KING's COLLEGE LONDON

7CCSMGPR Group Project Module

TEAM KINGSMEN

TRAFFIC SIMULATOR

Authors:
Vikram Prabhu
Vijay Kumar Badatala
Praveen Allu
Nuruzzaman Shahin
Bhabishya Shrestha

Supervisors:
Dr. Laurence Tratt
Dr. Elizabeth Black

March 28, 2016

Contents

1 INTRODUCTION

Majority of road accidents that happen in today's world are because of human error. Around 1.3 million people die in road accidents every year which on an average turns out to be around 3287 deaths a day. To minimize these road accidents, we need a program like a Traffic Management System. Traffic Management Systems can improve flow of vehicles and also improve the safety of people. It would be best to develop a Traffic Simulation Software that can take care of the above factors. In this way, transportation systems can be properly mathematically modelled by the use of a computer software which can help plan and design a better and safer traffic system.

2 OVERVIEW

The aim of this '7CCSMGPR Group Project' Module is to give students the experience in developing a rather complex software application whilst helping them to face challenges during the development and also the challenges involved in working in a team/group. The aim is to build a traffic simulation engine that will help to improve the flow of traffic. The simulation engine would also help in giving way to emergency vehicles even during blocked roads so that these vehicles are able to reach their destination in times of emergencies or natural calamities.
Our approach in tackling this problem started with the requirements elicitation and will most probably be followed by designing the architecture of the software application.

3 STRATEGY

While going through various approaches/strategies, we finally decided to adopt Object Oriented Programming. We decided to split into two teams, the 'User Interface Team' and the 'Simulation Team'. We adopted the Agile Methodology which helped both the teams at each stage as the life cycle of the development of the software advances. The latter or final stages of the project mostly included software testing, bug fixing and final documentation.

4 PROJECT AIM

The aim of this project was to develop a traffic simulation software that would aid in testing various traffic management system strategies. This software would allow users to use various parameters and generate any kind of road network. Users are also able to select different vehicle types, density of the traffic, synchronization of the traffic lights, behaviour of the drivers etc. The software would generally be able to simulate vehicles such as 'Light

Vehicles', 'Heavy Vehicles' and a special category of vehicles such as 'Emergency Vehicles'. The simulation software should also be able to show for different driver behaviours such as 'Normal' Driver and 'Over-Speeding' or 'Dangerous' Driver. It also accounts for weather conditions such as Sunny and Snowy.

5 PROJECT OUTCOME

Our software accomplishes the goal by allowing the user to configure traffic pattern options contributed by different vehicle types such as cars, emergency vehicles, the traffic density for various vehicle types, the traffic lights synchronisation, driver behaviour, climatic conditions.

6 LITERATURE REVIEW

There are various techniques utilized for movement simulation software underneath we give a brief outline of some of them that are more pertinent to our methodology.

6.1 TRAFFIC SIMULATION USING AGENT BASED MODELLING

Agent Based Model is a very effective method of building the simulation for real life software project. The model comprises of entities called agents that have their individual behaviour and the decision-making heuristics of the environment its in. Agent based models cannot be estimated in differential equations which is why the agents should have an ability to adapt to the environment its in and interact with the same. Agent based models are used in the following situations:

- When the granularities of the objects are not defined.
- When it is important that all the agents have dynamic relationships with all the other agents, and the agent relationships are made and dissolved.
- When the complex communication topology can cause problems to the entire main system.
- When some of the defined mathematical equations eliminate the most important aspect of the individual behaviour which can cause damage the entire main system.

Agent based traffic models also include more measurable behaviour of the drivers for instance, each and every agent will have their individual behaviour which can affect the vision, speed and distance to keep the other vehicles from colliding.

6.2 INTRODUCTION TO TRAFFIC SIMULATION

Simulation is a successful tool utilized for imitating and analysing a wide assortment of complex issues, hard to think about by different implies that may be excessively costly or risky. Traffic can be seen as an intricate framework along these lines so simulation is a suitable instrument to analyse movement frameworks. Traffic Simulation is the best in class technique used to survey and assess transport plans for decreasing congestion. As opposed to implementing a plan without knowing whether the result will be a win, the plan can be executed in a simulation to decide its viability.
Simulations can be classified as continuous or discrete. Continuous models take the type of mathematical equation utilizing variables that compare to genuine qualities. By comprehending the mathematical equations, the condition of the model at any given point in the simulation can be figured. Discrete simulations show the reality by demonstrating the condition of the framework and its state changes after time or occasions have passed. There are two sorts of discrete simulations: discrete time models and discrete even models. Discrete time models (time-cut) are those that split the simulations into settled time interims. At every interim, the condition of the model is upgraded utilizing capacities that portray the connections. Discrete event models (event arranged) are those which keep up a line of events booked to happen all together of time, every event speaking to the change of condition of a component in the model. The test system forms the event all together, and every one can modify the event line.

6.3 MICROSCOPIC SIMULATION

Traffic Simulators can be macroscopic or microscopic relying upon the level of point of interest required. Naturally visible test systems show the stream of traffic utilizing abnormal state numerical models regularly got from fluid dynamics, in this manner they are consistent simulations. They treat each vehicle the same, and use data and yield variables, for example, velocity, stream and density. These traffic simulations can't separate between individual vehicles, and typically don't cater for various vehicle sorts. They do not have the capacity to model complex roadways, itemised movement control highlights or distinctive driver practices. Macroscopic Simulators are most helpful for the recreation of wide-range movement frameworks, which don't require detailed modelling, for example, motorway systems and interregional street systems. This methodology is not extremely reasonable

on the grounds that, in actuality, there are a wide range of sorts of vehicle driven by various people who have their own styles and practices. Notwithstanding, it is quick and can be valuable and precise, however is not suited to urban models.
Microscopic Simulators model individual substances independently at an abnormal state of point of interest, and are classed as discrete simulations. Every vehicle is followed as it associates with different vehicles and nature. Interactions are normally represented via auto taking after and path evolving rationale. Principles and regulations are characterized to control what should and can't be possible in the recreation, for instance speed limits, privileges of way, vehicle pace and quickening. Traffic flow details generally connected with naturally visible simulations are the developing properties of the minute simulations. Microscopic simulators can demonstrate traffic flow more reasonably than microscopic simulators because of the additional subtle element included displaying vehicles exclusively. Microscopic test systems are generally used to assess new activity control and administration innovations and also performing investigation of existing traffic operations.

6.4 AGENT- BASED MODELLING

Simulation can be executed utilising multi-agent frameworks; this is known as multi-agent simulation or agent based modelling. The agent worldview maps conveniently onto numerous modelling situations since every individual in the situation can be specifically spoken to as a specialists. Behaviours can be modified into the specialists so people act similarly as the element they are demonstrating. A fascinating point is that people are given basic practices, and when numerous agents are simulated as a gathering, practices regularly rise that were not unequivocally customised into the specialists; these are known as rising phenomenon.
This proposes multi-agent simulation can be a excellent prototyping tool for creating and refining models. Since it can permit parameters of the people and the earth to be fluctuated effortlessly, it is conceivable to explore different avenues regarding numerous choices and increase results in real-time.

6.5 DIFFERENCES TO OBJECT- ORIENTED PROGRAMMING

To help understanding, the agent- oriented methodology could be considered as an augmentation to the item arranged (OO) approach. Objects are processing substances that epitomise some state, have strategies to change their state, and convey by passing messages. Objects in this manner have some type of control over their state, yet they don't have control over their conduct. Agents are considered to have control over their conduct, implying

that every agent has its own particular string of control and can perform activities at whatever point it picks without being followed up on by some other substance. Despite the fact that it is workable for all items to have their own string of control, it is not fundamental to the OO idea.
Agents additionally display adaptable conduct, which is not part of the OO worldview. This implies they are fit for responsive, proactive and social conduct. Receptive means they can see and react to changes in their surroundings, proactive means they can take the activity to accomplish their objectives, and social means they can connect with different agents to satisfy their goals.

7 TRAFFIC SIMULATION

7.1 SCENARIO FOR SIMULATION

Simulators supporting a motorway situation concentrate on multiple-lane fast motorways. A significant part of the many-sided quality required for a city domain does not should be displayed, and the simulation can concentrate on vehicle conduct and communication. Motorway situations can be simulated precisely by both macroscopic and microscopic simulators. The principle features of a microscopic motorway simulators are car following and lane changing perspectives. Junctions are once in a while demonstrated, permitting passage/exit rate to be fluctuated to test the productivity of the motorway under differing movement load.

7.2 ISSUES IN MODELLING

Defining the model is one of the main phases in building a traffic simulator. This includes choosing how to show objects (e.g. vehicles, traffic lights and people) in the simulation and what parameters every object will require. It likewise includes deciding how to represent the environment (e.g. street, paths and crossing points), and the impacts it has on the other objects.

7.3 MODELLING THE VEHICLES

In a simulation, vehicles and drivers would undoubtedly be demonstrated as one entity. In any case, in this real world they clearly are not, so when settling on the best way to model them it bodes well to take a look at each in turn. Displaying a vehicle is entirely straightforward; a couple of parameters can portray its features and conduct: maximum speed, maximum acceleration and deceleration. Acceleration is particularly critical as it influences the rate of queue release. Measurements are regularly actualised, empowering trucks and transports to be recognised from cars. Amid simulation, current position and heading in nature are required to monitor the present state.

Different points of interest are not typically required in light of the fact that they add to the many-sided quality without enhancing authenticity, however a few test systems give detailed physics or engine models; these are valuable when researching outflows or the itemised impacts of effect.

7.4 MODELLING THE DRIVERS

Decision drivers need to make can be split into macro and micro objectives. Macro objectives are the destination and course taken, while micro objectives include choices at every purpose of time in light of a legitimate concern for accomplishing the macro objective. The macro objective includes day by day arranging and course generation usefulness, frequently enter from O-D information. Micro objectives are choices including controlling the vehicle, for example, desired speed, overtaking and turning. Drivers all have distinctive driving styles, which are represented by their individual attributes, for example, forcefulness, certainty and driving knowledge. Driving styles are frequently approximated utilising parameters for essential behavioural components and after that these qualities are utilised as a feature of the thinking procedure to work out the choice a driver would take at any moment. Illustrations of these parameters are:

- Favoured speed: the velocity the driver likes to keep up on an empty street.
- Favoured rate of acceleration: which is not usually the vehicle's maximum rate, but rather a rate impacted by a driver's forcefulness and longing for comfort among different attributes.
- Favoured rate of deceleration: which is similar to acceleration.
- Gap Acceptance: which is the distance a driver likes to keep between himself and the vehicle ahead of him.

These parameters can show a considerable measure of driver attributes. A reckless driver would have a high favoured speed, a high favoured rate of acceleration and deceleration, and a little gap acceptance. A cautious driver would be much the inverse. Many simulators consolidate these parameters, and exhibit sensible driving conduct. To change over these parameters into actions, a thinking process must be assessed at every phase of the simulation.

7.5 ENVIRONMENT MODELLING

In traffic simulation, environment in which vehicles drive is a street network which is comprised of connection sections, hubs (intersections and crossing points) and control highlights which are normally part of the hub. Every connection can have one or more paths, and might work in one or both

directions. Hubs can be separated into the accompanying essential sorts: uncontrolled non-priority intersections, priority intersections, roundabouts, signal controlled intersections and grade partition intersections. Extraordinary instances of connection exist, for example, merges where two paths must converge into one, wanders where one path parts into two and weaving segments where a diverge nearly follows a merge. Cases of other control components are traffic calming plans and people intersections.
Models typically represent the network as a gathering of connections joined by hubs. Hubs have properties, for example, area (e.g. co-ordinates) and sort of hub. Joins have properties, for example, length, number of paths, speed limit, and so on. For street networks that are not independent, sources and sinks are made as positions in the system where vehicles arrive and leave individually. Sources have parameters that decide the rate of entry of vehicles into the network, and sinks dependably uproot vehicles that go to them.

7.6 SIMULATION ISSUES

Models of vehicles, drivers and the environment are of no utilisation unless they can be controlled amid simulation. Simulation includes utilising behavioural guidelines to change the model after some time. This includes moving every vehicle in view of its parameters and its driver's choices. To mimic a single car on a long straight street is fairly simple as the vehicle would simply accelerate to the driver's favoured speed. It turns out to be progressively complex as different vehicles, more paths, and more streets are presented; to manage these, vehicle following and lane changing models are utilised.

7.6.1 Vehicle Following

Vehicle following models are those that endeavour to portray how a vehicle acts in standard conditions. These models are called such on the grounds that they concentrate fundamentally on the relationship between the current car and the one ahead of it. Numerous vehicle following models are discussed, and a number of these depend on anti-collision ideas.
One understood model of this sort is the Gipps model, created in 1981. It expects that a vehicle dependably intends to have the capacity to stop securely if the one ahead of it performs an immediate stop. It ascertains the new speed of the accompanying vehicle utilising the flow speed of both vehicles, the accompanying driver's desired pace, maximum acceleration and deceleration, and the distance between vehicles. It ascertains two conceivable qualities for rate; one in light of the driver's sought speed restricted by vehicle execution, and one in view of security, i.e. the speed that should be taken to guarantee there is no collision. It then chooses the minimum

of these conceivable speeds to use in the simulation, which guarantees both impediments are applied.
Another class of models are known as psycho-physical vehicle following models. These models endeavour to catch both physical and human segments of vehicle control. They do this by keeping up a vehicle state, where the current state is resolved through contrasts in speed and the distance to the main vehicle. Every state has an alternate technique for figuring the acceleration of the vehicle, and upon a state change, the new acceleration is ascertained once and is then consistent until the next change of state. Since the acceleration is not figured at every time venture in the simulation, it has been demonstrated that the event arranged time advance strategy is extremely effective for this model. State changes can be resolved and recorded in a line so they can be handled all together.

7.6.2 Lane Changing

Lane changing models are those that depict when and how vehicles move to another lane on a multi-lane street, for example, a motorway. The behaviour of every vehicle relies on the vehicles around it, and every vehicle needs to settle on choices taking into account its information of the environment. Path changing models have been studied to a comparable extent as vehicle following models, and a few vital models have been conceived.
Lane change issues rise up from street rules, for instance in the U.K. passing is ordinarily just permitted on the right and vehicles ought to attempt to keep on the left lane. Driver qualities can likewise have influence, since a few drivers like to stay in the same lane regardless of the fact that moving to another lane would bring about a higher speed, and a few drivers want to stay in outer lanes regardless of the fact that there is space in the inner lanes.

8 REQUIREMENTS AND DESIGN

8.1 INITIAL REQUIREMENTS, GOALS AND PLAN

The various approaches that could be used for the development of the simulation engine in NetLogo were considered and assessed.

8.1.1 NetLogo

NetLogo is an agent based modelling environment intended for simulating complex regular and social phenomena. It is written in Java and is in this manner cross-stage, it is freeware, and has a huge benevolent client bunch. It empowers the client to model any number of agents in a variable-size environment utilising a basic programming language got from Logo. It is

intended for use by students and analysts to investigate the conduct of customised agents under differing conditions. It takes after the theory of 'low threshold, no ceiling', implying that new clients ought to think that its simple to begin, however advanced clients ought not think that its restricting. NetLogo accompanies broad documentation, tutorials, and more than 300 example models that exhibit all parts of the device. Models can be saved as Java Applets and keep running on website pages by any client with the Java Virtual Machine installed. It is conceivable to see the present condition of the environment and specialists in 2D or 3D, and operators can be given any vector shape to show their sort. Commands can be as techniques that are called by buttons on the interface, or entered straightforwardly in the summon console on the fundamental board. Its logo-esque language is very common, permitting code to be anything but difficult to read, write and understand.

NetLogo has a well designed graphical interface and interface manufacturer in one that permits the novice and expert alike to run, change and create models effortlessly. It gives numerous implicit gadgets to change simulation parameters at runtime, including sliders, catches, and drop-down menus, and permits yield as charts and variable screens. Simulation time is measured in discrete 'ticks', and simulation pace can be balanced by a slider over the presentation. It can give deterministic simulation if the random seed is set before the simulation is run.

Since all the given requirements were generic, we revisited and re-defined the requirements to generate a formal requirements report.

8.2 SOFTWARE DESIGN

8.2.1 Interaction Diagrams

Sequence Diagrams are a sort of interaction diagrams that demonstrate the stream of messages through a framework as a specific function is being executed. Below are two sequence diagrams portraying the sequence occurring during a simulation tests. It portrays a single time-step, and amid typical use, these sequences could be rehashed for the simulation duration.

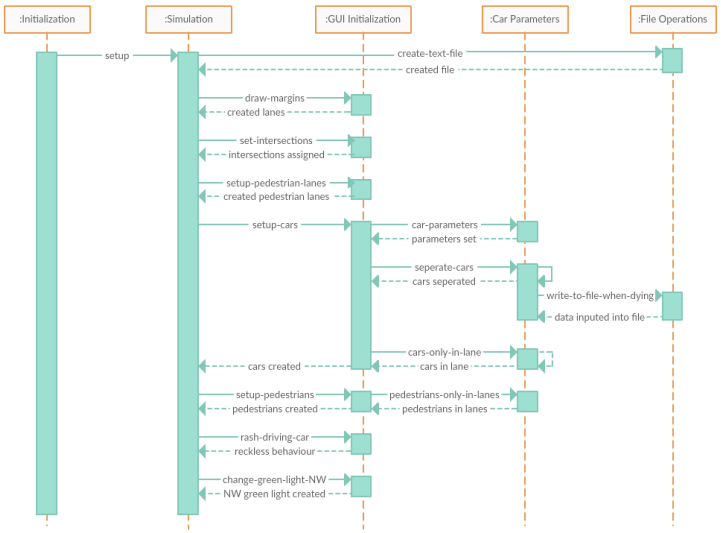

Figure 1: Sequence Diagram for "setup" procedure.

