Project Report On

Traffic Optimization

submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology

in

Computer Science & Engineering

By

Avinash Haresh (RET18CS055)

Under the guidance of
Mr. Ajith S

Department of Computer Science & Engineering
Rajagiri School of Engineering and Technology
Rajagiri Valley, Kakkanad, Kochi, 682039

June 2022

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
RAJAGIRI SCHOOL OF ENGINEERING AND TECHNOLOGY
RAJAGIRI VALLEY, KAKKANAD, KOCHI, 682039

RSET
RAJAGIRI SCHOOL OF
ENGINEERING & TECHNOLOGY
(AUTONOMOUS)

CERTIFICATE

This is to certify that Project report entitled ***"Traffic Optimization"*** *report of the design presented during the eighth semester by* ***Avinash Haresh(RET18CS055)*** *to the APJ Abdul Kalam Technological University in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology (B.Tech) in Computer Science & Engineering during the academic year 2021-2022.*

Dr. Dhanya P M	**Dr. Jisha G**	**Mr. Ajith S**
Head of Department	*Project Coordinator*	*Project Guide*
Dept. of CSE	*Asst.Professor*	*Asst.Professor*
RSET	*Dept. of CSE*	*Dept. of CSE*
	RSET	*RSET*

External Examiner ***Internal Examiner***

ACKNOWLEDGEMENTS

I wish to express my sincere gratitude towards Dr.P.S. Sreejith, Principal of RSET, and Dr.Dhanya P.M, Head of Department of Computer Science & Engineering for providing me the opportunity to present the project "Traffic Optimization".

I am highly indebted to my project coordinators, Dr.Jisha G, Assistant Professor, Department of CSE, Ms.Meenu Mathew, Assistant Professor, Department of CSE, and Dr.Sminu Izudheen, Assistant Professor, Department of CSE for their valuable support.

It is indeed my pleasure and a moment of satisfaction to express sincere thanks to my project guide Mr.Ajith S, for his patience and all priceless advice and for all the wisdom he has shared with me.

Last but not the least, I would like to express my sincere gratitude towards all other teachers and friends for their continuous support and constructive ideas.

Avinash Haresh

ABSTRACT

Traffic congestion is becoming a serious problem with the large number of cars on the roads. Vehicles queue length waiting to be processed at the intersection is rising sharply with the increase of the traffic flow, and the present traffic light system cannot efficiently schedule it. Static timer based traffic lights find it difficult to adapt to dynamic traffic conditions. In this project,traffic signal phases are optimized according to collected data i.e vehicles queued in each direction, to enable as many vehicles to pass safely with minimum waiting time. We are planning to develop a traffic light system simulation with vehicle detection and counting to provide an efficient solution to traffic in a four-way junction.

Contents

List of Figures

Chapter 1

Introduction

1.1 Problem Statement

The present-day traffic signal control only works on a fixed time delay basis irrespective of the vehicle density. Traffic lights needs an improvement over the present static timed method where the number of queued vehicles at each direction is not considered. It increases the traffic congestion and shows the inefficiency of our traffic optimization system.

1.2 Project Objective

The prime objective of my project is to develop a traffic junction simulation which includes a dynamic timer system with vehicle detection and counting to provide an efficient solution to traffic in a four-way junction.

1.3 Motivation

It is a fact that there has been a dramatic increase in the number of vehicles on the roads in the past decade. This significant increase enforced modern transportation systems to promote new ways of managing traffic at intersections. Existing systems control traffic by providing a static timed signals for roads at all directions in an intersection. This system cannot change according to the varying traffic conditions through out the day and it increases the traffic congestion.Therefore, dynamically adjusting traffic lights according to the real time data have become the mainstream. Numerous methods involving different ways to deviate the traffic have been invoked in the systems by controlling the transportation via identification and recognition.

1.4 Assumptions and Dependencies

- The system is applicable in 4 way intersection roads.
- We are considering a junction between a highway road and a city road.
- The highway road contains more frequent vehicles compared to the city road.
- All vehicles come at a minimum speed in order to stop at the right time.
- Pedestrian crossing is not considered in the system.

1.5 Project Scope

- **Reduce dependency on manual traffic control systems**

  Traffic officers manually controlling traffic during rush hours is a common sight. Usage of an adaptive system avoids any kind of manual labour as the system is fully automated and changes according to the traffic conditions.

- **Easy to install with very less hardware requirements**

  The hardware requirement for this system are CCTV cameras which are already present in roads. It can also be used by officers for surveillance.

1.6 Target Group

- **Traffic Control Department**

  This system does not require any manpower.So it helps the traffic control department to increase the efficiency of current traffic control system and to reduce the traffic congestion at various junctions.

1.7 Summary of Report

The primary goal of this project was to create a simulation that contains a 4-way intersection along with four traffic signals.Vehicles were randomly generated and allowed to take left and right turns.Vehicle count was taken at each side and green light time was allotted accordingly.

Simulation time is displayed at the top right corner of the simulation window.Number of vehicles passed through the junction will also be counted.Then we compare the efficiency of the current traffic system and our proposed system.

