 if(k>0)
   myArea.remove((Car)(p.objs[0]));
 }

 }
 }



    }
```

6.6 Tokens

6.6.1 ObjectRed.java

```
package sim.app.project;
import sim.engine.*;
import sim.util.*;
import sim.engine.*;
import sim.field.grid.*;
import sim.util.*;
import ec.util.*;

public class ObjectRed extends Object
{
public ObjectRed()
{
super();
}
}
```

6.6.2 ObjectBlue.java

```
package sim.app.project;
import sim.engine.*;
import sim.util.*;
import sim.engine.*;
import sim.field.grid.*;
import sim.util.*;
import ec.util.*;

public class ObjectBlue extends Object
{
public ObjectBlue()
{
super();
}
}
```

6.6.3 ObjectGreen.java

```
package sim.app.project;
import sim.engine.*;
import sim.util.*;
import sim.engine.*;
import sim.field.grid.*;
import sim.util.*;
import ec.util.*;

public class ObjectGreen extends Object
{
public ObjectGreen()
{
super();
}
}
```

7 References

Barcelo, J, et al. Microscopic Traffic Simulation for ATT Systems Analysis A Parallel Computing Version. Working paper. Physics of Transport and Traffic. University Duisburg-Essen Department of Physics. 9 June 2005 <http://www.traffic.uni duisburg.de/>.

Barlovic, Robert, et al. Adaptive Traffic Light Control in teh CHSCH Model for City Traffic. Working paper. Physics of Transport and Traffic. University Duis burg-Essen Department of Physics. 9 June 2005 <http://www.traffic.uni-duis

Chrobok, Roland, et al. OLSIM: Future Traffic Information. Working paper. Physics of Transport and Traffic. University Duisburg-Essen Depa rtment of Physics. 9 June 2005 <http://www.traffic.uni-duisburg.de/>.

Erol, Kutluhun, Renato Levy, and James Wentworth. Application of Agent Technology to Traffic Simulation. Working paper. 26 May 2005 <http://www.tfhrc.gov/advanc/agent.htm>.

FHWA. Revised Monograph on Traffic Flow Theory. Ed. Nathan Gartner, Dr., Carroll J Messer, Dr., and Ajay K Rathi, Dr. June 1992. FHWA. 26 May 2005 <http://www.tfhrc.gov/its/tft/tft.htm>.

Kaumann, Oliver, et al. Traffic Forecast Using On-Line Simulations. Working paper. Physics of Transport and Traffic. University Duisburg-Essen Department of Physics. 9 June 2005 <http:// www.traffic.uni-duisburg.de/>.

Mazur, F, et al. Future of Traffic Information: Online-Simulation of a Large Scale Freeway Network. Working paper. Physics of Transport and Traffic. University Duisburg-Essen Depa rtment of Physics. 9 June 2005 <http://www.traffic.uni-d uisburg.de/>.

Schadschneider, Andreas, and Michael Schreckenberg. Car-Oriented Mean-Field Theory for Traffic Flow Models. Unpublished essay. Physics of Transport and Traffic. University Duis burg-Essen Department of Physics. 9 June 2005 <http:// www.traffic.uni-duisburg.de/>.

Wahle, J, et al. Anticipatory Traffic Forecast Using Multi-Agent Techniques. Working paper. Physics of Transport and Traf fic. University Duisburg-Essen Department of Physics. 9 June 2005 <http://www.traffic.uni-duisburg.de/>.

Wahle, Joachim, et al. Decision Dynamics in a Traffic Scenario. Ms. 26 May 2005 <http://adsabs.harvard.edu/cgi-bin/nph-bib_query?bibcode=2000PhyA..287..669W&db_key=PHY>.