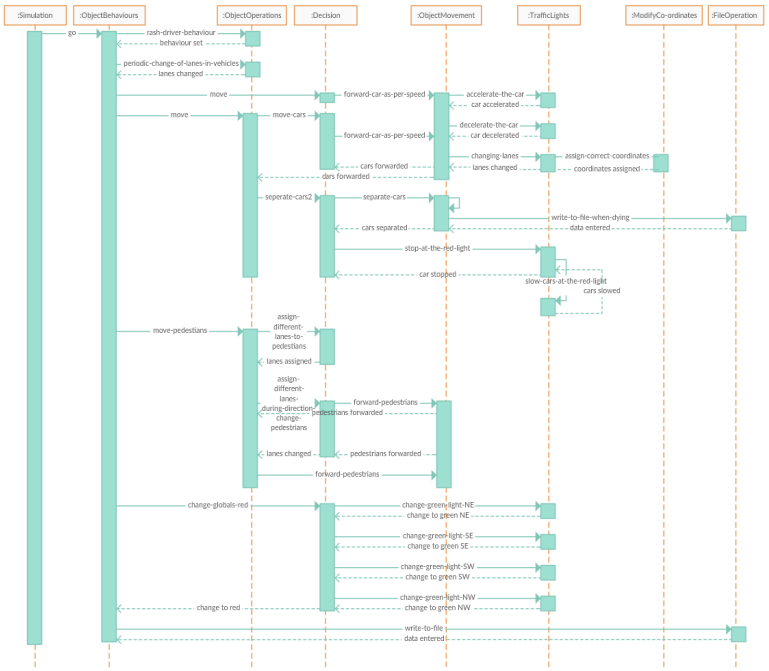

Figure 2: Sequence Diagram for "go" procedure.

8.2.2 Class Diagram

NetLogo gives numerous inbuilt turtle and patch variables, just the ones which are significant to the project are appeared on the chart. All properties and system are open since any agent can query one other, in this way zero visibility will be shown. Despite the fact that NetLogo does not give an object - oriented environment, conceptually the framework will be composed using OO procedures. Although every system could be called by any agent, every procedure is particular to one sort or type of agent. In this diagram, blue classes are those given by NetLogo, and white show those components included for this software.

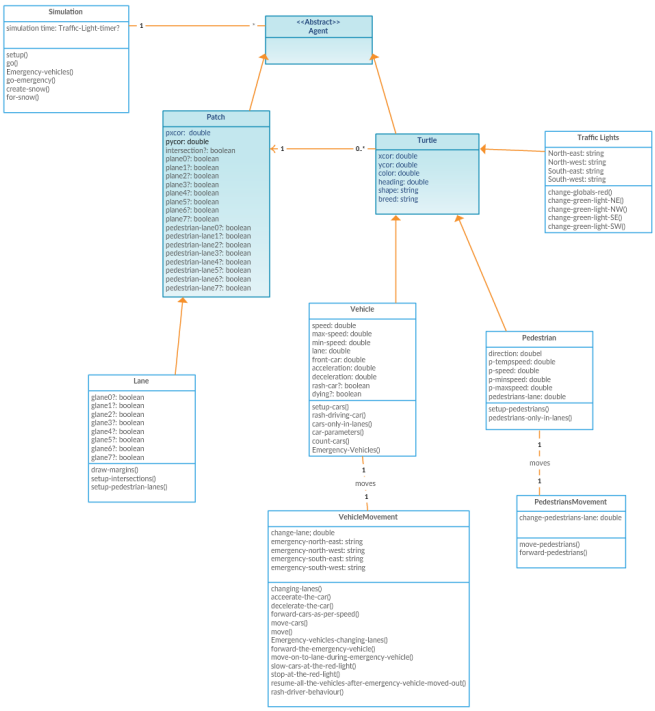

Figure 3: Class Diagram.

9 SOFTWARE DEVELOPMENT, TESTING AND DOCUMENTATION

We adopted the Agile Methodology of software development. As mentioned earlier, the group was divided into two teams namely User Interface Team which is the Front End Team and the Simulation Team which is the Back End Team. Each team was assigned major tasks and other sub tasks so that the whole group worked in parallel collaboration. Regular meetings were

held for further task identifications, task allocations, development, testing etc.
For software design and architecture, we approached the problem using the following architectural methodologies and patterns:

- We decomposed the problem and defined the use case.
- We identified the various components in the environment, their interactions and dependencies with UML. Also we identified the attributes and methods of each sub component using ArgoUML.
- We grouped the related components into different controllers and developed the contracts of the interface with the method signatures for the interactions between the User Interface and Simulation Engine.

10 PROJECT PLAN

As shown in the Gantt Chart below, we carefully segregated the time allotted to complete the development of the software project. We gave around 2-3 weeks to the planning phase. In this phase we learnt a new simulation software called NetLogo as most of the group members didn't have prior programming experience. The second and undoubtedly the longest phase was that of Development Phase. Each team segregated their tasks in sprints of around 7 days in which development, unit, functional testing, error testing and integration of units were involved. This was then followed by the testing phase done in the final week in which we carried out the performance testing. The Documentation or Report Generation was carried out as and when we were developing the system.

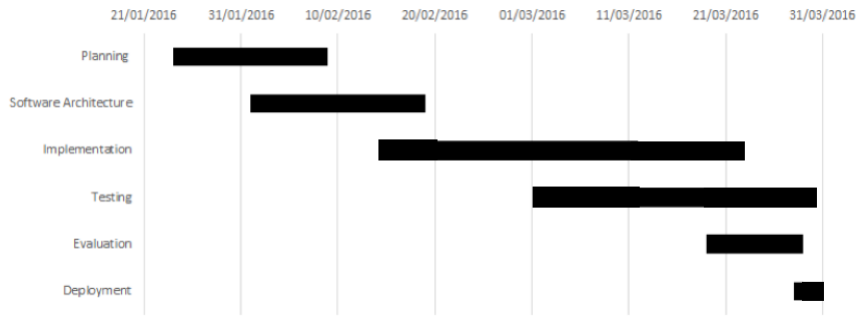

Figure 4: Gantt Chart.

Main Aims:

1. Selection of different traffic light cycles, various types of drivers, types of vehicles etc.

2. Implementation of driver behaviour.

3. Reports to be generated for various traffic management strategies.

11 USER INTERFACE (FRONT-END) DESIGN AND IMPLEMENTATION

The User Interface Team was responsible for the design of the system.

11.1 Design of Lanes

We designed a 4-road junction with 2 lane roads, where we grouped certain number of patches into one lane of a road. For example: Patches with x co-ordinate less than or equals to -3 and y co-ordinate equals to 3 constitute lane 0.

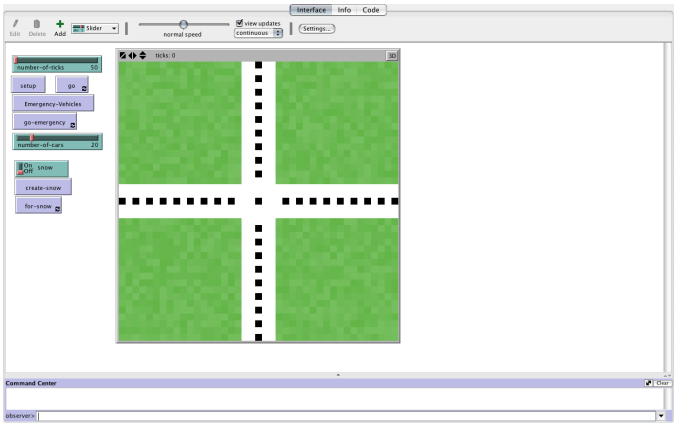

Figure 5: Design of Lanes.

11.2 Design of Traffic Lights

We used patches to generate traffic signals which have been highlighted in the below figure. The patches circled in blue are the 'Red' traffic lights which signals the vehicles to halt. The patch circled in red is the 'Green' traffic light which signals the vehicle to proceed further.

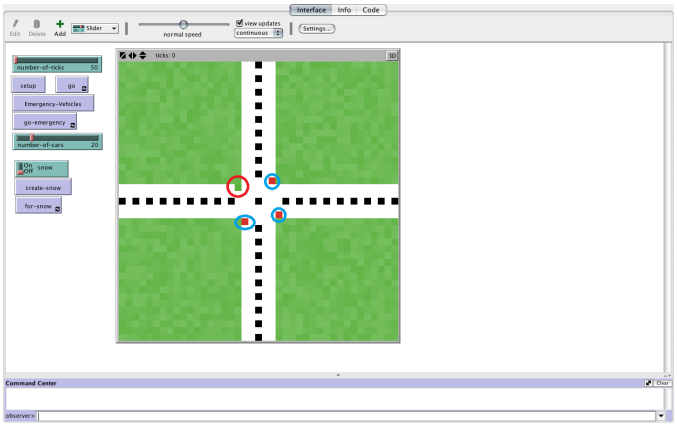

Figure 6: Design of Traffic Signalling System.

11.3 Generation of Pedestrian System

We used turtles and patches to design the complete Pedestrian System, where a number of patches are grouped together to design a pedestrian lane which is in black colour as shown in the figure below. Whereas, the turtles are used to show the pedestrians on the pedestrian lane. For example: The turtles circled in yellow are some of the pedestrians.
We have used different colours for different pedestrians so that we can easily differentiate between the pedestrians who are going in different directions.

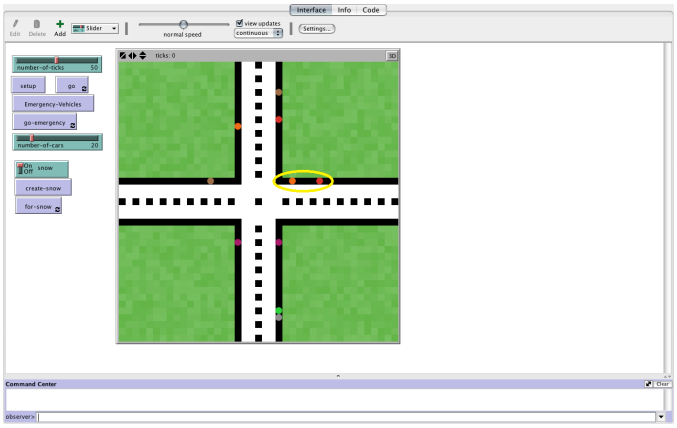

Figure 7: Pedestrian System Design.

11.4 Generation of Cars

We used different turtles to show the various cars on the lanes.

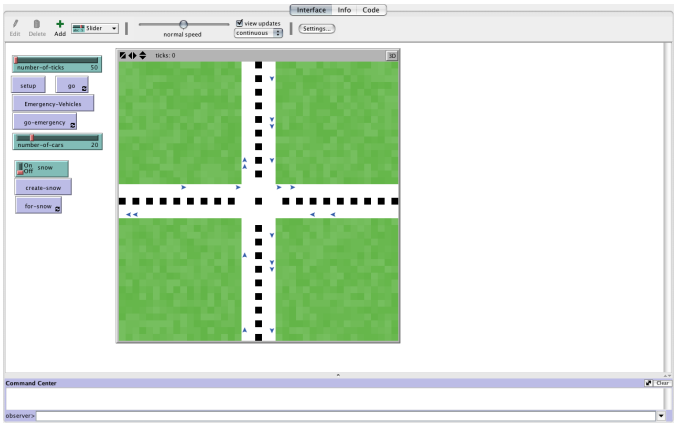

Figure 8: Generation of Cars.

In order to identify the Emergency Vehicles, we have designed the emergency vehicles as highlighted below. We can also distinguish the Emergency Vehicles with the unique siren given.

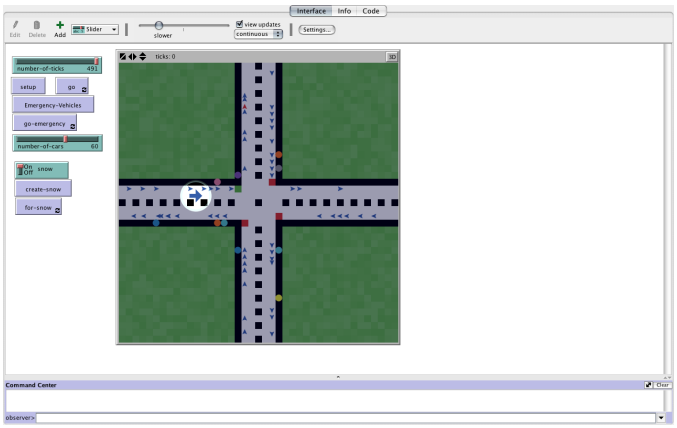

Figure 9: Emergency Vehicles.

11.5 Driver Behaviour

We have distinguished the driver behaviour in 2 ways namely:

- Cautious or Normal Driver
- Over-speeding or Dangerous Driver

The Cautious Driver is a driver who drives at a regulated speed. On a general basis we have considered every normal driver as cautious by nature. For example: The cautious driver is highlighted in the figure below.

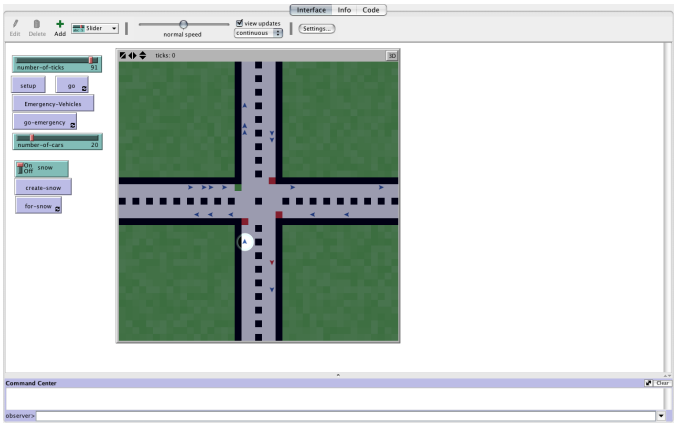

Figure 10: Cautious or Normal Driver.

The Dangerous driver is a driver who always over-speeds while driving and always tries to overtake the other vehicles rashly. For example: The Dangerous Driver is highlighted in red colour by default as shown in the figure below.

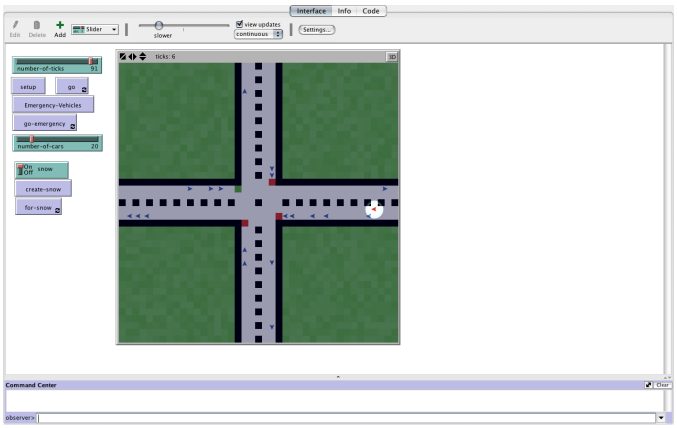

Figure 11: Over-speeding or Dangerous Driver.

11.6 Weather Conditions

We have considered 2 different weather conditions:

- Normal Condition
- Snowy Condition

In Normal condition we have considered the drivers drive the cars at a regular speed without any precautions.

In Snowy Condition we have considered the drivers drive the car at a relatively slower speed when compared to normal conditions due to the difficulty of driving the cars in the snowy weather.

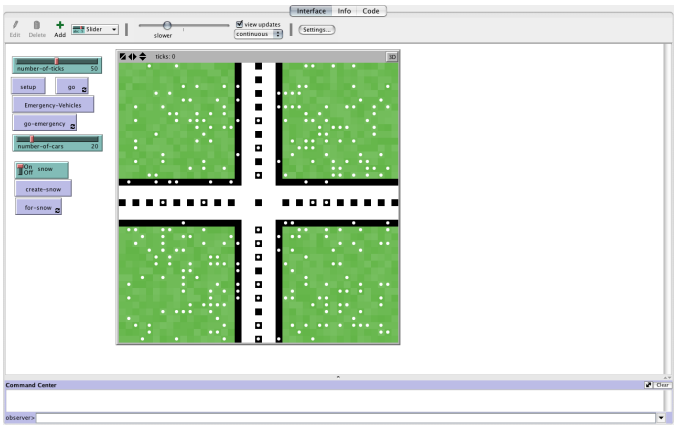

Figure 12: Snowy Weather Condition.

12 SIMULATION (BACK-END IMPLEMENTATION)

12.1 PATCHES AND TURTLES IN NETLOGO

- NetLogo teaches programming concepts using agents in the form of turtles, patches, links and observer.
- NetLogo allows exploration by modifying switches, sliders, choosers, inputs, and other interface elements
- In NetLogo, patches are agents with a fixed location, always exist and are arranged as a grid. Turtles are agents that are able to move. They have implicit access to variables of the patch they are on.
- NetLogo world has x-axis and y-axis with origin at (0,0) [origin can be changed by the user]. Each co-ordinate corresponds to one patch. Therefore each patch will have a dedicated x-cor and y-cor.

12.2 IMPORTANT PROCEDURES USED IN SIMULATION (BACK-END IMPLEMENTATION)

NetLogo, even though it implements agent based modelling, conceptually object oriented approach is used to design the system. We don't have any concept of classes in this simulator but we do program procedures to attain some functionalities and these procedures can be called by any agent.