Chapter 2

Literature Survey

Reference [1] proposes a solution using video processing. The video from the live feed is processed before being sent to the servers where a C++ based algorithm is used to generate the results. Hard code and Dynamic coded methodologies are compared, in which the dynamic algorithm showed an improvement of 35 percentage.

This research [2] proposes an Arduino-UNO based system that aims to reduce traffic congestion and waiting time. This system acquires images through the camera and then processes the image in MATLAB, where the image is converted to a threshold image by removing saturation and hues, and traffic density is calculated. Arduino and MATLAB are connected using USB and simulation packages, which are preinstalled. Depending on traffic count and traffic density, the Arduino sets the duration of green light for each lane. But this method has several flaws. The cars often overlap with each other and it is difficult to get a proper count of how many vehicles are on the road. Moreover, different objects interfered with the detection as they too were converted to black and white and there was no way of making a distinction between regular objects like billboards, poles, and trees with vehicles.

Reference [3] proposes a framework, which has the capability to continuously convey the vehicle count and generate an alarm in case of large vehicle gathering to the controlling station at various cities. The number of vehicles passing through a location well before the required traffic junction can be estimated using the help of image processing techniques. Further, the monitoring details can be shared to a distant controlling centre situated anywhere in the city through internet usage.

Reference [4] proposes a system which is an image processing based adaptive signal controlling. The timing will be calculated each time change automatically depending upon the traffic load. Proposed system will be functioning based on traditional system along with automated signalling. System will have artificial vision with the help of digital camera mounted on motor for its rotation to face lanes and sense the traffic on the road.

Reference [5] proposes a traffic control and management system which will detect the movement of vehicles, identify,track and count the numbers of vehicles in the lane by analysing a real-time live video feed form the camera with the help of computer vision and after detecting,classifying and counting the numbers of vehicle on a specific lane it will control traffic light according to the set threshold value, threshold value will be based on two criteria one is the density count of the lane and other is priority of that lane.

Chapter 3

System Architecture

In this chapter, the system architecture is discussed along with the the basic functioning of the system. In this traffic simulation system, we can analyse results between traffic congestion caused by the current traffic system and dynamic traffic system.

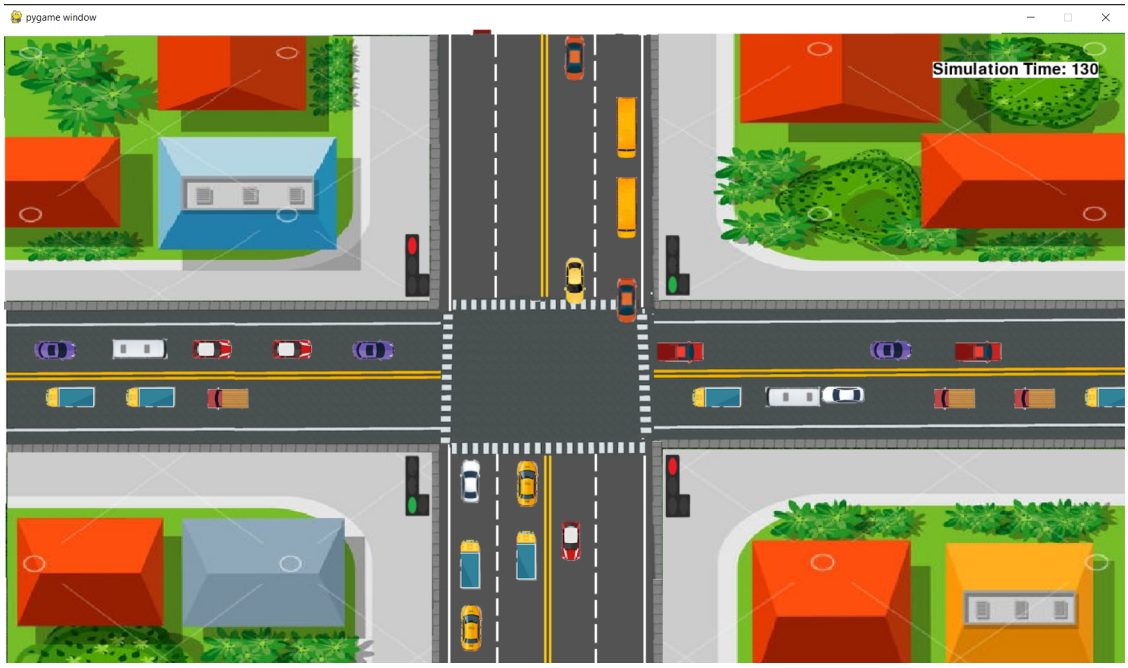


Figure 3.1: Simulation

In the traffic optimization system, the cameras at the four intersections will detect the vehicles and gives the count.According to the density of vehicles, the green light time will be allocated to that particular road.Then it will be cut and green signal will be given for the opposite direction.Left turn is automatic.We are allowing both the sides simultaneously and then each right turn so that more number of vehicles can cross the junction.

The simulation window is shown in figure 3.1. The procedure continues for the complementary roads also.Since the vehicle density at the National Highways are greater than the town roads, more priority will be given to National Highways.