In general, we design procedures corresponding to a particular breed or agent of only one type. In our project we used four breeds namely cars, ambulances, pedestrians and water. They are initiated by the following lines of code in the project: breed[cars car] breed[pedestrians pedestrian] breed[ambulances ambulance] breed[water]

The two main methods in our project which call some of the important procedures are:

1. setup()
2. go()

NetLogo simulator has two important tabs:

- Interface
- Code

Interface – In this tab one can actually see the simulation scenarios and can select different behaviors using sliders, choosers, buttons, etc.
Code – In this tab documentation of the code is done. Depending upon the way we code in this tab, interface tab will change the NetLogo world accordingly. The user interface tab of our project looks as follows:

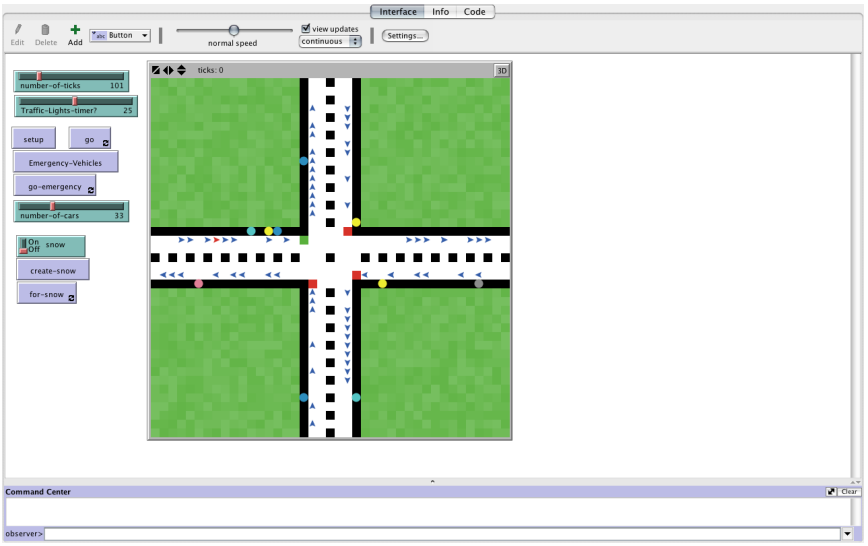

Figure 13: User Interface.

Now, user can start simulation by clicking setup button on the interface. We already uploaded a sequence diagram which depicts all the procedures which were initiated by the setup procedure.
Among the procedures called by setup(), some of the most important methods are :

12.2.1 Draw-margins

This method is used to generate lanes along x-axis and y-axis in the NetLogo world. We categorize lane into eight types. In our project, Traffic simulator, in order to create road effect, we divided whole NetLogo world in such a way that some group of patches corresponds to a single lane. The patches corresponding to each lane are as follows:
Lane0 - patches with pxcor < -2 and (pycor = 2 or pycor = 1)
Lane1 - patches with pxcor < -2 and (pycor = -2 or pycor = -1)
Lane2 - patches with pxcor > 2 and (pycor = 2 or pycor = 1)
Lane3 - patches with pxcor > 2 and (pycor = -2 or pycor = -1)
Lane4 - patches with (pxcor = 2 or pxcor = 1) and pycor > 2
Lane5 - patches with [(pxcor = -2 or pxcor = -1) and pycor > 2
Lane6 - patches with (pxcor = 2 or pxcor = 1) and pycor < -2
Lane7 - patches with (pxcor = -2 or pxcor = -1) and pycor < -2
During simulation, the following are the lanes in NetLogo World

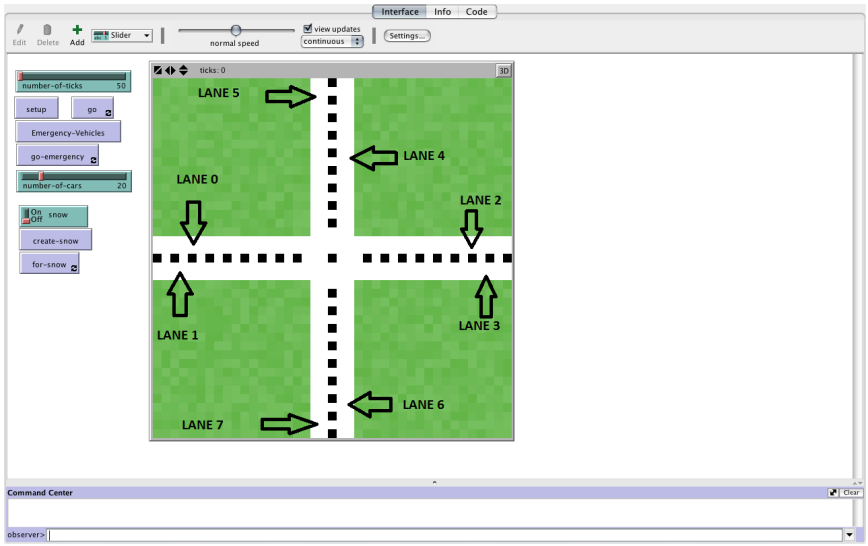

Figure 14: Lanes.

12.2.2 Setup-cars

This procedure is responsible for creating turtles (which are treated as cars in our project). This method in-turn instantiates another method namely "car-parameters" which assigns various parameters like speed, min-speed, max-speed. etc to all the cars. The following are calibrating parameters for vehicles in our project
Maximum-speed : 90mph
Minimum speed : 45mph

Acceleration: 4m/(sec xsec)Deceleration : 8 m/ (sec x sec)
While assigning various parameters to cars, we observed some random behaviour like turtles got undesirable xcor and ycor values, random heading. To overcome this unpredictable pattern we used some other code fragments namely

1. cars-only-in-lanes : This part of code is used to make cars move on patches according to their lanes which is determined by the variable "lane".

2. assign-correct-coordinates: when lanes turned into other lanes, this method makes sure to place them in the acceptable lanes as per their turn and this is decided by the variable "change-lane".

12.2.3 separate-cars

When creating cars for the first time, there is a chance of placing more than one turtle on one patch. If we allow this, it leads to serious problems as we don't get desired behaviour because of more than one car at one place. To overcome this problem, we used "separate-cars" procedure.
This procedure will compel every turtle to verify whether there are any other turtles on the very same patch it presents. If present, the turtle will move forward one patch ahead. In this way, turtle moves forward until no other turtle is present on that location except itself.
The code of this method is as follows:

```
to separate-cars

      if any? other cars-here
       [
         ifelse ( [pcolor] of patch-ahead 1 != red )
         [
          fd 1
          separate-cars
         ]
         [
           set dying? true
           write-to-file-when-dying
           die
         ]
       ]
end
```

Figure 15: Separate-Cars Code Snippet.

12.2.4 forward-cars-as-per-speed

After creating cars (turtles), it is very important to make them move similar to the real world vehicles on the roads. The code for this method is as follows:

This is done by code fragment "forward-cars-as-per-speed". This procedure in turn uses two other small methods namely

```
to forward-cars-as-per-speed

 |        set front-car one-of cars-on patch-ahead 1
          ifelse ( front-car = nobody )
          [ accelerate-the-car ]
          [ decelerate-the-car ]


         if  ( speed < min-speed ) [ set speed min-speed + 1 ]
         if  ( speed > max-speed ) [ set speed max-speed  ]

        fd speed
        changing-lanes
        ;separate-cars
end
```

Figure 16: Forward-Cars-As-Per-Speed Code Snippet.

a) accelerate-the-car : This code is implemented by the cars to accelerate themselves when there is no car ahead of them. Each car will know its front car by the variable "front-car". The code for this method is as follows:

```
to accelerate-the-car

   set speed speed +  acceleration
end
```

Figure 17: Accelerate-the-Car.

b) decelerate-the-car: As the name of the method "decelerate-the-car" signifies, it decelerates the car if there is any other car infront of it thus achieving real life behavior of cars. The code for this is as follows :

```
to decelerate-the-car
     set speed [speed] of front-car - deceleration
end
```

Figure 18: Decelerate-the-Car.

12.2.5 change-green-light-NW

As our problem here is a traffic simulator, traffic lights forms an integral part of the system. In our project, we have 4-lane junction. Therefore we considered traffic lights at four different positions namely

1. North-west
2. North-east
3. South-east
4. South-west

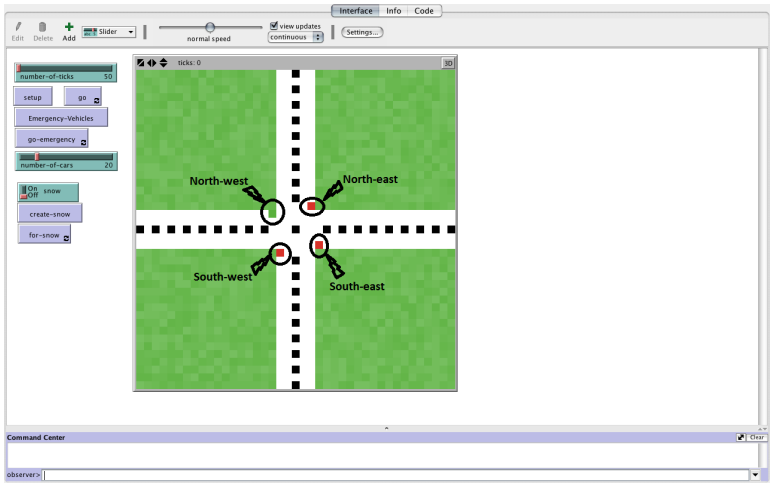

Figure 19: Traffic Lights.

In NetLogo world, during simulation the traffic lights are placed at respective locations as shown in the figure.

Now, the code of the method "change-green-light-NW" is used to make North-west light as green and remaining as red. The code snippet for the above method is as follows:

```
to change-green-light-NW

      if ( North-west = "green" )
       [
         ask patch -3 2   [ set pcolor green ]
         ask patch  2 3   [ set pcolor red ]
         ask patch  3 -2  [ set pcolor red ]
         ask patch -2 -3  [set pcolor red ]
       ]

end
```

Figure 20: Changing Green Light Code Snippet.

The other methods which follow the same pattern as above are

- change-green-light-NE
- change-green-light-SE
- change-green-light-SW

12.2.6 go()

User can initiate "go" procedure by clicking go button on the interface tab. When one observers closely there is a small symbol on the bottom right side which clearly depicts that it is a forever button i. e. If you click

this button once, all the methods which will be called by go procedure are instantiated repeatedly until you click it once again. (In general, once you click any forever option enabled button, it will remain black signifying that it is running repeatedly and this can be clearly visible as the tick number goes on increasing. Once you click it again, it will turn normal and the method calling is stopped)
The important methods that are being called by the go procedure are:

12.2.7 move()

This is one of the most trivial methods among all others. It is responsible for the movement of cars depending upon the traffic lights. For example, if North-east is green", then all the vehicles in the lanes 0, 7, 3 should be static and vehicles in the lanes 1, 2, 5, 4, 6 can move according to their speeds. This is achieved by calling move method repetitively. This move method in turn call other methods namely

1. move-cars()
   For example, if a vehicle moves from lane0 to lane2, then the lane variable associated with that vehicle should be updated from 0 to 2 because at the moment the vehicle is in lane2. This is very important issue to be addressed as we have coded our project entirely around dividing the whole NetLogo world into lanes. If this was not addressed, then we couldn't have achieved any normal real life behavior like lane changing. The procedure "move-cars" is responsible for making vehicles following that behavior of updating of its lane variable depending upon the lane on which they stood. A code snippet from move-cars method is as follows:

```
to move-cars

  if ( lane = 2 )
  [
    if ([plane0?] of patch-here )
    [
      set lane 0
    ]
  ]
  if ( lane = 0 )
  [
    if ([plane2?] of patch-here )
    [
      set lane 2
    ]
  ]

  if ( lane = 4 )
  [
    if ( [plane6?] of patch-here )
    [
      set lane 6
    ]
  ]
```

Figure 21: Move Cars Code Snippet.

2. separate-cars2()
   When North-west is green, eventually North-east, South-east and South-west are red. Thus vehicles with lane variable equal to 4, 3, 7 should stop moving. In real life, when there is a red light at a junction, all the vehicles will stop in a row right before the red light thus following the traffic rules.
   Moreover, in our project we used microscopic level approach, in which vehicle-following forms one of the main criteria. This real life behaviour is achieved by cars calling "separate-cars2" method repetitively while they are moving on lanes.
   This method uses another method namely "stop-at-the-red-light", as the name suggests stops the vehicles at the red light. When cars call these methods "separate-cars2" and "stop-at-the-red-light", the following behaviour can be observed during the simulation.

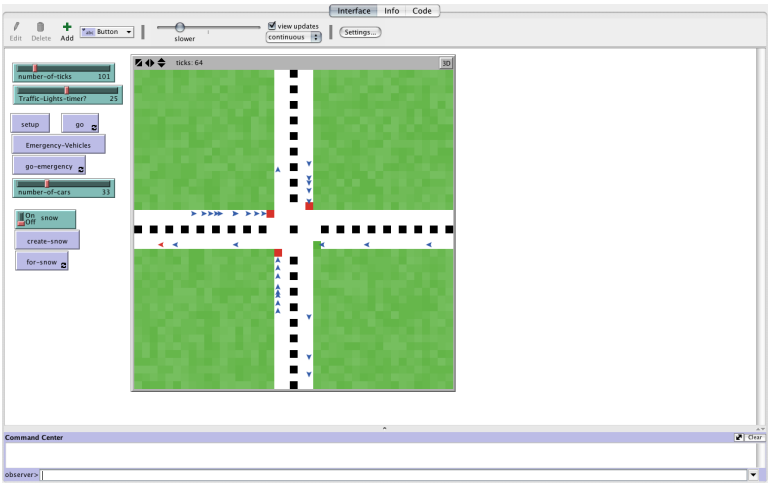

Figure 22: Separate-cars2 Code Snippet.

12.2.8 Change-globals-red()

The other important method that is employed by the go() procedure is "change-globlas-red()". Showing traffic lights at the junctions is not itself address the problem of traffic simulation rather they should be changed periodically as in real life traffic light system. This is achieved as follows:
On interface tab, we gave an option to select from a slider namely "Traffic-Lights-timer?". This value is used as a parameter to change traffic lights systematically. For example, if user chooses 25 as the value for that slider, then for every 25 ticks traffic lights change from one color to another color. So, if at present the colors at

North-west = “red”
North-east = “red”
South-east = “green”
South-west = “red” and Traffic-Lights-timer? = 30 and tick = 260 (let us assume it is updated momentarily just now). Then, immediately when tick = 290, automatically the following changes will be observed in traffic light.
North-west remains “red”
North-east remains “red”
South-east turns “red “
South-west turns “green “(Thus allowing lanes from lane7 to move).
The above can be verified by the following diagrams

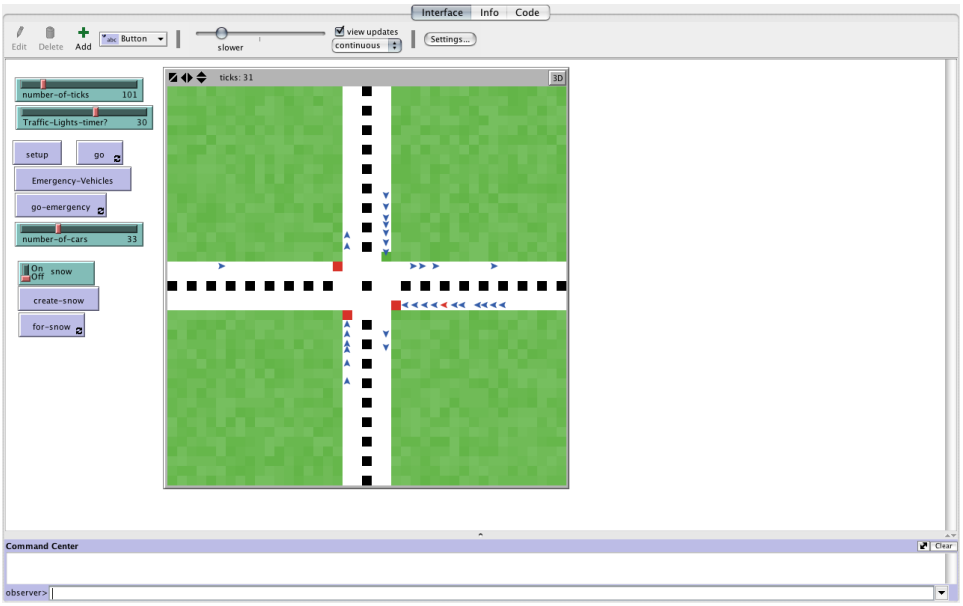

Figure 23: North-east-green.

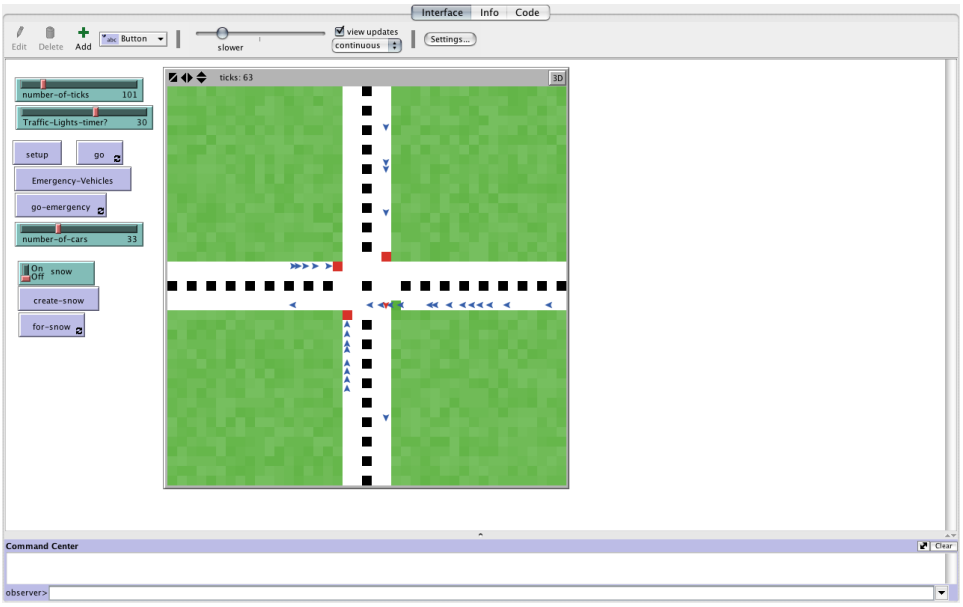

Figure 24: South-east-green.

File Operations:
In our project, one of the main requirement is to generate a report which collectively maintains all the data regarding the objects that are employed in simulation i. e. cars, pedestrians, emergency vehicles, etc. For this to be achieved we used to file procedures namely :

1. create-text-file
2. write-to-file

12.2.9 create-text-file()

The code under this method is as follows:

```
to create-text-file
  file-open user-new-file

end
```

Figure 25: creat-text-file.

In this procedure, we use a command namely "file-open user-new-file" which will enable the user to create a text file and can save the file at his desired location. The following screen-shot explains this diagrammatically.

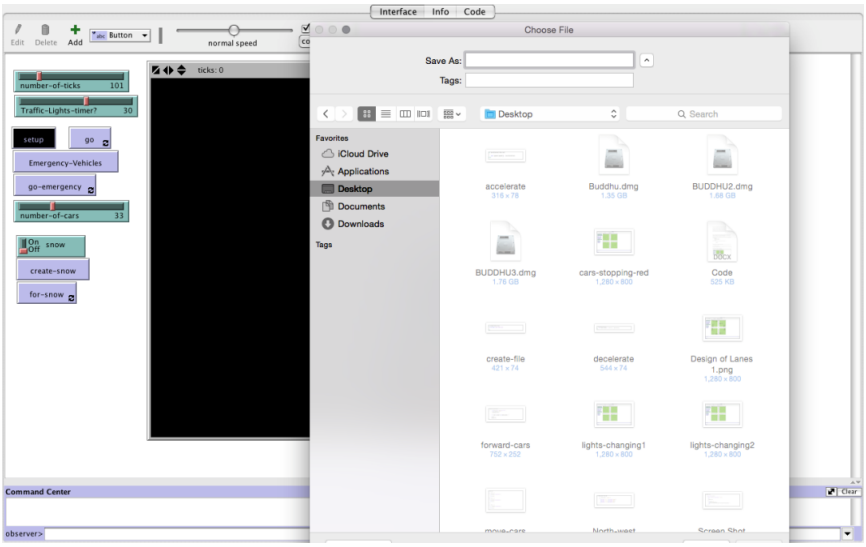

Figure 26: Creating file.

After creating a file, we have to enter data regarding the simulation for every tick. This is fulfilled by calling "write-to-file" method at the end of "go()" procedure. The code of this "write-to-file" is as follows:

```
to write-to-file
  file-print " "
  file-print ( word " Tick Number - " ticks )
  file-print ( word " ------------------------------------------------------------" )
  ask cars
  [
    file-print ( word " Who - " who "---speed - " speed "---lane - " lane " ---wait-time - " wait-time " ----front car - " front-car)
    file-print " "
  ]

  if ( counter >= 1 )
  [
    file-print( word " **********************")
    file-print ( word " Average wait time of all cars is - " mean [ wait-time] of cars )
    file-print( word " **********************")

  ]
end
```

Figure 27: write-to-file.

13 SOFTWARE TESTING

Testing finishes an assortment of things, however in particular it quantifies the quality of the software an individual is developing. This perspective presupposes there are defects in the software holding up to be found and this perspective is once in a while disproved or debated upon.

13.1 UNIT TESTING

The primary goal of unit testing is to take the most minute piece of the testable software in the application, separate it from the remaining part of the code, and determine whether it behaves exactly as it is supposed to act. Each unit is tested separately before integrating it into the modules to test the interfaces between those modules. NetLogo 'Test' Extension lets one define and run unit tests for NetLogo code.
We ensured that the units worked manually. However, we didn't have a proper unit test case coverage. Hence, it was carried out only for the selective important methods by the developers.

13.2 FUNCTIONAL TESTING

Functional testing is a testing process used within software development in which the software is tested to check if it adheres to all the requirements. Functional testing is a good way of checking a particular software to check if it has all the required functionalities that is specified within all the functional requirements.
It was the responsibility of each team to ensure that the features to be released for the sprint were checked before its integration. It was also necessary to ensure that the feature did not slow down the existing functionalities.

13.3 ERROR TESTING

Tests will be completed to guarantee that each and every requirement is implemented accurately; these requirements will be ticked off as they are executed and tested. Different situations will be tested to guarantee that no blend of events can bring about an error at runtime. These tests will be continuous and will be determined as and when they occur, in this way they don't require unequivocal test plans. Some illustration test situations are:

- Running a simulation without any vehicles.
- Running a simulation without any roads.
- Attempt to create more number of vehicles than the number of roads.
- Enter invalid inputs through sliders.

13.4 TESTING ENVIRONMENT PROPERTIES

OS: Windows 7 Home Premium
Architecture: 64 bit
Processor: Intel(R) Core(TM) i5-2450M
CPU Processing Speed: 2.50 GHz

14 TEAM WORK

14.1 DIVISION OF TEAMS

As mentioned earlier, we adopted the Agile Development Methodology for the development of our software. The group was divided into two sub teams namely the User Interface Team (Front End) and Simulation Team (Back End), with each of the team and its members focusing on one of the two major components at a high level of architecture. Each team then recognised their major tasks and sub-tasks to ensure cross team and parallel development. We followed a sprint model with regular status updates in each scrum meeting for the identification of tasks, allocation of tasks, development, testing, bug fixing and the documentation of each Sprint.

14.2 WORK DIVISION

The Agile Development approach was adopted. This helped the two teams, the User Interface Team and the Simulation Team to collaborate well among each other. During the meetings, the teams identified the tasks involved, divided the tasks among the group members and each member was assigned a role. The members of the team met twice every week to discuss important matters which could not be well explained over text messages, emails or phone calls.
Members and their respective roles:

- Project Coordinator: Vikram Prabhu
- Software Architect: Bhabishya Shrestha managed most of the tasks. Vijay Kumar, Praveen Allu, Nuruzzaman Shahin and Vikram Prabhu also contributed to the tasks.
- Quality Tester: Nuruzzaman Shahin managed most of the testing phase. Vijay, Bhabishya, Vikram and Praveen also contributed to the testing phase.
- User Interface Developers: Praveen Allu headed the User Interface Team. The other developers in the User Interface team were Bhabishya Shrestha and Nuruzzaman Shahin.
- Simulation Developers: Vijay Kumar headed the Simulation Team. The other developer in the Simulation team was Vikram Prabhu.

14.3 COLLABORATION TOOLS

We used the following collaboration tools:

- Design and Architecture: ArgoUML

- Project Management: Freedcamp
- Version Control: GitHub
- Documentation Report: LateX
- Means of communication: WhatsApp Messenger, Gmail, Outlook

15 CRITICAL EVALUATION OF THE PROJECT

15.1 THE STRENGTHS

15.1.1 THE STRENGTHS IN CONCEPT

Our product allows the user to model the road network very closely to the real-life traffic scenario and make high level infrastructure choices. For example:

- User can model the diversity of vehicles driver behaviours and road/weather conditions.
- Allowing a traffic engineer to test any traffic light configuration and their synchronisation at any point in the simulation. This helps him engineer efficient traffic flow while ensuring safety.

The behaviour of the vehicles has been attained by maintaining minute variations in the behaviours of various drivers, which in-turn:

- The desired speed of the vehicle.
- The safe distance it keeps between other vehicles in front of it.

15.1.2 THE STRENGTHS IN THE APPROACH CHOSEN

Once we chose the approach, our group spent a large amount of time just considering various scenarios, drawing the UML diagrams and improving the concept before even starting a single line of code. After decomposing our system into its independent components and classes, we allocated the work amongst ourselves in a mature and non- conflicting way. This way, we avoided conflicts among the group members. It also prevented any repetition in code which would directly lead to wastage of efforts. This, along with a clear understanding of our skillset and matching responsibilities according to our strengths allowed us to successfully develop this project.

15.2 THE WEAKNESSES

Any piece of design and code of a software is unavoidable without errors and bugs due to time constraints which highly matters in this current project. We have addressed most of these issues during the different stages of the software development life cycle. But the following issues were identified but unable to address due to lack of time and resources:
The scope of improvement for automated unit testing
The source code entirely was intensively tested manually during all the stages of development and integration with the functional, integration and performance testing but it only failed to accomplish a sufficient level of automated unit testing coverage. This was done during the development stage.
Scope of better allocation of resources and time for identified improvements
At the start, the leading of the testing task was assigned to a member exclusively but during the course of the project development it was distributed amongst the whole group due to lack of the assigned member's involvement. A much better job could have been done by identifying the potential of the team and allocating the time and resources to enhance the areas we all were lacking in.

16 FUTURE SCOPE

- User Interface enhancements with appealing road network design and simulation.
- Roads with more than two lanes.
- The Undo and Redo functionality for road network design and configuration.
- Implementing 3 Dimensional road networks which include bridges.
- Destination based routing for more than one destination.

17 PEER ASSESSMENT

We agreed to equally split the 100 points amongst the five group members. This was be done without the knowledge of the other group members where in each member allocated their points to the other group members based on their performance and contribution to the project. Each member had a total of twenty points which was allocated to the other four members of the group excluding himself.
We agreed to use this method of assessment as this was an unbiased method. There was a slight problem when it came to the two teams – User Interface Team and the Simulation Team as the members of one team did not know the

contribution and performances of the members in the other team. Hence, we then came to a conclusion that each member of the KingsMen Group would contribute to each and every task and that's when assigning the number of points to each member became easier.

17.1 Handling the Team Conflict

Conflicts arise even amongst the best of friends. All the conflicts that arose in the KingsMen Group was handled in a matured manner without creating a fuss about anything at all even in the absence of a member during the group meetings. The same was followed even in the accomplishment of deadlines for particular tasks.

- If any member was absent for any group meeting, it was the sole responsibility of the member to update himself by checking FreedCamp and also his emails to see what he had missed out and also to check if any task was assigned to him with its corresponding deadline.
- There were times during the meetings or discussions when there was a difference of opinion among the members of the group. An unbiased opinion was taken into consideration. The most beneficial option for the project was then agreed upon.
- If a particular member failed to meet any of the task deadlines, it was dealt with high priority.

We assigned the following points to each member

Member Name	**Points Allocated**
Vikram Prabhu	20
Vijay Kumar Badatala	20
Praveen Allu	20
Bhabishya Shrestha	20
Nuruzzaman Shahin	20

18 BIBLIOGRAPHY

1. NetLogo User Manual: http://ccl.northwestern.edu/netlogo/docs/
2. Wilensky, U. (1998). NetLogo Traffic 2 Lanes model. http://ccl.northwestern.edu/netlogo/models/Traffic2Lanes. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL
3. North, M.J., and Burkhart, R.M. (2002). Agent-Based Methods, Toolkits, and Techniques. In M. J. North, C. M. and D. L. Sallach (Eds.), Proceedings of the Agent 2002 Conference on Complex

Interaction and Social Emergence (pp. 3-10). IL: Argonne National Laboratory and University of Chicago

4. Tisue, S., and Wilensky, U. (2004). NetLogo: Design and Implementation of a Multi-Agent Modeling Environment. Paper presented at the Agent2004 Conference, Chicago, IL Steven F. Railsback; Volker Grimm (2011). *Agent-Based and Individual-Based Modeling: A Practical Introduction.* Cambridge: Princeton University Press. ISBN 978-0-691-13674-5

5. Uri Wilensky; William Rand (2015). *An introduction to agent-based modeling: Modeling natural, social and engineered complex systems with NetLogo.* Cambridge: MIT Press. ISBN 978-0-262-73189-8.

6. José M. Vidal (2010). *Fundamentals of Multiagent Systems Using NetLogo*

7. TOMAS, V.R. and GARCIA, L.A., 2005. A cooperative multiagent system for traffic management and control, *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, 2005, ACM Press pp52-59

19 APPENDICES

19.1 GITLOG

Author	Date	Message
mohammad.syed@kcl.ac.uk	2016-02-09	Initial report update
vikramprabhu	2016-02-09	update intial report
vikramprabhu	2016-02-09	Revert “Initial report update”
vikramprabhu	2016-02-09	latex code update
Jay0505	2016-03-17	Create source code
Jay0505	2016-03-17	Update source code
Jay0505	2016-03-17	Merge pull request #2 from Jay0505/Jay0505-patch-1
Jay0505	2016-03-17	Update source code
Jay0505	2016-03-17	Merge pull request #3 from VikramPrabhu/Jay0505-patch-1
Jay0505	2016-03-17	Update source code
Jay0505	2016-03-17	Merge pull request #4 from VikramPrabhu/Jay0505-patch-1
VikramPrabhu	2016-03-17	Update source code
VikramPrabhu	2016-03-17	Merge pull request #5 from VikramPrabhu/VikramPrabhu-patch-1
Jay0505	2016-03-17	Updating Source Code
VikramPrabhu	2016-03-17	Update source code
VikramPrabhu	2016-03-17	Merge pull request #7 from VikramPrabhu/VikramPrabhu-patch-1
Jay0505	2016-03-17	Merge pull request #6 from VikramPrabhu/Jay0505-patch-1
VikramPrabhu	2016-03-17	Update source code
VikramPrabhu	2016-03-17	Merge pull request #8 from VikramPrabhu/VikramPrabhu-patch-2
Jay0505	2016-03-17	Update source code
Jay0505	2016-03-17	Merge pull request #9 from VikramPrabhu/Jay0505-patch-2
BhabishyaShrestha	2016-03-17	Update source code
BhabishyaShrestha	2016-03-17	Merge pull request #10 from VikramPrabhu/BhabishyaShrestha-patch-1
VikramPrabhu	2016-03-17	Update source code
Jay0505	2016-03-17	Update source code
VikramPrabhu	2016-03-17	Update source code
Jay0505	2016-03-17	Update source code
Jay0505	2016-03-17	Merge pull request #11 from VikramPrabhu/Jay0505-patch-3
praveenallu	2016-03-21	Update source code
praveenallu	2016-03-21	Merge pull request #12 from VikramPrabhu/praveenallu-patch-1
Jay0505	2016-03-21	Update source code
Jay0505	2016-03-21	Merge pull request #13 from VikramPrabhu/Jay0505-patch-4
BhabishyaShrestha	2016-03-21	Update source code
BhabishyaShrestha	2016-03-21	Merge pull request #14 from VikramPrabhu/BhabishyaShrestha-patch-2

BhabishyaShrestha	2016-03-21	Update source code
BhabishyaShrestha	2016-03-21	Merge pull request #15 from VikramPrabhu/BhabishyaShrestha-patch-3
VikramPrabhu	2016-03-21	Update source code
VikramPrabhu	2016-03-21	Merge pull request #16 from VikramPrabhu/VikramPrabhu-patch-3
NuruzzamanShahin	2016-03-21	Update source code
NuruzzamanShahin	2016-03-21	Update source code
NuruzzamanShahin	2016-03-21	Merge pull request #17 from VikramPrabhu/NuruzzamanShahin-patch-1
Jay0505	2016-03-21	Update source code
Jay0505	2016-03-21	Merge pull request #18 from VikramPrabhu/Jay0505-patch-5
praveenallu	2016-03-21	Update source code
praveenallu	2016-03-21	Merge pull request #19 from VikramPrabhu/praveenallu-patch-2
praveenallu	2016-03-21	Update source code
praveenallu	2016-03-21	Merge pull request #20 from VikramPrabhu/praveenallu-patch-3
Jay0505	2016-03-21	Update source code
NuruzzamanShahin	2016-03-22	Update source code
NuruzzamanShahin	2016-03-22	Merge pull request #21 from VikramPrabhu/NuruzzamanShahin-patch-2
VikramPrabhu	2016-03-22	Update source code
VikramPrabhu	2016-03-22	Merge pull request #22 from VikramPrabhu/VikramPrabhu-patch-4
praveenallu	2016-03-22	Update source code
praveenallu	2016-03-22	Merge pull request #23 from VikramPrabhu/praveenallu-patch-4
Jay0505	2016-03-24	Update source code
Jay0505	2016-03-24	Merge pull request #24 from VikramPrabhu/Jay0505-patch-6
VikramPrabhu	2016-03-24	Update source code
VikramPrabhu	2016-03-24	Merge pull request #26 from VikramPrabhu/VikramPrabhu-patch-6
VikramPrabhu	2016-03-24	Update source code
VikramPrabhu	2016-03-24	Merge pull request #27 from VikramPrabhu/VikramPrabhu-patch-7
BhabishyaShrestha	2016-03-29	Update source code
BhabishyaShrestha	2016-03-29	Merge pull request #28 from VikramPrabhu/BhabishyaShrestha-patch-4
VikramPrabhu	2016-03-29	Update source code
VikramPrabhu	2016-03-29	Merge pull request #29 from VikramPrabhu/VikramPrabhu-patch-8
VikramPrabhu	2016-03-29	Update source code
VikramPrabhu	2016-03-29	Merge pull request #30 from VikramPrabhu/VikramPrabhu-patch-9
praveenallu	2016-03-29	Update source code
praveenallu	2016-03-29	Merge pull request #31 from VikramPrabhu/praveenallu-patch-5
praveenallu	2016-03-29	Update source code

praveenallu	2016-03-29	Merge pull request #32 from VikramPrabhu/praveenallu-patch-6
VikramPrabhu	2016-03-30	Create README.md
VikramPrabhu	2016-03-30	Merge pull request #33 from VikramPrabhu/VikramPrabhu-patch-10
VikramPrabhu	2016-03-30	Update README.md
VikramPrabhu	2016-03-30	Update README.md
VikramPrabhu	2016-03-30	Update README.md
VikramPrabhu	2016-03-30	Update README.md
VikramPrabhu	2016-03-30	Update README.md
VikramPrabhu	2016-03-30	Update README.md
VikramPrabhu	2016-03-30	Update README.md
VikramPrabhu	2016-03-30	Create Documentation
VikramPrabhu	2016-03-30	Delete Documentation
VikramPrabhu	2016-03-30	Create Final Report
VikramPrabhu	2016-03-30	Delete Final Report
VikramPrabhu	2016-03-30	Create Final Report
VikramPrabhu	2016-03-30	Create Ini Rep
VikramPrabhu	2016-03-30	Uploaded Initial Report
VikramPrabhu	2016-03-30	Delete Ini Rep
VikramPrabhu	2016-03-30	Delete initial_report_*team_KingsMen*.tex
VikramPrabhu	2016-03-30	Delete Initial Report (team_KingsMen).pdf
VikramPrabhu	2016-03-30	Rename source code to Source Code
VikramPrabhu	2016-03-30	Create Final-Report
VikramPrabhu	2016-03-30	Delete Final-Report
VikramPrabhu	2016-03-30	Create Fin Rep
VikramPrabhu	2016-03-30	Delete Final Report
NuruzzamanShahin	2016-03-30	Update Source Code
NuruzzamanShahin	2016-03-30	Merge pull request #34 from VikramPrabhu/NuruzzamanShahin-patch-3
NuruzzamanShahin	2016-03-30	Update Source Code
NuruzzamanShahin	2016-03-30	Merge pull request #35 from VikramPrabhu/NuruzzamanShahin-patch-4
BhabishyaShrestha	2016-03-30	Update Source Code
BhabishyaShrestha	2016-03-30	Merge pull request #36 from VikramPrabhu/BhabishyaShrestha-patch-5
BhabishyaShrestha	2016-03-30	Update Source Code
BhabishyaShrestha	2016-03-30	Merge pull request #37 from VikramPrabhu/BhabishyaShrestha-patch-6
BhabishyaShrestha	2016-03-30	Update Source Code
BhabishyaShrestha	2016-03-30	Merge pull request #38 from VikramPrabhu/BhabishyaShrestha-patch-7