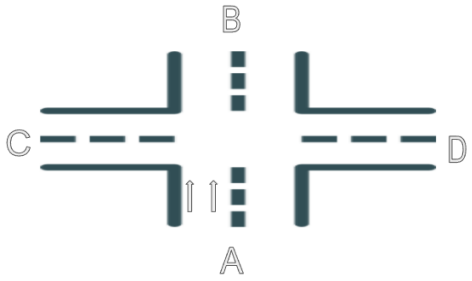


Figure 3.2: Road Representation

The system provides an amber light between red and green light so it will help the drivers to stop at red signal.Some vehicles at high speed will not be able to stop at the right time.So the system includes amber light with a specific time to warn the drivers to stop when the signal changes to red.

Vehicles are aligned in two lane road in highways and single lane in local town road.Those vehicles which are taking the right turn will be at right side of the road and those who are taking straight and left turn will be at the left side of the the road.So there will not be any block between the vehicles going at different directions.

Chapter 4

Detailed System Design

4.1 Functional Requirement

- Simulation that contains 4-way intersection and vehicles.
- Random generation of vehicles from each side.
- Smooth and random turning of vehicles
- Vehicle counting and algorithm implementation.
- Dynamic change of signals
- Display the simulation time and vehicles passed.

4.2 Use Case Diagram

The use case diagram is a diagram that demonstrates the different ways the user might interact with the system. The use case diagram consists of actors (user) and use cases. In the case of traffic light optimization shown in 4.1, the actors are the traffic officials.

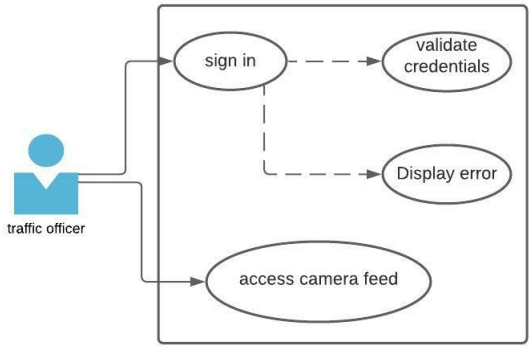


Figure 4.1: Use Case Diagram

The use case for traffic officials is to login into the system with their official credentials and then use the camera system to access the camera feed for further uses like looking for evidence, assessing traffic conditions etc.

4.3 Sequence Diagram

The user can login into the system with their official login ID and Password. If the credentials given by the users are wrong, they are sent back to the home page. If the credentials provided by the users are correct, they are given permission to access the camera feed for official uses. The sequence of this process is shown in figure 4.2.

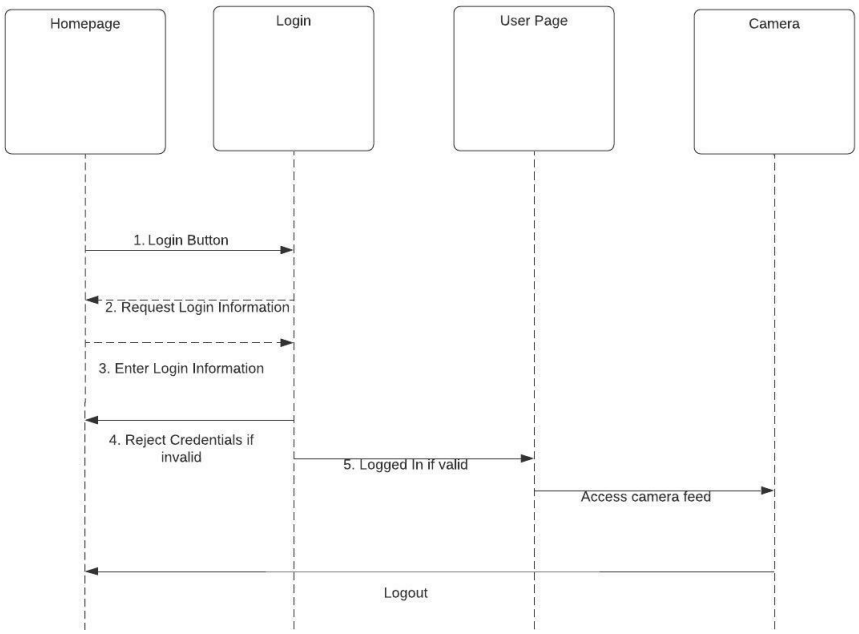


Figure 4.2: Sequence Diagram

4.4 Software Requirement

1. Pygame

Pygame is a cross-platform set of Python modules designed for writing video games. It includes computer graphics and sound libraries designed to be used with the Python programming language.This is based on the assumption that the most expensive functions inside games can be abstracted from the game logic, making it possible to use a high-level programming language, such as Python, to structure the game.

4.5 Background and objects

Pygame is a cross-platform set of Python modules designed for writing video games. It includes computer graphics and sound libraries designed to be used with the Python programming language.This is based on the assumption that the most expensive functions inside games can be abstracted from the game logic, making it possible to use a high-level programming language, such as Python, to structure the game.