BhabishyaShrestha	2016-03-30	Update Source Code
BhabishyaShrestha	2016-03-30	Merge pull request #39 from VikramPrabhu/BhabishyaShrestha-patch-8
VikramPrabhu	2016-03-30	Create LaTeX Source
VikramPrabhu	2016-03-30	LaTeX Source Folder of Initial Report
VikramPrabhu	2016-03-30	Delete LaTeX Source
Jay0505	2016-03-30	Update Source Code
Jay0505	2016-03-30	Merge pull request #40 from VikramPrabhu/Jay0505-patch-7
VikramPrabhu	2016-03-30	Created UML Diagrams
VikramPrabhu	2016-03-30	Merge pull request #41 from VikramPrabhu/VikramPrabhu-patch-11
Jay0505	2016-03-30	Update Source Code
VikramPrabhu	2016-03-30	Added files via upload
Jay0505	2016-03-30	Merge pull request #42 from VikramPrabhu/Jay0505-patch-8
VikramPrabhu	2016-03-30	Merge pull request #43 from VikramPrabhu/VikramPrabhu-UMLDiagrams
VikramPrabhu	2016-03-30	Delete uml
Jay0505	2016-03-30	Update Source Code
Jay0505	2016-03-30	Merge pull request #44 from VikramPrabhu/Jay0505-patch-9
Jay0505	2016-03-30	Update Source Code
Jay0505	2016-03-30	Merge pull request #45 from VikramPrabhu/Jay0505-patch-10
Jay0505	2016-03-30	Update Source Code
Jay0505	2016-03-30	Merge pull request #46 from VikramPrabhu/Jay0505-patch-11
Jay0505	2016-03-30	Update Source Code
Jay0505	2016-03-30	Merge pull request #47 from VikramPrabhu/Jay0505-patch-12
Jay0505	2016-03-30	Update Source Code
Jay0505	2016-03-30	Merge pull request #48 from VikramPrabhu/Jay0505-patch-13
VikramPrabhu	2016-03-30	Created Folder for Screenshots and Images
VikramPrabhu	2016-03-30	Merge pull request #49 from VikramPrabhu/VikramPrabhu-Images
VikramPrabhu	2016-03-30	Uploaded Screenshots and code snippets
VikramPrabhu	2016-03-30	Delete Img
Jay0505	2016-03-30	Update Source Code
Jay0505	2016-03-30	Merge pull request #50 from VikramPrabhu/Jay0505-patch-14
Jay0505	2016-03-30	Update Source Code
Jay0505	2016-03-30	Merge pull request #51 from VikramPrabhu/Jay0505-patch-15
Jay0505	2016-03-31	Update Source Code
Jay0505	2016-03-31	Merge pull request #52 from VikramPrabhu/Jay0505-patch-16
praveenallu	2016-03-31	Update Source Code

praveenallu	2016-03-31	Merge pull request #53 from VikramPrabhu/praveenallu-patch-7
praveenallu	2016-03-31	Update Source Code
praveenallu	2016-03-31	Merge pull request #54 from VikramPrabhu/praveenallu-patch-8
praveenallu	2016-03-31	Update Source Code
praveenallu	2016-03-31	Merge pull request #55 from VikramPrabhu/praveenallu-patch-9
praveenallu	2016-03-31	Update Source Code
praveenallu	2016-03-31	Merge pull request #56 from VikramPrabhu/praveenallu-patch-10
praveenallu	2016-03-31	Update Source Code
praveenallu	2016-03-31	Merge pull request #57 from VikramPrabhu/praveenallu-patch-11
praveenallu	2016-03-31	Update Source Code
praveenallu	2016-03-31	Merge pull request #58 from VikramPrabhu/praveenallu-patch-12
praveenallu	2016-03-31	Update Source Code
praveenallu	2016-03-31	Merge pull request #59 from VikramPrabhu/praveenallu-patch-13
VikramPrabhu	2016-03-31	Update Source Code
VikramPrabhu	2016-03-31	Merge pull request #60 from VikramPrabhu/VikramPrabhu-patch-12
VikramPrabhu	2016-03-31	Update Source Code
VikramPrabhu	2016-03-31	Merge pull request #61 from VikramPrabhu/VikramPrabhu-patch-13
VikramPrabhu	2016-03-31	Update Source Code
VikramPrabhu	2016-03-31	Merge pull request #62 from VikramPrabhu/VikramPrabhu-patch-14
VikramPrabhu	2016-03-31	Update Source Code
VikramPrabhu	2016-03-31	Merge pull request #63 from VikramPrabhu/VikramPrabhu-patch-15
VikramPrabhu	2016-03-31	Updated code
VikramPrabhu	2016-03-31	Merge pull request #64 from VikramPrabhu/VikramPrabhu-patch-16
Jay0505	2016-03-31	Updated Source Code
Jay0505	2016-03-31	Updated Source Code
VikramPrabhu	2016-03-31	Guidelines to run the simulation
VikramPrabhu	2016-03-31	Not for GitHub but for the Project
VikramPrabhu	2016-03-31	Source Code PDF File