Figure 4.3: objects

Chapter 5

Proposed Solution

Our proposed solution uses video from cameras placed at junction as input to count the number of vehicles waiting at the junction in each direction. YOLO program is used on the input from the cameras to count the number of vehicles of each classes such as cars, bikes etc. These values are then used to calculate a green light time efficient enough to pass the vehicles through the junction. The green light time is restricted to a minimum and maximum value to avoid starvation of a particular lane. A simulation is developed to demonstrate the system's effectiveness and compare it with the existing static system. This simulation results gives output in the form of number of vehicles that have passed the junction in a particular amount of time(Simulation Runtime).

5.1 Simulation Module

A simulation was developed from scratch using Pygame to simulate real-life traffic. It assists in visualizing the system and comparing it with the existing static system. It contains a 4-way intersection with 4 traffic signals. To get the effectiveness of the system, the simulation considers a scenario where the intersection is between a highway and city to city road. The vehicles are generated in a way that more vehicles will be there on the highway than the city to city road. Vehicles such as cars, trucks and buses come in from all 4 directions randomly. There will also be randomness in whether a vehicle turn left, right or goes straight after crossing the junction.

The rotation of the traffic is using a method of comparison where the vehicle density is taken in two opposite sides and are then compared. The green signal time is calculated for both the sides. The right green signal is given for the direction with more density. The

right green signal is given as 20 percent of the calculated total time. After that the the straight green signal is given for both the opposite direction which is enough for vehicles intending to go straight from both direction. After that, the right green signal is given for the direction with the side with lesser density. This is then continued for the other two opposite sides and so on.

After the simulation is complete, the program produces the output of the duration of the simulation and the number of vehicles have passed the junction during this time. This value is used to compare between the current system and the dynamic system. Both the simulations are run for a duration of 2, 4, 6, 8, 10 minutes.

5.2 Random generation of vehicles

Vehicles are generated randomly to show the simulation more realistic.New vehicles are generated every one second.There are three types of vehicles considered for this simulation; Car,Truck and Bus.

5.3 Smooth and random turning of vehicles

Vehicles are turned to left and right randomly.Vehicles are rotated in three angles 30,60,90 degrees.Pygame module will reduce the object quality while rotating.So it is turned by considering as three frames.Left turn is automatic and right turn functionality is generated randomly. Right turn will be turned off while vehicles move simultaneously from both sides to avoid collision.

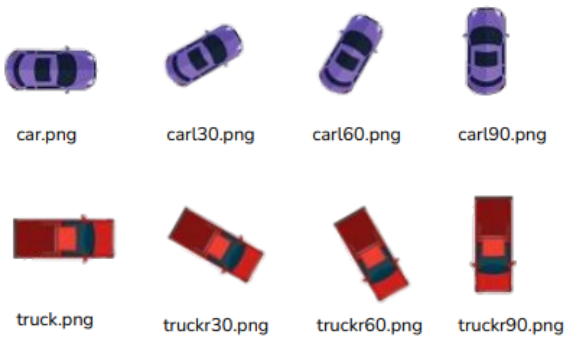


Figure 5.1: Turning of vehicles

5.4 Vehicle counting and algorithm implementation

Vehicles generated from each side will be counted for the calculation of green signal time at the corresponding sides.Whenever the vehicles cross the red line shown in 5.2, it will be considered as passed. The equation used for green signal time is shown in 5.3.

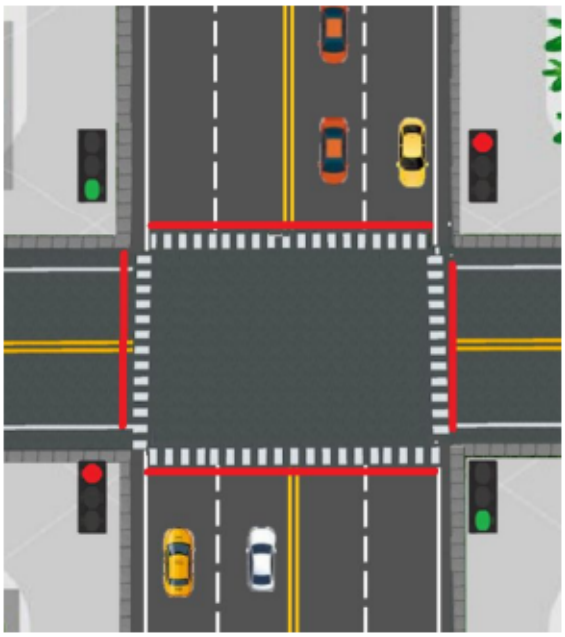


Figure 5.2: Vehicle counting

Green Signal Time = 2* (vehiclelcount + vehiclercount) +1

OR

Green Signal Time = 2* (vehicleucount + vehicledcount) +1

Figure 5.3: Equation for green signal time

5.5 Dynamic signal change

Green light time changes at each side according to the vehicle density.First right turn will be ON for 20 percent of the full green light time.Then the right turn will be OFF and simultaneously both sides will be opened.The minimum green light time when there is no vehicles is 10 seconds.The maximum green light time when there are so many vehicles is 40 seconds.The correct order of the signal change is shown in 5.4.This method helps more vehicles to pass through a junction.

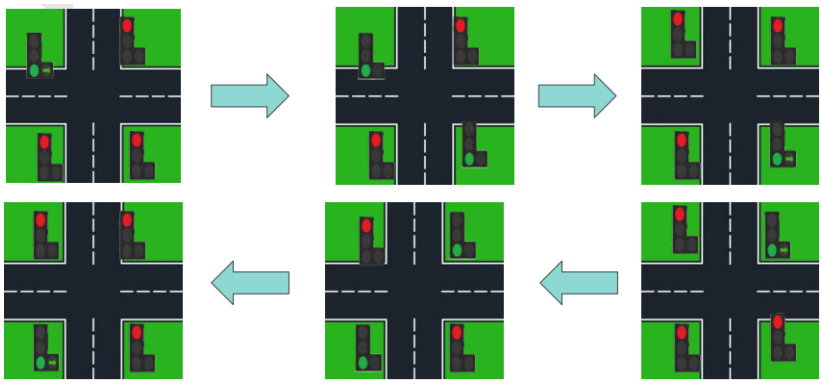


Figure 5.4: Order of signal change

Chapter 6

Risks and Challenges

6.1 Risks

- **Violation of traffic rules by the drivers**

  The system works on dynamic signal timers and if drivers doesn't obey the traffic rules,it leads to an accident.

- **Pedestrian crossing**

  Green light time is not fixed. Pedestrians may not get sufficient time to cross the road. They must be aware of the the signal time.

6.2 Challenges

- **Arrival of an Emergency Vehicle**

  Since the whole system is a continuously working dynamic system, it has to be stopped on time to allow the emergency vehicles to pass.

- **Aligning of vehicles at the junction.**

  Vehicles taking right turn should align at the right side or else it may cause a traffic jam.

Chapter 7

Results and Analysis

To measure how the proposed adaptive system compares to the existing static system, 5 simulations of both systems were run for a period of 2, 4, 6, 8 and 10 minutes respectively with varying traffic distributions in all four directions. In all the simulations, the bigger two lane road contained more vehicles than small single lane road. Figure 2 shows the results obtained with this experiment. From the chart, it is found that the number of vehicles passing the junction in dynamic system is more than the static system and is increasing with the simulation run time.

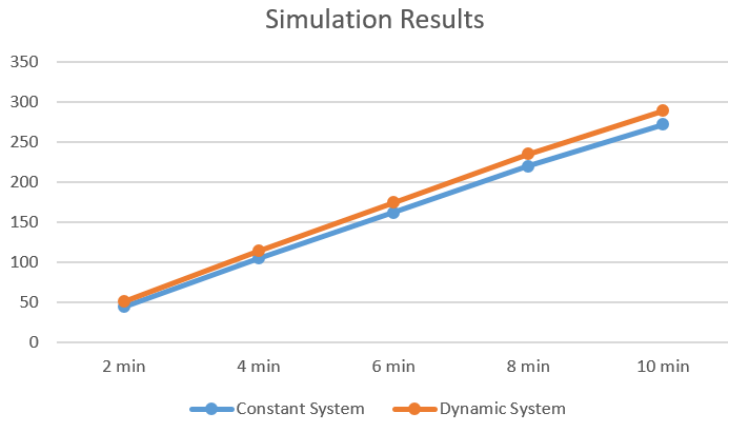


Figure 7.1: Simulation Results

Time Duration	Vehicles Passed With Constant Timer	Vehicles Passed With Dynamic Timer
2 min	44	51
4 min	105	114
6 min	162	175
8 min	220	235
10 min	272	289

Figure 7.2: Results of current system and proposed system

Chapter 8

Conclusion and Future Scope

8.1 Conclusion

In conclusion, the proposed system sets the green signal time adaptively according to the traffic density at the signal and ensures that the direction with more traffic is allotted a green signal for a longer duration of time as compared to the direction with lesser traffic. This will lower the unwanted delays and reduce congestion and waiting time, which in turn will reduce fuel consumption and pollution.

As per the simulation results, the proposed system shows and improvement over the current system. This result can be further improved by modifying the system according to the needs of the traffic junction and analyzing the traffic conditions of the area it is located.

Since CCTV cameras are present in almost every junctions in the country, there isn't much further hardware requirements for this system and maintenance cost is also compared to systems that uses sensors.Thus, the proposed system can thus be integrated with the CCTV cameras in major cities in order to facilitate better management of traffic..

8.2 Future Scope

- **Automatic handling of emergency vehicles**

  If specially trained models are there for detecting emergency vehicles such as ambulance,fire engines etc,they can be given priority automatically and the signals can be changed accordingly.

- **Fuel consumption and pollution rate**

  If this system becomes popular everywhere, there will be a huge reduction in pollution rate and fuel consumption.

Glossary

Intersection: The 4-way junction designed with various image editing softwares to demonstrate the movement of traffic. The intersection is filled with vehicles coming from all sides randomly. To add to the realism, one of the roads is made a highway road with two lanes and the other a city road with one lane. The highway road will tend to have more vehicles on it than the city road.

Constant Simulation: A simulation created to demonstrate the working of traffic signals which allow vehicles to pass through the intersection with the same green time regardless of the number of vehicles present in a particular direction.