19.2 SOURCE CODE

```
;source code

extensions [ sound ] ;; to produce various sounds during simulation
especially ambulance

breed[pedestrians pedestrian]
breed[cars car]
breed[ambulances ambulance]
breed[water]


globals
[
  no-of-cars-met-with-accident ;;to determine how many cars met with
an accident
  emergency-dead               ;; to determine whether emergency
vehicle moved out of scope or not
  emergency-lane               ;; used as a temporary variable to
store the lane of ambulance
  ecounter                     ;; used in determining when to make
ambulance move out of space
  emergency-north-east         ;; used to store the
  emergency-north-west         ;; state of traffic lights
  emergency-south-east         ;; when emergency vehicle
  emergency-south-west         ;; created

  pedestrian-lane0             ;;
  pedestrian-lane1             ;;
  pedestrian-lane2             ;;  To group Which patches
  pedestrian-lane3             ;;  corresponds to which
  pedestrian-lane4             ;;  pedestrian lane
  pedestrian-lane5             ;;
  pedestrian-lane6             ;;
  pedestrian-lane7             ;;

  glane0                       ;;
  glane1                       ;;
  glane2                       ;;  To group which patches
  glane3                       ;;  corresponds to which
  glane4                       ;;  road lanes
  glane5                       ;;
  glane6                       ;;
  glane7                       ;;

  intersections                ;;;;co-ordinates where all the lanes
meet
  North-west                   ;;;; patch at (-2,2)
  North-east                   ;;;; patch at (2,2)
  South-west                   ;;;; patch at (-2,-2)
  South-east                   ;;;; patch at (2,-2)

  counter
  snow-counter
  tempcounter
```

```
  tempcounter1
  Is-ambulance-already-created?

]



patches-own
[
  pedestrian-lane0?             ;;
  pedestrian-lane1?             ;;
  pedestrian-lane2?             ;; To determine which
  pedestrian-lane3?             ;; patch belongs to
  pedestrian-lane4?             ;; which pedestrian lane
  pedestrian-lane5?             ;;
  pedestrian-lane6?             ;;
  pedestrian-lane7?             ;;

  plane0?                       ;;
  plane1?                       ;;
  plane2?                       ;; To determine which
  plane3?                       ;; patch belongs to which
  plane4?                       ;; road lanes
  plane5?                       ;;
  plane6?                       ;;
  plane7?                       ;;

  intersection?

]



cars-own
[
  wait-time                   ;; For how much time a car is waiting
at a red signal
  actual-speed                ;; speed of the turtle corresponding
to real world
  actual-acceleration         ;; acceleration of the car in the
real world
  actual-deceleration         ;; deceleration of the car in the
real world
  speed                       ;; speed of each car
  max-speed                   ;; maximum speed of the car
  min-speed                   ;; minimum speed of the car
  lane                        ;; randomly deciding on to which lane
car has to be placed
  actual-lane                 ;; depending upon the lane assigned,
randomly assigning x-cor or y-cor
  change-lane                 ;; new lane into which each car has
to be changed
```

```
  front-car                    ;; if there is any car in the front
  acceleration                 ;; tells how much a car should
accelerate
  deceleration                 ;; tells how much a car should
decelerate
  rash-car?                    ;; To determine whether a driver is
reckless or not
  dying?                       ;; To know whether a car is met with
an accident or not
]


ambulances-own
[
  ambulance-actual-speed         ;; speed of the ambulance
corresponding to real world
  ambulance-actual-acceleration  ;; acceleration of the ambulance
corresponding to real world
  ambulance-actual-deceleration  ;; deceleration of the ambulance
correspondhin to real world
  front-vehicle                  ;; determines which vehicle is
infront of the ambulance
  speed                          ;; speed of the ambulance
  max-speed                      ;; maximum speed an ambulance can
reach
  min-speed                      ;; minimum speed an ambulance
should maintain
  decision-speed                 ;; used to move the ambulances from
halt state to movement state
  lane                           ;; lane on which ambulance should
be moved
  change-lane                    ;; determines the lane into which
ambulance should take a turn
  acceleration                   ;; accleration of the ambulance
  deceleration                   ;; deceleration of the ambulance
]



pedestrians-own
[
 direction                       ;; direction in which pedestrians
should move along the lanes
 p-tempspeed                     ;; to temporarily store the speed
of the pedestrian
 p-speed                         ;; speed of the pedestrians
 p-minspeed                      ;; minimum speed a pedestrian
should move
 p-maxspeed                      ;; maximum speed a pedestrian
should move
 p-acceleration                  ;; acceleration of the pedestrian
 p-deceleration                  ;; deceleration of the pedestrian
 pedestrian-lane                 ;; lane on which a pedestrian is
```

```
moving
 change-pedestrian-lane          ;; lane onto which a pedestrian
should change while taking a turn
]


water-own
[
  snow-speed                     ;; speed of the snow turtles
]

;;----------------------Procedure to create snow turtles when snow
option is selected-------------------------------------------------
to create-snow

  ifelse ( snow )
  [

 create-water 100
 [
   set snow-speed 2
   set color white
   set shape "circle"
   set size 0.5
   set heading 180
   set xcor random min-pxcor
   set ycor random max-pycor
 ]

  create-water 100
 [
   set snow-speed 2
   set color white
   set shape "circle"
   set size 0.5
   set heading 180
   set xcor random min-pxcor
   set ycor random min-pycor
 ]
  create-water 100
 [
   set snow-speed 2
   set color white
   set shape "circle"
   set size 0.5
   set heading 180
   set xcor random max-pxcor
   set ycor random min-pycor
 ]
  create-water 100
 [
   set snow-speed 2
   set color white
   set shape "circle"
```

```
   set size 0.5
   set heading 180
   set xcor random max-pxcor
   set ycor random max-pycor
 ]
  for-snow
  ]
  [
    user-message ( word "Turn snow on ")  ;; Asking user to select
snow option first before creating snow turtles
  ]
end

;;-----------------------Procedure responsible for snow turtles to
move so as to create snow effect-----------------------------------
to for-snow

  if ( snow )
  [
      ask water
  [
    fd snow-speed
  ]

    if ( ticks = number-of-ticks )
    [
    ask water [ die ]
    stop
    ]
  ]
end

;;--------------------------------Procedure responsible for creation
of lanes along axis, for creating cars,
;;                                intersections, and also
initializes all the variables used in subsequent
;;
procedures-------------------------------------------;;

to setup

  clear-all
  reset-ticks
  create-text-file
  set snow-counter 0
  set Is-ambulance-already-created? false
  set no-of-cars-met-with-accident 0
   ask patches
  [
    set intersection? false
    set pedestrian-lane0? false          ;; At the start of the
simulation and before
    set pedestrian-lane1? false          ;; creating pedestrian
lanes, no patch will
```

```
    set pedestrian-lane2? false          ;; come under pedestrian-
lanes group. Therefore
    set pedestrian-lane3? false          ;; making this field false
will ensure that no
    set pedestrian-lane4? false          ;; pedestrian patch is
there before creation of
    set pedestrian-lane5? false          ;; pedestrian lanes.
    set pedestrian-lane6? false          ;;
    set pedestrian-lane7? false          ;;

    set plane0? false                    ;;
    set plane1? false                    ;; At the start of the
simulation and before creating lanes
    set plane2? false                    ;; in the NetLogo world, no
patch will come under road lanes group.
    set plane3? false                    ;; Therefore making this
field false will ensure that no patch
    set plane4? false                    ;; belongs to road lane
before creation of roads
    set plane5? false                    ;;
    set plane6? false                    ;;
    set plane7? false                    ;;
  ]
  set tempcounter 0
  set tempcounter1 0
  draw-margins
  set-intersections
  setup-pedestrian-lanes
  setup-cars
  setup-pedestrians
  rash-driving-car
  set North-west "green"
  set counter 0
  change-green-light-NW

end

;;-------------------------------------Procedure is responsible for
giving the user a provision to create a
;;                                     text file and to store that
text file at his/her desired location.
;;                                     After opening this file,status
of all the turtles are entered--------------------------------;;

to create-text-file
  file-open user-new-file
end

;;-------------------------------------Procedure is responsible for
entering the report
;;                                       generated by all the turtles
after each tick. This procedure
;;                                       enters wait time, speed, who
is the front-vehicle regarding each
```

```
;;                                        turtle after every
tick-----------------------------------------------------------;;

to write-to-file
  file-print " "
  file-print ( word " Tick Number - " ticks )
  file-print ( word "
-------------------------------------------------------------" )
  ask cars
  [
    file-print ( word " Who - " who "---speed - " actual-speed "---
lane - " lane " ---wait-time - " wait-time " ----front car - "
front-car)
    file-print " "
  ]

  if ( counter >= 1 )
  [
    file-print( word " ***********************")
    file-print ( word " Average wait time of all cars is - " ((mean
[ wait-time] of cars) * 10) "mts" )
    file-print( word " ***********************")
    file-print (word " Average speed of all the cars is - " mean
[ actual-speed] of cars "mph")
    file-print (word "  ************************" )
    file-print (word " Density of cars on lane-0 - " count cars with
[lane = 0] )
    file-print (word " Density of cars on lane-1 - " count cars with
[lane = 1] )
    file-print (word " Density of cars on lane-2 - " count cars with
[lane = 2] )
    file-print (word " Density of cars on lane-3 - " count cars with
[lane = 3] )
    file-print (word " Density of cars on lane-4 - " count cars with
[lane = 4] )
    file-print (word " Density of cars on lane-5 - " count cars with
[lane = 5] )
    file-print (word " Density of cars on lane-6 - " count cars with
[lane = 6] )
    file-print (word " Density of cars on lane-7 - " count cars with
[lane = 7] )
  ]
end



to write-to-file-when-dying
  ask cars
  [
    file-print( word "The car - " who " died at lane - " lane )
  ]
end

;;------------------------------Procedure is responsible for
```

```
creation of intersection.
;;                              In the sense that, it specifies
which all patches comes
;;                              under the intersections. After
determining the patches, it
;;                              will set the intersection? property
to true.-----------------------------------------;;

to set-intersections
  set intersections patches with [ ( pxcor >= -2 and pxcor <= 2 and
pycor = 2 ) or
                                   ( pxcor = -2  and pycor >= -2 and
pycor <= 2 ) or
                                   ( pxcor = 2   and pycor >= -2 and
pycor <= 2 ) or
                                   (pxcor >= -2  and pxcor <= 2 and
pycor = -2 ) or
                                   (pxcor >= -2 and pxcor <= 2 and
pycor = 1 ) or
                                    (pxcor >= -2 and pxcor <= 2 and
pycor = 0 ) or
                                    (pxcor >= -2 and pxcor <= 2 and
pycor = -1 )
  ]

  ask intersections [ set intersection? true ]
end

;;-----------------------------Procedure is responsible for
creation of lanes along the
;;                              x-axis and y-axis. In total 8 lanes
are created along both
;;                              the axis and also determines which
patch comes under which lane.
;;                              If one patch come under lane 0 then
plane0? property of that patch
;;                              is made true and all remaining
false-------------------------------------------------;;

to draw-margins
  let x-cor max-pxcor mod 2
  let y-cor max-pycor mod 2

    ;---------to get white and red margins if the max-pxcor and max-
pycor are even numbers------------------------;;

    if ( max-pxcor mod 2 = 0) and (max-pycor mod 2 = 0 )
    [
      ask patches with [ (pxcor >= x-cor - 2)  and (pxcor <= x-cor +
2) ]
       [set pcolor white ]

    ;;;;------------------To get the green grass effect alongside
the roads
```

```
      ask patches with [ (pxcor < x-cor - 2 ) or ( pxcor > x-cor +
2 ) ]
      [set pcolor scale-color green ((random 1000) + 8000) 10
15000 ]


   ;;;;------------------To get White margins along the Y-axis
      ask patches with [ (pycor >= y-cor - 2) and (pycor <= y-cor +
2) ]
      [set pcolor white ]

    ]


   ;;---------to get white margins and red margins and if max-pxcor
and max-pycor are odd
numbers-----------------------------------------;;

   if ( max-pxcor mod 2 = 1 ) and ( max-pycor mod 2 = 1 )
   [
       ask patches with [ (pxcor >= x-cor + -3)  and (pxcor <= x-cor
+ 1) ]
       [set pcolor white ]

    ;;;;---------------------To get the green grass effect alongside
the roads
       ask patches with [ (pxcor < x-cor + -3 ) or ( pxcor >= x-cor
+ 2 ) ]
       [set pcolor scale-color green ((random 1000) + 8000) 10
15000 ]


   ;;;;-----------------To get White margins along the Y-axis
       ask patches with [ (pycor >= y-cor + -3) and (pycor <= y-cor
+ 1) ]
       [set pcolor white ]

    ]


     ;;;;---------------To get the effect of two lanes on a single
road
      ask patches with [ (pxcor mod 2 = 0) and (pycor = 0) and
(pxcor != 2) and (pxcor != -2) ]
       [ set pcolor black ]
      ask patches with [ (pycor mod 2 = 0) and (pxcor = 0) ]
        [set pcolor black ]

       ;;;;;;------------To delete the red patches and make them
white at the intersection of the two roads
        ask patches with [ (pxcor >= -2 ) and (pxcor <= 2 ) and
( pycor = 0) or
                                (pycor >= -2 ) and (pycor <= 2) and
(pxcor = 0) ]
```

```
         [set pcolor white ]

;;-----------------------------------------------------------------
-----------------------------------------------------------------
----;;
 ;;-----------------------------------To assign patches to their
respective
lanes-----------------------------------------------------------
-;;


 set glane0 patches with [ pxcor < -2 and (pycor = 2  or pycor =
1)]   ;;
 set glane1 patches with [ pxcor < -2 and (pycor = -2 or pycor =
-1)]  ;; assigning patches
 set glane2 patches with [ pxcor > 2  and (pycor = 2  or pycor =
1)]   ;; to different lanes
 set glane3 patches with [ pxcor > 2  and (pycor = -2 or pycor =
-1)]  ;; as per their
 set glane4 patches with [ (pxcor = 2  or pxcor = 1)  and pycor >
2]   ;; co-ordinates
 set glane5 patches with [ (pxcor = -2 or pxcor = -1) and pycor >
2]   ;;
 set glane6 patches with [ (pxcor = 2  or pxcor  = 1) and pycor <
-2]  ;;
 set glane7 patches with [ (pxcor = -2 or pxcor = -1) and pycor <
-2]  ;;



ask glane0 [ set plane0?
true]                                          ;;
ask glane1 [ set plane1?
true ]                                         ;; If a patch belongs
to lane A
ask glane2 [ set plane2?
true ]                                         ;; then planeA?
property of that patch
ask glane3 [ set plane3?
true ]                                         ;; is made true, so
that by seeing that
ask glane4 [ set plane4?
true ]                                         ;; particular field one
can know to which
ask glane5 [ set plane5?
true]                                          ;; lane that patch
belongs to
ask glane6 [ set plane6?
true ]                                         ;;
ask glane7 [ set plane7?
true]                                          ;;


ask patches with [ pxcor = 0 and pycor = 0 ] [ set pcolor black ]
```

```
end

;;-----------------------------------Procedure is responsible for
creating pedestrian-lanes.
;;                                    Creation of pedestrian-lanes
includes grouping patches depending
;;                                  upon the lane on which they
exist.-------------------------;;

to setup-pedestrian-lanes

  ask patches with [ ( pxcor >= 3 and pycor = 3 ) or
                     ( pxcor >= 3 and pycor = -3 ) or
                     ( pxcor = 3  and pycor >= 3 ) or
                     ( pxcor = 3  and pycor <= -3 ) or
                     ( pxcor = -3 and pycor <= -3 ) or
                     ( pxcor = -3 and pycor >= 3 ) or
                     ( pxcor <= -3 and pycor = 3 ) or
                     ( pxcor <= -3 and pycor = -3 ) ]
   [ set pcolor black ]

 set pedestrian-lane0 patches with [ pxcor <= -4 and pycor = 3 ]
 set pedestrian-lane1 patches with [ pxcor <= -4 and pycor = -3 ]
 set pedestrian-lane2 patches with [ pxcor >= 4 and pycor = 3 ]
 set pedestrian-lane3 patches with [ pxcor >= 4 and pycor = -3 ]
 set pedestrian-lane4 patches with [ pxcor = 3 and pycor >= 4 ]
 set pedestrian-lane5 patches with [ pxcor = -3 and pycor >= 4 ]
 set pedestrian-lane6 patches with [ pxcor = 3 and pycor <= -4 ]
 set pedestrian-lane7 patches with [ pxcor = -3 and pycor <= -4 ]

 ask pedestrian-lane0 [ set pedestrian-lane0? true ]
 ask pedestrian-lane1 [ set pedestrian-lane1? true ]
 ask pedestrian-lane2 [ set pedestrian-lane2? true ]
 ask pedestrian-lane3 [ set pedestrian-lane3? true ]
 ask pedestrian-lane4 [ set pedestrian-lane4? true ]
 ask pedestrian-lane5 [ set pedestrian-lane5? true ]
 ask pedestrian-lane6 [ set pedestrian-lane6? true ]
 ask pedestrian-lane7 [ set pedestrian-lane7? true ]


end

;;-----------------------------------Procedure is responsible for
the creation
;;                                    of pedestrians along the
pedestrian-lanes. This method
;;                                    will assign values to different
parameters of pedestrians
;;                                    and makes sure that they only
move in lanes-------------------------;;


to setup-pedestrians
```

```
let color-list [yellow pink cyan gray orange sky violet ]
    create-pedestrians 10
    [
      set size 1
      set shape "circle"
      set color one-of color-list
      set p-speed random 0.5
      set p-minspeed 0.2
      set p-maxspeed 0.5
      set p-deceleration 0.05
      set p-acceleration 0.0035

      ;set direction random 2
      set pedestrian-lane random 8
      set change-pedestrian-lane 0

      if ( pedestrian-lane = 0 )
      [
        set xcor random min-pxcor
        set ycor 3
        set heading 90

      ]

      if ( pedestrian-lane = 1 )
      [
        set xcor random min-pxcor
        set ycor -3
        set heading -90

      ]

      if ( pedestrian-lane = 2 )
      [
        set xcor random max-pxcor
        set ycor 3
        set heading 90
      ]

      if ( pedestrian-lane = 3 )
      [
        set xcor random max-pxcor
        set ycor -3
       set heading -90
      ]

      if ( pedestrian-lane = 4 )
      [
        set xcor 3
        set ycor random max-pycor
        set heading 180
      ]

      if ( pedestrian-lane = 5 )
```

```
        [
          set xcor -3
          set ycor random max-pycor
          set heading 0
        ]

        if ( pedestrian-lane = 6 )
        [
          set xcor 3
          set ycor random min-pycor
          set heading 180
        ]

        if ( pedestrian-lane = 7 )
        [
          set xcor -3
          set ycor random min-pycor
          set heading 0
        ]

        pedestrians-only-in-lanes
      ]
end

;;-----------------------------------Procedure is responsible for
creating pedestrian turtles
;;                                    only in lanes and not outside
of the lanes. It checks the
;;                                    co-ordinates of each and then
decides to do the needed operation--------------------;;

to pedestrians-only-in-lanes
  if ( pedestrian-lane = 0 or pedestrian-lane = 1 )
  [
    if ( xcor >= -3 )
    [
      let temp random min-pxcor
      ifelse ( temp >= -3 )
      [
        pedestrians-only-in-lanes
      ]
      [
        set xcor temp
      ]
    ]
  ]

  if ( pedestrian-lane = 2 or pedestrian-lane = 3 )
  [
     if ( xcor <= 3 )
     [
       let temp random max-pxcor
       ifelse ( temp <= 3 )
       [
```

```
        pedestrians-only-in-lanes
      ]
      [
        set xcor temp
      ]
    ]
  ]

  if ( pedestrian-lane = 4 or pedestrian-lane = 5 )
  [
    if ( ycor <= 3 )
    [
      let temp random max-pycor
      ifelse ( temp <= 3 )
      [
        pedestrians-only-in-lanes
      ]
      [
       set ycor temp
      ]
    ]
  ]

  if ( pedestrian-lane = 6 or pedestrian-lane = 7 )
  [
    if ( ycor >= -3 )
    [
      let temp random min-pycor