Dynamic Simulation: A simulation created to demonstrate the working of a dynamic traffic signal timer which allow vehicles to pass through the intersection with green time varying according to the vehicle density present in each direction.

Green signal time: It is a numeric value in seconds which determines how much time is allotted for vehicles to pass through in an intersection. These vehicles may go forward, left or right with each direction being a random possibility.

Amber light: It is a traffic light which indicates vehicles to clear the road when the light is changing from Green to Red. A fixed value is set for 3 seconds.

Pygame: It is a cross-platform set of Python modules designed for writing video games. It includes computer graphics and sound libraries designed to be used with the Python programming language.

Vehicle idling time: It is the time a vehicle waits in queue for change in the signal from Red to Green. The proposed system aims to decrease this value.

References

[1] Sasi Priya, S. Rajarajeshwari, K. Sowmiya, P. Vinesha and A Janan, *"Dynamic Traffic Light Control System"*, Department of ECE, Sri Krishna College of Engineering and Technology, 2021, DOI:10.35940/ijrte.F8609.038620.

[2] Y. N. Udoakah and I. G. Okure, 2020, *"Design Implementation of a Density-based Traffic light control with surveillance System"*, University of Uyo, 2017, http://dx.doi.org/10.4314/njt.v36i4.34

[3] Satbir Singh, Baldev Singh, Ramandeep, Baljit Singh, Amitava Das, *"Automatic Vehicle Counting for IoT based Smart Traffic Management System for Indian urban Settings,"* 2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU) DOI: 10.1109/IoT-SIU45981.2019.

[4] Prashant Jadhav, Pratiksha Kelkar, Kunal Patil, Snehal Thorat, "Smart Traffic Control System Using Image Processing," International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056.

[5] Manish Kumar Singh, Krishna Deep Mishra , Subrata Sahana, "An Intelligent Realtime Traffic Control Based on Vehicle Density,"2021 International Journal of Engineering Technology and Management Sciences, Issue: 3 Volume No.5 May – 2021 DOI: 10.46647/ijetms.2021.v05i03.004 ISSN: 2581-4621.

Appendix A
Presentation slides

CS492 PROJECT

PROJECT PRESENTATION

GUIDED BY

MR. AJITH S

GROUP MEMBERS

ADAM BASIL ANIL - 8
ASWIN TOMY - 50
ATHUL MURALI - 51
AVINASH HARESH - 54

1

2

Contents

3

PROBLEM STATEMENT

- The present-day traffic signal control only works on a fixed time delay basis irrespective of the vehicle density.
- Traffic jams not only cause extra delay and stress for the drivers, but also increase fuel consumption, add transportation costs, and increase air pollution.
- Traffic lights needs an improvement over the present static timed method where the number of queued vehicles at each direction is not considered.

4

Conventional Systems

- **Manual Controlling**: As the name suggests, it requires manpower to control the traffic. The traffic police are allotted for a required area to control traffic. The traffic police carry signboard, sign light and whistle to control the traffic.
- **Automatic Controlling**: Automatic traffic signals are controlled by timers and electrical sensors in traffic light, a constant numerical value is loaded in timer. The lights automatically getting ON and OFF based on timer value.
- **Electronic Sensors**: Another advanced method is placing some loop detectors or proximity sensors on the road. The sensors gives data about the traffic on the road. According to the sensor data, the traffic signals are controlled.

5

SOLUTION

When the timers of these traffic signals are set according to vehicle density in each direction, the system becomes adaptive.

-Takes the vehicle count at each side.

-Increase the green signal time for sides having more vehicle density.

-Preset a maximum and minimum Green Time to avoid starvation of a lane.

Advantages

-It reduces the idle time being wasted at the junction.

-It reduces the traffic congestion.

-No need of manpower.

-Decreases fuel consumption.

6

SCOPE & OBJECTIVES

PROJECT SCOPE

- Proposed system can be used for the development of current traffic control system.
- Reduction in pollution rate caused by traffic congestion.
- Decrease in the consumption of fuel by automobiles.

OBJECTIVES

- To make a simulation to show the efficiency of the dynamic traffic timer system.
- To reduce traffic congestion at the junction.
- To decrease the fuel consumption of vehicles waiting at junction.

7

ASSUMPTIONS & CONSTRAINTS

ASSUMPTIONS:

- There is a highway on one side of the road and a local city road in the other side.
- Right turning vehicles are less common compared to straight-going vehicles.
- Traffic rules are followed by all vehicles.

CONSTRAINTS:

- Only works on 4-way intersections.

8

HARDWARE & SOFTWARE REQUIREMENTS

HARDWARE REQUIREMENTS

- A Local PC with basic configuration and storage.

SOFTWARE REQUIREMENTS

- Packages: Pygame
- Visual Studio Code

9

LITERATURE SURVEY

PAPER NAME	AUTHOR	ADVANTAGE
Smart Control of Traffic Light Using Artificial Intelligence	Mihir M. Gandhi, Devansh S. Solanki, Rutwij S. Daptardar, Nirmala Shinde Baloorkar	Switching the traffic lights based on the vehicle density to reduce congestion
Smart Traffic Control System Using Image Processing	Vismay Pandit , Jinesh Doshi, Dhruv Mehta , Ashay Mhatre and Abhilash Janardhan	Edge detection has been carried out using Prewitt edge detection operator and according to percentage of matching traffic light durations can be controlled.

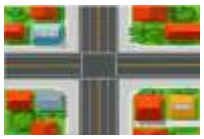
10

PROJECT OVERVIEW

- In this project, a simulation is created from scratch using Pygame to replicate the movement of vehicles across a traffic intersection.
- It is a 4-way intersection with traffic signals in each direction.
- A second simulation window was created to compare the efficiency of our dynamic traffic system over current traffic system.
- The simulation has two outputs. (Simulation time & Number of vehicles passed)

11

Creation of background and objects



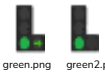
mod_int.png

car.png truck.png

bus.png

12

Creation of background and objects



green.png green2.png

red.png yellow.png

Initializing objects

Displaying objects

13

MODULAR DIVISION

MODULE 1	Random vehicle generation
MODULE 2	Smooth and random turning of vehicles
MODULE 3	Vehicle counting and Implementation
MODULE 4	Dynamic signal change algorithm

14

Random vehicle generation

- VehicleGenerate() function is used to generate vehicles.
- Vehicle type and lane number is randomized using a random function to make the simulation more realistic.
- Vehicle Type [car,bus,truck].
- A new vehicle is added to the simulation after every one second.

15

Random vehicle generation

16

Smooth and Random turning of vehicles

- In this module, we added the ability of vehicles to turn to either left or right.
- This was performed using 3 different objects (images) of vehicles to show the turning.
- It is done to avoid the blurring of vehicles.
- Objects are running in a loop in 3 directions:
  - 30 degree
  - 60 degree
  - 90 degree

17

Smooth and random turning of vehicles

car.png car30.png car60.png car90.png

truck.png truckr30.png truckr60.png truckr90.png

18

Vehicle Counting

- Vehicles passing through those coordinates (marked as red) will be counted as passed.
- Random vehicles which enters into the screen will be considered for counting at each side.
- This vehicle count will be the factor for the equation of green signal time.

19

Algorithm Implementation

Green Signal Time = 2* (vehiclelcount + vehiclercount) +1

OR

Green Signal Time = 2* (vehicleucount + vehicledcount) +1

where,

vehiclelcount = vehicle count at left direction

vehiclercount = vehicle count at right direction

vehicleucount = vehicle count at up direction

vehicledcount = vehicle count at down direction

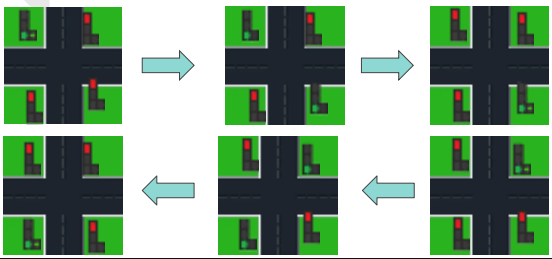
20

Dynamic change of signals

- 'counter' is the variable used for the green signal time.
- If the next green signal is for up and down direction, counter is set as 2* (vehicleucount + vehicledcount) +1.
- If the next green signal is for left and right direction, counter is set as 2* (vehiclelcount + vehiclercount) +1.
- Maximum green signal time is set as 40 seconds.
- Minimum green signal time is set as 10 seconds.
- 20% of counter value is given for right turn.
- If the counter value exceeds the maximum time, counter is set as 40.
- If the counter value is less than the minimum time, counter is set as 10.

21

Order of signal change



22

RESULTS

There are two outputs for this simulation:

- ➔ Simulation Time
- ➔ Number of vehicles passed

We compare the number of vehicles passed in the current traffic system and dynamic traffic system at a constant time inorder to check the efficiency of the new system.

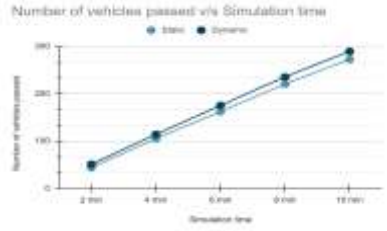
23

RESULTS

Time Duration	Vehicles Passed With Constant Timer	Vehicles Passed With Dynamic Timer
2 min	44	51
4 min	105	114
6 min	162	176
8 min	220	235
10 min	272	289

24

RESULTS



25

CONCLUSION

It is clear that the proposed system sets the green time adaptively according to the traffic density at the junction, giving more green time duration for sides having more vehicle density.

The graph shows that the dynamic system is more efficient than the current traffic system. When the simulation time increases, efficiency also increases. It reduces the waiting time of vehicles at the junction. This in turn reduces pollution and decreases fuel consumption.