      ifelse ( temp >= -3 )
      [
        pedestrians-only-in-lanes
      ]
      [
        set ycor temp
      ]
    ]
  ]
end

;;----------------------------------Procedure is responsible for
movement of
;;                                  pedestrians along the lanes. It
also checks
;;                                  the speed of all pedestrian
turtles-------------;;

to forward-pedestrians

  if ( p-speed < p-minspeed ) [ set p-speed p-minspeed ]
  if ( p-speed > p-maxspeed ) [ set p-speed p-maxspeed ]
  fd p-speed
end

;;----------------------------------Procedure is responsible for
```

```
calling other
;;                                  methods which are responsible
for changing lanes.
;;                                 Depending upon the co-ordinates
on which a pedestrian
;;                               stand, it will decide whether to
change lane or not--------------------------;;


to move-pedestrians
  ask pedestrians
  [
    assign-different-lanes-to-pedestrians

    if ( pedestrian-lane = 0 or pedestrian-lane = 5)
    [
      ifelse ( [pxcor] of patch-here = -3 and [pycor] of patch-here
= 3 )
      [
        ifelse ( change-pedestrian-lane = 0 )
        [
          assign-different-lanes-during-direction-change-to-
pedestrians

        ]
        [
         set p-tempspeed p-speed
         set p-speed 0
        ]
       ]
       [ forward-pedestrians ]
    ]

    if ( pedestrian-lane = 1 or pedestrian-lane = 7 )
    [
      ifelse ( [pxcor] of patch-here = -3 and [pycor] of patch-here
= -3 )
      [
         ifelse ( change-pedestrian-lane = 0 )
        [
          assign-different-lanes-during-direction-change-to-
pedestrians

        ]
        [
         set p-tempspeed p-speed
         set p-speed 0
        ]
      ]

      [ forward-pedestrians ]
    ]

    if ( pedestrian-lane = 3 or pedestrian-lane = 6 )
```

```
    [
      ifelse ( [pxcor] of patch-here = 3 and [pycor] of patch-here =
-3 )
      [
        ifelse ( change-pedestrian-lane = 0 )
        [
          assign-different-lanes-during-direction-change-to-
pedestrians

        ]
        [
         set p-tempspeed p-speed
         set p-speed 0
        ]
       ]
      [ forward-pedestrians ]
    ]

    if ( pedestrian-lane = 4 or pedestrian-lane = 2 )
    [
      ifelse ( [pxcor] of patch-here = 3 and [pycor] of patch-here =
3 )
      [
        ifelse ( change-pedestrian-lane = 0 )
        [
          assign-different-lanes-during-direction-change-to-
pedestrians

        ]
        [
         set p-tempspeed p-speed
         set p-speed 0
        ]
      ]
      [ forward-pedestrians ]
    ]

  ]
end

;;-------------------------------------------------Procedure is
responsible for changing of lanes.
;;                                                  Depending upon the
random value assigned to change-pedestrian-lane,
;;                                                  this method makes
turtle to change its direction and also its lane----------;;

to assign-different-lanes-during-direction-change-to-pedestrians


    if ( change-pedestrian-lane = 0 )
        [
          if ( pedestrian-lane = 0 )
          [
```

```
    set xcor -3
    set ycor 4
    set heading 0
    forward-pedestrians
   ]

  if ( pedestrian-lane = 1 )
  [
    set xcor -3
    set ycor -4
    set heading 180
    forward-pedestrians
   ]


  if ( pedestrian-lane = 7 )
  [
    set xcor -4
    set ycor -3
    set heading -90
    forward-pedestrians
   ]


  if ( pedestrian-lane = 4)
  [
    set xcor 4
    set ycor 3
    set heading 90
    forward-pedestrians
   ]

 if ( pedestrian-lane = 3 )
  [
    set xcor 3
    set ycor -4
    set heading 180
    forward-pedestrians
   ]


  if ( pedestrian-lane = 2 )
  [
    set xcor 3
    set ycor 4
    set heading 0
    forward-pedestrians
   ]

  if ( pedestrian-lane = 6 and heading = 0)
  [
    set xcor 4
    set ycor -3
    set heading 90
```

```
            forward-pedestrians
           ]

        ]

end

;;-----------------------------------------Procedure is
responsible for changing lanes of pedestrians.
;;                                          For example, if a
pedestrian is of lane 2 when created, and while
;;                                          moving if it is on Lane
4, the pedestrian-lane property of the turtle
;;                                          is changed from 2 to
4-----------------------------------------------------;;



to assign-different-lanes-to-pedestrians

    if ( [pedestrian-lane2?] of patch-here )
    [
      set pedestrian-lane 2
    ]

    if ( [pedestrian-lane3?] of patch-here )
    [
      set pedestrian-lane 3
    ]

    if ( [pedestrian-lane6?] of patch-here )
    [
      set pedestrian-lane 6
    ]

    if ( [pedestrian-lane7?] of patch-here )
    [
      set pedestrian-lane 7
    ]

    if ( [pedestrian-lane0?] of patch-here )
    [
      set pedestrian-lane 0
    ]

    if ( [pedestrian-lane1?] of patch-here )
    [
      set pedestrian-lane 1
    ]

    if ( [pedestrian-lane4?] of patch-here )
    [
      set pedestrian-lane 4
    ]
```

```
    if ( [pedestrian-lane5?] of patch-here )
    [
      set pedestrian-lane 5
    ]

end

;;-------------------------------------------Procedure is
responsible for calling
;;                                            different methods
which will instantiate and
;;                                          initiate different car
properties-------------------------------------------------;;

to setup-cars

 ifelse ( number-of-cars <= floor ( (((max-pxcor - 5 ) * 30)  /
100 ) * 8) )  ;; Checks whether the no of cars creating are less
than the size of all the road lanes together.
 [
      create-cars number-of-cars
      [
        car-parameters
        separate-cars ;;to prevent more than one turtle on the same
patch
        cars-only-in-lanes ;; cars should not be present beyond the
traffic lights when the simulation starts
      ]
 ]
 [
    user-message(word " Reduce the number-of-cars and then try
again")
  ]
end

;;-------------------------------------procedure is responsible for
creating
;;                                       new cars when the number of
cars created
;;                                        are less when compared to
the start of the simulation and
;;                                        call other methods to
initiate car parameters--------------------;;

to count-cars
let no_of_cars count cars

if ( no_of_cars < number-of-cars )
[
  create-cars ( number-of-cars - no_of_cars )
  [
    car-parameters
    separate-cars
```

```
    cars-only-in-lanes
    ]
    ]
    end


to car-parameters
set rash-car? false
set dying? false
set size 1
     set color blue
     set speed 0.2 + random-float 1.1
     set max-speed 1.2
     set min-speed 0.3
     set deceleration 0.05
     set acceleration 0.0035
      set lane (random 8)

;;------------------------------Assigning co-ordinates to cars
(turtles) depending upon the lanes they are assigned to----------;;

      if ( lane = 0 )
       [
         set xcor random min-pxcor
         set ycor 2
       ]

      if ( lane = 1)
       [
         set xcor random min-pxcor
         set ycor -2
        ]

       if ( lane = 2 )
       [
         set xcor random max-pxcor
         set ycor 2
       ]

       if ( lane = 3 )
       [
         set xcor random max-pxcor
         set ycor -2
        ]

       if ( lane = 4 )
       [
          set xcor 2
          set ycor random max-pycor
       ]

       if ( lane = 5 )
        [
          set xcor -2
```

```
          set ycor random max-pycor
        ]

        if ( lane = 6 )
        [
          set xcor 2
          set ycor random min-pycor
        ]

       if ( lane = 7 )
        [
          set xcor -2
          set ycor random min-pycor
        ]

        ;;---------------------------------if not on correct
lanes-------------------------------------------------------------;;

          if ( lane = 0 ) or ( lane = 2 )
         [
           if ( ycor > 2 )
            [

              die
            ]
         ]

         if ( lane = 1 ) or ( lane = 3 )
         [
           if ( ycor < -2 )
            [

              die
            ]
         ]

         if ( lane = 4 ) or ( lane = 6 )
         [
           if ( xcor > 2 )
            [

              die
            ]
         ]

         if ( lane = 5 ) or ( lane = 7 )
          [
             if ( xcor < -2 )
              [

                die
              ]
          ]
  ;;-----------------------------------------Setting headings to
```

```
the cars
;;-------------------------------------------------------------------
------------------;;


       if ( lane = 0 ) or ( lane = 2 )
        [ set heading 90 ]

       if ( lane = 3)
       [ set heading 270 ]

       if (lane = 1)
       [ set heading 270 ]

       ;if ( lane = 3 ) or ( lane = 1 )
        ;[ set heading 270 ]

       if ( lane = 5 ) or ( lane = 7 )

        [ set heading 0 ]


       if ( lane = 6 ) or ( lane = 4 )
        [ set heading 180 ]


        assign-actual-speed-to-cars

end

;;---------------------------------------correlates the NetLogo
speed of the car
;;                                        to the actual speed of the
car in the real world-----------------------;;

to assign-actual-speed-to-cars

  if ( speed >= 0.1 and speed <= 0.5 ) [ set actual-speed 20 ]
  if ( speed > 0.5 and speed <= 0.8 )   [ set actual-speed 45 ]
  if ( speed > 0.8 and speed <= 1.3 )   [ set actual-speed 70 ]
  set actual-acceleration 4
  set actual-deceleration 8
end

;;---------------------------------------When turtles are created,
more than one turtle
;;                                       can be placed on one patch.
Following method will check
;;                                       and assign turtles on
different patches-----------------------------------;;


to separate-cars
```

```
 if any? other cars-here
 [
   ifelse ( [pcolor] of patch-ahead 1 != red )
    [
     fd 1
     separate-cars
    ]
    [
      set dying? true
      write-to-file-when-dying
      set no-of-cars-met-with-accident no-of-cars-met-with-accident
+ 1
       die
     ]
  ]
end

;;-----------------------------------------------------------------
-----------------------------------------------------------------
--------------------------------------;;

;;-------------------------------Procedure to prevent cars beyond
traffic lights when simulation
starts-----------------------------------------------------------
----------;;

to cars-only-in-lanes

if ( lane = 0 ) or ( lane = 1 )
[
 if ( xcor >= -3 )
 [
  let temp-xcor random min-pxcor
  ifelse ( temp-xcor < -2 )
  [ set xcor temp-xcor ]
  [ cars-only-in-lanes ]
 ]
]

if ( lane = 2 ) or ( lane = 3 )
[
 if ( xcor <= 3 )
 [
  let temp-xcor random max-pxcor
  ifelse ( temp-xcor > 2 )
  [ set xcor temp-xcor ]
  [ cars-only-in-lanes ]
 ]
]

if ( lane = 4 ) or ( lane = 5 )
[
 if ( ycor <= 3 )
 [
```

```
  let temp-ycor random max-pycor
  ifelse ( temp-ycor > 2 )
  [ set ycor temp-ycor ]
  [ cars-only-in-lanes ]
 ]
]

if ( lane = 6 ) or ( lane = 7 )
[
 if ( ycor >= -3 )
 [
  let temp-ycor random min-pycor
  ifelse ( temp-ycor < -2 )
  [ set ycor temp-ycor ]
  [ cars-only-in-lanes ]
 ]
]

end

;;-----------------------------------------Procedure selects one-of
the cars
;;                                          as rash car and call
other methods which
;;                                          formulates rash
behaviour-----------------------------------------------------------;;

to rash-driving-car

  ask one-of cars [set rash-car? true ]
  let rash-car one-of cars with [rash-car? = true ]
  ask cars with [ rash-car? = true ]
  [
    set color red
  ]

end



;;------------------------------ To let cars move along the lanes
-------------------------------------------------------------------
--------------------------------------;;

to go

;;----------------------------if snow is off, all snow turtles will
die-----------------------;;

if (not snow )
[
 if( snow-counter = 1 )
 [
  ask water [ die ]
```

```
 ]
]


if ( ticks = number-of-ticks )
[
 ask turtles [ die ]
 clear-all
 stop
]
cars-new-behaviour
rash-driver-behaviour

;;------------------------Periodically calculates the number of cars
in the NetLogo world-----------------------------;;

if ( (counter mod 4000) = 0 )
[
 count-cars
]

periodic-change-of-lanes-in-vehicles
move
move-pedestrians
set counter counter + 1

;;------------------------Responsible for chanmging of lights from
red to orange-------------------------------------;;

if ( counter mod Traffic-Lights-timer? = (Traffic-Lights-timer? -
4 ))
 [
   if ( North-west = "green" )
   [ ask patches with [ pxcor = 2 and pycor = 3 ] [ set pcolor
orange ] ]

   if ( North-east = "green" )
   [ ask patches with [ pxcor = 3 and pycor = -2 ] [ set pcolor
orange ] ]

   if ( South-east = "green" )
   [ ask patches with [ pxcor = -2 and pycor = -3 ] [ set pcolor
orange ] ]

   if ( South-west = "green" )
   [ ask patches with [ pxcor = -3 and pycor = 2 ] [ set pcolor
orange ] ]
 ]

;;------------------------Responsible for calling traffic lights
change method---------------------;;

if counter mod Traffic-Lights-timer? = 0
[
```

```
 change-globals-red
]

tick
write-to-file
end

;;-------------------------------------------------------------------
-------------------------------------------------------------------
---------;;

to accelerate-the-car
   set speed speed + acceleration
end

to decelerate-the-car
  set speed [ speed ] of front-car - deceleration
end

;;--------------------------------This procedure call various
methods
;;                                  which are responsible for the
movement of cars
;;                                  depending upon the traffic light
ahead of them-----------------------------------;;

to move

    if ( North-west = "green" )
    [

      ask cars
      [
        if [intersection?] of patch-here or [plane5?] of patch-here
        [
          if ( lane = 7 )
          [
            let tempspeed speed
            set lane  5
            set speed 1.2
            forward-cars-as-per-speed
          ]
        ]

      ]
    ]

if ( North-west = "green"  )
[
 ask cars
 [
  if ( lane = 0 or lane = 2 or lane = 5 or lane = 6 or lane = 1)
  [
  set wait-time 0
```

```
   move-cars
  ]
 if ( lane = 4 or lane = 7 or lane = 3 )
  [
   set wait-time wait-time + 1
   separate-cars2
  ]
 ]
]

;;-----------------------------------------------------------------
---;;

    if (North-east = "green" )
    [
       ask cars
      [
        if [intersection?] of patch-here or [plane2?] of patch-here
        [
          if ( lane = 0 )
          [
            let tempspeed speed
            set lane  2
            set speed 1.2
           forward-cars-as-per-speed
          ]
        ]
      ]
    ]


if ( North-east = "green" and North-east != orange )
[
 ask cars
 [
  if ( lane = 4 or lane = 6 or lane = 2 or lane = 5 or lane = 1)
  [
   set wait-time 0
   move-cars
  ]
  if ( lane = 0 or lane = 7 or lane = 3 )
  [
    set wait-time wait-time + 1
    separate-cars2
  ]
 ]
]

;;-----------------------------------------------------------------
-------;;

    if ( South-east = "green")
    [
       ask cars
```

```
      [
        if [intersection?] of patch-here or [plane6?] of patch-here
        [
          if ( lane = 4 )
          [
            let tempspeed speed
            set lane  6
            set speed max-speed
            forward-cars-as-per-speed

          ]
        ]
      ]
    ]

if ( South-east = "green" and South-east != orange)
[
 ask cars
 [
  if ( lane = 3 or lane = 1 or lane = 2 or lane = 5 or lane = 6)
  [
   set wait-time 0
   move-cars
  ]
  if ( lane = 0 or lane = 4 or lane = 7 )
  [
  set wait-time wait-time + 1
   separate-cars2
  ]
 ]
]

;;-------------------------------------------------------------------
--------;;

     if ( South-west = "green")
     [

      ask cars
      [
        if [intersection?] of patch-here or [plane1?] of patch-here
        [
          if ( lane = 3 )
          [
            let tempspeed speed
            set lane  1
            set speed max-speed
            forward-cars-as-per-speed
          ]
        ]
      ]
     ]

if ( South-west = "green" )
```

```
[
 ask cars
 [
  if ( lane = 7 or lane = 5 or lane = 2 or lane = 6 or lane = 1)
  [
  set wait-time 0
   move-cars
  ]
  if ( lane = 0 or lane = 3 or lane = 4 )
  [
   set wait-time wait-time + 1
   separate-cars2
  ]
 ]
]


end

;;-----------------------------------------Procedure responsible
for changing lane
;;                                          property of cars when
they move from one lane to another
;;                                          and also call other
procedures responsible for movement of
cars-------------------------;;

to move-cars

  if ( lane = 2 )
  [
    if ([plane0?] of patch-here )
    [
      set lane 0
    ]
  ]
  if ( lane = 0 )
  [
    if ([plane2?] of patch-here )
    [
      set lane 2
    ]
  ]

  if ( lane = 4 )
  [
    if ( [plane6?] of patch-here )
    [
      set lane 6
    ]
  ]
  if ( lane = 6 )
  [
    if ([plane4?] of patch-here)
```

```
    [
      set lane 4
    ]
  ]

  if ( lane = 7 )
  [
    if ([plane5?] of patch-here)
    [
      set lane 5
    ]
  ]
  if ( lane = 5 )
  [
    if ([plane7?] of patch-here)
    [
      set lane 7
    ]
  ]

  if ( lane = 3 )
  [
    if ([plane1?] of patch-here)
    [
      set lane 1
    ]
  ]
  if ( lane = 1 )
  [
    if ( [plane3?] of patch-here)
    [
      set lane 3
    ]
  ]


    forward-cars-as-per-speed
  end

;;-------------------------Procedure responsible for forwarding cars
;;                         as per their speeds, checks speeds and
call other procedures
;;                         responsible for changing of
lanes-------------------------------------------------------;;

to forward-cars-as-per-speed

           set front-car one-of cars-on patch-ahead 1
           ifelse ( front-car = nobody )
           [ accelerate-the-car ]
           [ decelerate-the-car ]


          if  ( speed < min-speed ) [ set speed min-speed + 1 ]
```

```
          if  ( speed > max-speed ) [ set speed max-speed  ]

         fd speed
         changing-lanes

end

;;------------------------Changing of traffic lights as time
progresses-----------------------------------------------;;

to change-globals-red

            if ( North-west = "green" )
             [
               set North-west "red"
               set North-east "green"
               change-green-light-NE
               stop
             ]
              if ( North-east = "green" )
             [
               set South-east "green"
               set North-east "red"
               change-green-light-SE
               stop
             ]

            if ( South-east = "green" )
             [
               set South-west "green"
               set South-east "red"
               change-green-light-SW
               stop
             ]
              if ( South-west = "green" )
             [
               set North-west "green"
               set South-west "red"
               change-green-light-NW
               stop
             ]
end

;;---------------------------------Changing of traffic lights a
time
progresses-----------------------------------------------------
---------;;

to change-green-light-NW

      if ( North-west = "green" )
       [
         ask patch -3 2   [ set pcolor green ]
         ask patch  2 3   [ set pcolor red ]
```

```
          ask patch  3 -2  [ set pcolor red ]
          ask patch -2 -3  [set pcolor red ]
        ]

end

to change-green-light-NE

        if ( North-east = "green" )
        [
          ask patch -2 2   [ set pcolor white ]
          ask patch  3 2   [ set pcolor white ]
          ask patch 2 3    [ set pcolor green ]
          ask patch  2 -3  [ set pcolor white ]
          ask patch -2 -3  [set pcolor red ]
          ask patch -3 -2   [ set pcolor white ]
          ask patch 3 -2    [ set pcolor red ]
          ask patch -3 2    [set pcolor red ]
        ]

end

to change-green-light-SE
        if ( South-east = "green" )
        [
          ask patch -3 2   [ set pcolor red ]
          ask patch -3 -2   [ set pcolor white ]
          ask patch 3 2    [ set pcolor white ]
          ask patch  2 3   [ set pcolor red ]
          ask patch  3 -2  [ set pcolor green ]
          ask patch -2 -3  [ set pcolor red ]
          ask patch 2 -3   [ set pcolor white ]

        ]

end

to change-green-light-SW
        if ( South-west = "green" )
        [
          ask patch -3 2   [ set pcolor red ]
          ask patch  2 3   [ set pcolor red ]
          ;ask patch  2 -2  [ set pcolor red ]
          ask patch -2 -3  [set pcolor  green ]
          ask patch 3 2    [ set pcolor white ]
          ask patch -3 2   [ set pcolor red ]
          ask patch 3 -2   [ set pcolor red ]
        ]

end

;;-----------------------------------------Procedure responsible
for modifying various parameters
;;                                        of cars depending upon the
```

```
lane from which they are taking a turn
;;                                        into which lane ( new
lane )----------------------------;;

to changing-lanes

  if ( lane = 0 )
  [
    if (change-lane = 1 )
    [
        if ( pxcor = 1 and pycor = 1 )
        [
          set heading 180
          set lane 6
          assign-correct-coordinates
        ]
    ]
   if ( change-lane = 2 )
    [
      if ( pxcor = -2 and pycor = 2 )
      [
        set heading 0
        set lane 5
        assign-correct-coordinates
      ]
    ]
   ]

  ;;-------------------------------------------------------------------
------------;;

  if ( lane = 3 )
  [
    if ( change-lane = 2 )
    [
     if ( pxcor = 2 and pycor = -2 )
      [
       set heading 180
        set lane 6
        assign-correct-coordinates
      ]
    ]
      if ( change-lane = 1 )
      [
       if ( pxcor = -1 and pycor = -1 )
        [
          set heading 0
          set lane 5
          assign-correct-coordinates
        ]
      ]
    ]

  ;;-------------------------------------------------------------------
```

```
------------;;

     if ( lane = 4 )
    [
    if ( change-lane = 1 )
    [
     if ( pxcor = 1 and pycor = -1 )
    [
        set heading -90
        set lane 1
        assign-correct-coordinates
      ]
    ]

      if ( change-lane = 2 )
      [
        if ( pxcor = 2 and pycor = 2 )
        [
          set heading 90
          set lane 2
          assign-correct-coordinates
        ]
      ]
    ]

;;-----------------------------------------------------------------
----------;;

   if ( lane = 7 )
   [
    if ( change-lane = 2 )
    [
     if ( pxcor = -2 and pycor = -2 )
      [
        set heading -90
        set lane 1
        assign-correct-coordinates
      ]
    ]
      if ( change-lane = 1 )
      [
        if ( pxcor = -1 and pycor = 1 )
        [
          set heading 90
          set lane 2
          assign-correct-coordinates
        ]
      ]
    ]

end

to assign-correct-coordinates
```

```
if ( lane = 1)
[
  if ( change-lane != 1 )
  [
    if ( ycor < -2 or (ycor < 0 and ycor > -2 ) )
    [
      set ycor -2
    ]
  ]
  if ( change-lane = 1)
  [
    if ( ycor != -1 )
      [
       set ycor -1
      ]
  ]
]
 if (lane = 2)
 [
   if ( change-lane != 1)
   [
       if ( ycor > 2 or ( ycor > 0 and ycor < 2 ))
       [
          set ycor 2
        ]
   ]
   if ( change-lane = 1 )
   [
     if ( ycor != 1)
     [
       set ycor 1
     ]
   ]
 ]

if (lane = 6)
[
  if ( change-lane != 1)
  [
      if ( xcor > 2 or (xcor > 0 and xcor < 2))
      [
        set xcor 2
      ]
  ]
  if ( change-lane = 1 )
  [
   if ( xcor != 1)
   [
     set xcor 1
   ]
  ]
]

if (lane = 5)
```

```
  [
    if ( change-lane != 1)
    [
        if ( xcor < -2  or ( xcor < 0 and xcor > -2))
        [
          set xcor -2
        ]
    ]
    if ( change-lane = 1 )
    [
      if ( xcor != -1 )
      [
        set xcor -1
      ]
    ]
  ]

end

;;-----------------------------------------------Procedure
responsible for stopping cars
;;                                                cars infront of red
light. When a car is stopped
;;                                            at a red signal, all the
other cars stops right behind the stopped vehicle---------------;;

to separate-cars2


  if ( lane = 0)
  [
    if ( xcor >= min-pxcor and ( ycor > 0 and ycor <= 2 ))
    [
      separate-cars
      ifelse ( change-lane = 1 ) [ temp-procedure ]
      [stop-at-the-red-light]
    ]
  ]

  if ( lane = 4 )
  [
    if ( ycor <= max-pycor and (xcor > 0 and xcor <= 2) )
    [
      separate-cars
       ifelse ( change-lane = 1 ) [ temp-procedure ]
      [stop-at-the-red-light]
    ]
  ]

  if ( lane = 3 )
  [
    if ( xcor <= max-pxcor and (ycor >= -2 and ycor < 0))
    [
      separate-cars
```

```
       ifelse ( change-lane = 1 ) [ temp-procedure ]
      [stop-at-the-red-light]
    ]
  ]

  if ( lane = 7 )
  [
    if( (xcor < 0 and xcor >= -2) and ycor >= min-pycor)
    [
      separate-cars
       ifelse ( change-lane = 1 ) [ temp-procedure ]
      [stop-at-the-red-light]
    ]
  ]

end

;;-------------------------Procedure responsible for stopping
vehicles in front of red light----------------------------;;

to stop-at-the-red-light
  let t-ahead one-of cars-on patch-ahead 1
  if ( t-ahead = nobody and [pcolor] of patch-ahead 1 != red and
[pcolor] of patch-ahead 1 != orange )
  [
    fd 1
    slow-cars-at-the-red-light
    if ( (tempcounter mod 5) = 0 )
    [
      stop-at-the-red-light
    ]
  ]

end


to temp-procedure

  let t-ahead one-of cars-on patch-ahead 1
  if ( t-ahead = nobody )
  [
    if (lane = 0 )
    [
      if ( ([pcolor] of patches with [pxcor = -3 and pycor = 2] !=
[15] and
        [pcolor] of patches with [pxcor = -3 and pycor = 2] != [25])
or
        [pxcor] of patch-ahead 1 != -3  )
      [
       fd 1
       slow-cars-at-the-red-light1
       if ( (tempcounter1 mod 5) = 0 )
       [
         temp-procedure
```

```
       ]
      ]
    ]

     if (lane = 4 )
    [
      if ( ([pcolor] of patches with [ pxcor = 2 and pycor = 3] !=
[15] and
        [pcolor] of patches with [ pxcor = 2 and pycor = 3] !=
[25] )or
        [pycor] of patch-ahead 1 != 3 )
       [
        fd 1
        slow-cars-at-the-red-light1
        if ( (tempcounter1 mod 5) = 0 )
        [
          temp-procedure
        ]
       ]
    ]

     if (lane = 3 )
    [
      if ( ([pcolor] of patches with [pxcor = 3 and pycor = -2] !=
[15] and
        [pcolor] of patches with [pxcor = 3 and pycor = -2] != [25])
or
        [pxcor] of patch-ahead 1 != 3 )
       [
        fd 1
        slow-cars-at-the-red-light1
        if ( (tempcounter1 mod 5) = 0 )
        [
          temp-procedure
        ]
       ]
    ]

     if (lane = 7 )
    [
      if ( ([pcolor] of patches with [pxcor = -2 and pycor = -3] !=
[15] and
        [pcolor] of patches with [pxcor = -2 and pycor = -3] !=
[25]) or
        [pycor] of patch-ahead 1 != -3 )
       [
        fd 1
        slow-cars-at-the-red-light1
        if ( (tempcounter1 mod 5) = 0 )
        [
          temp-procedure
        ]
       ]
    ]
```

```
  ]
end



to slow-cars-at-the-red-light
  set tempcounter tempcounter + 1
end

to slow-cars-at-the-red-light1
   set tempcounter1 tempcounter1 + 1
end

;;---------------------------------------Periodically changes the
change-lane property of vehicles into lanes
;;                                      which is responsible for their
turning into a particular lane-----------------------------;;

to periodic-change-of-lanes-in-vehicles
   if ( ( ticks mod 300) = 0 )
   [
     ask cars
     [
       set change-lane random 3
     ]
   ]
end

;;-------------------------------------Procedure responsible for
simulation of emergency vehicles and also
;;                                                  initializes
their parameters-----------------------------;;
to Emergency-Vehicles

 ifelse (not Is-ambulance-already-created? )
 [
  set ecounter 0
  set emergency-dead 0
  set Is-ambulance-already-created? true
   create-ambulances 1
   [
     set size 2
     set decision-speed 1.2
     ;set lane random 8
     if ( North-west = "green" ) [ set lane 0 ]
     if ( North-east = "green")  [ set lane 4 ]
     if ( South-west = "green" ) [ set lane 7 ]
     if ( South-east = "green" ) [ set lane 3 ]
     set shape "arrow"
     set change-lane random 3
     set speed 1.2
     set color blue
```

```
     if ( lane = 0 )
       [
         setxy min-pxcor 1
         set heading 90
       ]

       if ( lane = 3 )
       [
         setxy max-pxcor -1
         set heading -90
        ]

       if ( lane = 4 )
       [
          setxy 1 max-pycor
          set heading 180
       ]

       if ( lane = 7 )
        [
          setxy -1 min-pycor
          set heading 0
        ]


        assign-actual-speed-to-ambulances
        set emergency-lane lane
        stop-vehicles-during-emergency

   ]
 ]
 [
   user-message ( word " ambulance already created")
 ]

end

;;---------------------Repeatedly calls other functions which are
responsible for movement of emergency vehicles----------;;

to go-emergency

 if ( ticks = number-of-ticks)
 [
  stop
]

ask ambulances
[
 set hidden? false
 ]
  Emergency-vehicles-changing-lanes
forward-the-emergency-vehicle
```

```
end

;;-----------------------Procedure responsible for changing lanes by
emergency vehicles----------------;;

to Emergency-vehicles-changing-lanes

    ask ambulances
    [
    if ( lane = 0 )
    [
   if ( change-lane = 0 or change-lane = 3 )
   [
     if ( pxcor = -3 and pycor = 1 )
     [
     set ecounter 1
     set lane 2

     ]
   ]
    if (change-lane = 1 )
    [
        if ( pxcor = 1 and pycor = 1 )
        [

          set heading 180
          set lane 6
          set ecounter 1
        ]
    ]
    if ( change-lane = 2 )
    [
      if ( pxcor = -1 and pycor = 1 )
      [
        set heading 0
        set lane 5
        set ecounter 1
      ]
    ]
   ]
  ;;-------------------------------------------------------------------
------------;;
  if ( lane = 3 )
  [
     if ( change-lane = 0 or change-lane = 3 )
   [
     if ( pxcor = 3 and pycor = -1 )
     [
     set ecounter 1
     set lane 1

     ]
   ]
```

```
    if ( change-lane = 2 )
    [
     if ( pxcor = 1 and pycor = -1 )
      [
        set heading 180
        set lane 6
        set ecounter 1
      ]
    ]
      if ( change-lane = 1 )
      [
       if ( pxcor = -2 and pycor = -1 )
        [
          set heading 0
          set lane 5
          set ecounter 1
        ]
      ]
    ]

  ;;-------------------------------------------------------------------
------------;;

     if ( lane = 4 )
  [
    if ( change-lane = 0 or change-lane = 3 )
   [
     if ( pxcor = 1 and pycor = -2 )
     [
     set ecounter 1
     set lane 6

     ]
   ]

    if ( change-lane = 1 )
    [
     if ( pxcor = 1 and pycor = -2 )
      [
        set heading -90
        set lane 1
        set ecounter 1
      ]
    ]
      if ( change-lane = 2 )
      [
        if ( pxcor = 1 and pycor = 1 )
        [
          set heading 90
          set lane 2
          set ecounter 1
        ]
      ]
    ]
```

```
;;-----------------------------------------------------------------
----------;;
   if ( lane = 7 )
  [
   if ( change-lane = 0 or change-lane = 3 )
   [
     if ( pxcor = -1 and pycor = -3 )
     [
     set ecounter 1
     set lane 5

     ]
   ]
    if ( change-lane = 2 )
    [
     if ( pxcor = -1 and pycor = -1 )
      [
        set heading -90
        set lane 1
        set ecounter 1

      ]
    ]
      if ( change-lane = 1 )
      [
        if ( pxcor = -1 and pycor = 2 )
        [
          set heading 90
          set lane 2
          set ecounter 1
        ]
      ]
    ]
    ]

end

;;--------------------------------------Procedure correlates the
parameters of emergency vehicles
;;                                        in the NetLogo world to the
values of the real world------------------------;;

to assign-actual-speed-to-ambulances

   if ( speed >= 0.1 and speed <= 0.5 ) [ set ambulance-actual-speed
20 ]
  if ( speed > 0.5 and speed <= 0.8 )   [ set ambulance-actual-speed
45 ]
  if ( speed > 0.8 and speed <= 1.3 )   [ set ambulance-actual-speed
70 ]
  set ambulance-actual-acceleration 4
  set ambulance-actual-deceleration 8
end
```

```
;;--------------------------Accelerates and Decelerates the
emergency vehicles-----------------------------;;

to accelerate-the-ambulance
  set speed speed + acceleration
  end


to decelerate-the-ambulance
  set speed [speed] of front-vehicle - deceleration
 end

;;-----------------------------------------Procedure responsible
for the forward movement of cars
;;                                          and also call other
procedures responsible for behaviour of emergency vehicles--------;;

 to forward-the-emergency-vehicle

   ask ambulances
   [
     if ( ecounter = 1 )
     [
       if ( xcor >= max-pxcor or xcor <= min-pxcor or ycor >= max-
pycor or ycor <= min-pycor )
       [
         set emergency-dead 1
         set Is-ambulance-already-created? false
         die
       ]
     ]
     if (( xcor < max-pxcor or xcor > min-pxcor or ycor < max-pycor
or ycor > min-pycor ))
       [
          set front-vehicle one-of cars-on patch-ahead 1
           ifelse ( front-vehicle = nobody )
           [ accelerate-the-ambulance ]
           [ decelerate-the-ambulance ]
           fd speed

           ambulance-decision-during-red-light
           test-sound
       ]
     ]
   if ( emergency-dead = 1 )
   [
    resume-all-the-vehicles-after-emergency-vehicle-moved-out
   ]
 end

;;--------------------------Procedure responsible for sound evolving
from emergency vehicles-------------------------;;
```

```
to  test-sound
  ;sound:play-sound "/Users/jay/Downloads/Ambulance.wav"
end

;;---------------------------------- Changes lane property of
emergency vehicles
;;                                    when it moved from one lane to
another different lane altogether-----------------;;

to change-lanes-for-emergency-vehicles

   if ( [intersection?] of patch-here )
  [
  if ( lane = 0  )
  [
    if ( change-lane = 2 ) [ set lane 5]
    if ( change-lane = 0 or change-lane = 3 ) [ set lane 2 ]
    if ( change-lane = 1 ) [ set lane 6 ]
  ]

  if ( lane = 4  )
  [
   if ( change-lane = 0 or change-lane = 3 ) [ set lane 6 ]
   if ( change-lane = 2 ) [ set lane 2 ]
   if ( change-lane = 1 ) [ set lane 1 ]
  ]

  if ( lane = 3 )
  [
   if ( change-lane = 0 or change-lane = 3 ) [ set lane 1 ]
   if ( change-lane = 2 ) [ set lane 6 ]
   if ( change-lane = 1 ) [ set lane 5 ]
  ]

  if ( lane = 7 )
  [
   if ( change-lane = 0 or change-lane = 3 ) [ set lane 5 ]
   if ( change-lane = 2 ) [ set lane 1 ]
   if ( change-lane = 1 ) [ set lane 2 ]
  ]
  ]

end

;;-----------------------------------Depending upon the lane into
which an emergency vehicle should turn, its behaviour at the red
light will differ. For example,
;;                if emergency vehicle is on lane 0 and it is red
light and if the vehicle should have to turn into lane 5, it will
turn even when
;;                it is red light. But if it has to turn into lane 2
or lane 6 it will stop at red light, as there will be other vehicles
moving along
;;                those
```

```
lanes-------------------------------------------------------------
------------;;

to ambulance-decision-during-red-light

  if ( lane = 0 )
  [
     if ( [pcolor] of patches with [ pxcor = -3 and pycor = 2 ] =
[15] and ( change-lane != 2 )
                 and [pxcor] of patch-ahead 1 = -3)
       [
         set speed 0
       ]
       if ( [pcolor] of patches with [ pxcor = -3 and pycor = 2 ] =
[55]
                 and ([pxcor] of patch-here = -4 or [pxcor] of patch-
here > -3 )and speed = 0)
       [
         set  speed decision-speed
       ]
   ]

  if ( lane = 4 )
  [
    if ( [pcolor] of patches with [ pxcor = 2 and pycor = 3] =
[ 15 ] and ( change-lane != 2 )
          and [pycor] of patch-ahead 1 = 3 )
      [
        set speed 0
      ]
      if ( [pcolor] of patches with [ pxcor = 2 and pycor = 3] =
[ 55 ]
          and ([pycor] of patch-here = 4 or [pycor] of patch-here <
3 ) and speed = 0 )
      [
        set  speed decision-speed
      ]
  ]
  if ( lane = 3 )
  [
    if ( [pcolor] of patches with [ pxcor = 3 and pycor = -2 ] =
[15] and ( change-lane != 2 )
          and [pxcor] of patch-ahead 1 = 3)
      [
        set speed 0
      ]

      if ( [pcolor] of patches with [ pxcor = 3 and pycor = -2 ] =
[55]
        and ([pxcor] of patch-here = 4 or [pxcor] of patch-here <
3 )and speed = 0)
      [
        set  speed decision-speed
      ]
```

```
  ]

  if ( lane = 7 )
  [
    if ( [pcolor] of patches with [ pxcor = -2 and pycor = -3 ] =
[15] and (change-lane != 2 )
       and [pycor] of patch-ahead 1 = -3)
    [
      set speed 0
    ]

    if ( [pcolor] of patches with [ pxcor = -2 and pycor = -3 ] =
[55]
       and ([pycor] of patch-here = -4 or [pycor] of patch-here >
-3 ) and speed = 0)
    [
      set  speed decision-speed
    ]

  ]

end

;;----------------------Responsible for making car move onto a lane
from intersection during emergency vehicle------------;;

to move-on-to-lane-during-emergency-vehicle

  ask cars
    [
      if ( [intersection?] of patch-here )
      [
        fd 1
        move-on-to-lane-during-emergency-vehicle
      ]
    ]
end

;;-------------------Stops the vehicles when there is an emergency
vehicle in the world-------------------------;;


to stop-vehicles-during-emergency

    set emergency-north-east North-east
    set emergency-north-west North-West
    set emergency-south-east South-east
    set emergency-south-west South-west

  if ( emergency-lane = 0 )
  [
    ask patches with [ pxcor = 3 and pycor = -2] [ set pcolor red]
    ask patches with [ pxcor = 2 and pycor = 3]  [ set pcolor red]
    ask patches with [ pxcor = -2 and pycor = -3][ set pcolor red]
```

```
    set North-east "red"
    set South-east "red"
    set South-west "red"
  ]
  if ( emergency-lane = 4 )
  [
    ask patches with [ pxcor = -3 and pycor = 2] [ set pcolor red]
    ask patches with [ pxcor = 3 and pycor = -2] [ set pcolor red]
    ask patches with [ pxcor = -2 and pycor = -3][ set pcolor red]
    set North-west "red"
    set South-east "red"
    set South-west "red"
  ]
  if ( emergency-lane = 3 )
  [
    ask patches with [ pxcor = -3 and pycor = 2] [ set pcolor red]
    ask patches with [ pxcor = 2 and pycor = 3]  [ set pcolor red]
    ask patches with [ pxcor = -2 and pycor = -3][ set pcolor red]
    set North-east "red"
    set North-west "red"
    set South-west "red"
  ]
  if ( emergency-lane = 7 )
  [
    ask patches with [ pxcor = -3 and pycor = 2] [ set pcolor red]
    ask patches with [ pxcor = 3 and pycor = -2] [ set pcolor red]
    ask patches with [ pxcor = 2 and pycor = 3]  [ set pcolor red]
    set North-east "red"
    set North-west "red"
    set South-east "red"
  ]

  move-on-to-lane-during-emergency-vehicle

  ask cars with [ lane = 5 or lane = 1 or lane = 6 ]
  [ forward-cars-as-per-speed ]
end

;;--------------------Resumes all other cars when emergency vehicle
moved out (In case if all cars are stopped)---------;;

to resume-all-the-vehicles-after-emergency-vehicle-moved-out

  if ( (North-west = "red" or North-west = 0) and
       (North-east = "red"  or North-east = 0) and
       (South-east = "red"  or South-east = 0) and
       (South-west = "red"  or South-west = 0))
  [
    set North-east emergency-north-east
    set North-West emergency-north-west
    set South-east emergency-south-east
    set South-west emergency-south-west

    if ( North-west = "green" )
```

```
     [
      ask patch -3 2 [ set pcolor green]
     ]

     if ( North-east = "green" )
     [
       ask patch 2 3  [ set pcolor green]
     ]

     if ( South-east = "green")
     [
        ask patch 3 -2  [ set pcolor green]
     ]

     if ( South-west = "green" )
     [
      ask patch -2 -3  [ set pcolor green]
     ]
  ]
end

;;------------------------Procedure responsible for assigning rash
behaviour to the rash car--------------------;;

to rash-driver-behaviour

  ask cars with [ rash-car? = true ]
  [
     if ( front-car != nobody and front-car != 0)
    [
     if ( lane = 0 or lane = 2)
     [
      set xcor xcor + 1
      set ycor  1
    ]

    if ( lane = 1 or lane = 3)
    [
      set xcor xcor - 1
      set ycor  -1
    ]

    if ( lane = 4 or lane = 6)
    [
      set xcor 1
      set ycor ycor - 1
    ]

    if ( lane = 5 or lane = 7)
    [
      set xcor -1
      set ycor ycor + 1
    ]
  ]
```

```
  if ( front-car = nobody )
  [
  if ( (lane = 0 or lane = 2 ) and ycor = 1)
  [
    if ( not any? cars-at (xcor + 2) 2 )
    [
      set xcor xcor + 2
      set ycor 2
    ]
  ]


  if ( (lane = 3 or lane = 1 ) and ycor = -1 )
  [
    if ( not any? cars-at ( xcor - 2) -2 )
     [
        set xcor xcor - 2
        set ycor -2
     ]
  ]

  if ( (lane = 4 or lane = 6 ) and xcor = 1 )
  [
    if ( not any? cars-at 2 (ycor - 2) )
      [
        set xcor 2
        set ycor ycor - 2
      ]
    ]

  if ( (lane = 5 or lane = 7 ) and xcor = -1 )
  [
    if ( not any? cars-at -2 (ycor + 2) )
      [
        set xcor -2
        set ycor ycor + 2
      ]
  ]
  ]

  if ( lane = 4 and ([pycor] of patch-here <= 3  or [intersection?]
of patch-here ) and (North-east = "red" or North-east = 0 ))
      [
         die
      ]
      if ( lane = 0 and ([pxcor] of patch-here >= -3 or
[intersection?] of patch-here ) and (North-west = "red" or North-
west = 0))
      [
         die
      ]

      if ( lane = 3 and ([pxcor] of patch-here <= 3 or
```

```
[intersection?] of patch-here ) and (South-east = "red" or South-
east = 0) )
      [
        die
      ]

      if ( lane = 7 and ([pycor] of patch-here >= -3 or
[intersection?] of patch-here ) and (South-west = "red" or South-
west = 0 ) )
      [
         die
      ]
]

end


to cars-new-behaviour

  ask cars
  [
     if ( front-car != nobody and front-car != 0)
    [
      if ( change-lane = 1 )
      [
       if ( lane = 0 or lane = 2)
       [
         set xcor xcor + 1
         set ycor  1
       ]

      if ( lane = 1 or lane = 3)
      [
       set xcor xcor - 1
       set ycor  -1
      ]

      if ( lane = 4 or lane = 6)
      [
       set xcor 1
       set ycor ycor - 1
      ]

      if ( lane = 5 or lane = 7)
      [
       set xcor -1
       set ycor ycor + 1
     ]
    ]
   ]
  ]

end
```