26

PROJECT DEMO

27

THANK YOU

28

Appendix B
Questionnaire

1. **What are the advantages of Pygame over other simulation development platforms?**

- **It's Simple** and easy to use. Kids and adults make shooter games with pygame. Pygame is used in the OLPC project and has been taught in essay courses to young kids and college students. It's also used by people who first programmed in z80 assembler or c64 basic.
- **Truly portable**. Supports Linux (pygame comes with most main stream linux distributions), Windows (95, 98, ME, 2000, XP, Vista, 64-bit Windows, etc), Windows CE, BeOS, MacOS, Mac OS X, FreeBSD, NetBSD, OpenBSD, BSD/OS, Solaris, IRIX, and QNX. The code contains support for AmigaOS, Dreamcast, Atari, AIX, OSF/Tru64, RISC OS, SymbianOS and OS/2, but these are not officially supported. You can use it on hand held devices, game consoles and the One Laptop Per Child (OLPC) computer.
- **You control your main loop**. You call pygame functions, they don't call your functions. This gives you greater control when using other libraries, and for different types of programs.
- **Comes with many operating systems**. Just an apt-get, emerge, pkg_add, or yast install away. No need to mess with installing it outside of your operating system's package manager. Comes with binary pos system installers (and uninstallers) for Windows or MacOSX. Pygame does not require setup tools with even ctypes to install.
- **Multi core CPUs can be used easily**. With dual core CPUs common, and 8 core CPUs cheaply available on desktop systems, making use of multi core CPUs allows you to do more in your game. Selected pygame functions release the dreaded python GIL, which is something you can do from C code.
- **Fast response to reported bugs**. Some bugs are patched within an hour of being reported. Do a search on our mailing list for BUG... you'll see for yourself. Sometimes we suck at bug fixes, but mostly we're pretty good bug fixers. Bug reports are quite rare these days, since a lot of them have been fixed already.
- **Small amount of code**. It does not have hundreds of thousands of lines of code for things you won't use anyway. The core is kept simple, and extra things like GUI libraries, and effects are developed separately outside of pygame.

2. **What are the impacts of traffic jams on humans?**

- Twofold higher risk of Type 2 diabetics is also observed for people exposed to intense traffic. Long-term exposure to particulate matter increases type two diabetes risk in the general population, as does living close to a major road. Also, particulate matter and the increase in oxidative stress have ill-effects on the respiratory system. Besides, traffic noise is significantly associated with adverse pregnancy outcomes and can affect the hormonal and digestive system.
- Studies have also shown that the level of blood adrenaline hormone increases in heavy-traffic compared with during low-traffic conditions. Also, the ability to estimate distance and recognition is reduced due to high-traffic compared to during low-traffic conditions.

The increase of the adrenaline during driving on heavy-traffic days can lead to stress because one's accuracy and judgment may be reduced.

- Traffic jams cause vehicles to wait in queues for long times. This results in the consumption of a large amount of fuel, a non-renewable energy. Along with that drivers can save on the cost of fuel with less traffic jams.
- A traffic jam means more vehicular emissions and more pollutants inundating the air. This slowly degrades the quality of air and affects all those who sit for hours at the end amidst traffic congestion. Hence one may say that traffic jams significantly increase air pollution.
- An interesting thing to note is that the pollution gathered inside the cars in traffic jams is far higher than that around the cars moving outside. Some of the dangerous and long-lasting effects of traffic jam-induced air pollution are chronic lung diseases, high blood pressure, stroke and asthma.

3. What is YOLO and how is it used to detect and categorize vehicles?

- YOLO is an algorithm that uses neural networks to provide real-time object detection. This algorithm is popular because of its speed and accuracy. It has been used in various applications to detect traffic signals, vehicles, etc.
- Object detection in YOLO is done as a regression problem and provides the class probabilities of the detected images.

- Main features of this algorithm are:
    1. **Speed**: This algorithm improves the speed of detection because it can predict objects in real-time.
    2. **High accuracy**: YOLO is a predictive technique that provides accurate results with minimal background errors.

The image is first divided into grid cells. Each grid cell forecasts B bounding boxes and provides their confidence scores. The cells predict the class probabilities to establish the class of each object. Using this the type of object is detected and categorized.

Appendix C
Program Code

Traffic control using static timers

Static.py

```
import pygame
import random
import time
import timeit

# initialize pygame
pygame.init()
start = timeit.default_timer()
black = (0, 0, 0)
white = (255, 255, 255)
simtime = 0
font = pygame.font.Font(None, 30)
# create game screen
screen = pygame.display.set_mode((1400, 800))
background = pygame.image.load('images/mod_int.png')

# players
carl = pygame.image.load('images/left/car.png')
carll30 = pygame.image.load('images/left/carl30.png')
carll60 = pygame.image.load('images/left/carl60.png')
carll90 = pygame.image.load('images/left/carl90.png')
carlr30 = pygame.image.load('images/left/carr30.png')
carlr60 = pygame.image.load('images/left/carr60.png')
carlr90 = pygame.image.load('images/left/carr90.png')
carr = pygame.image.load('images/right/car.png')
carrl30 = pygame.image.load('images/right/carl30.png')
carrl60 = pygame.image.load('images/right/carl60.png')
carrl90 = pygame.image.load('images/right/carl90.png')
carrr30 = pygame.image.load('images/right/carr30.png')
carrr60 = pygame.image.load('images/right/carr60.png')
carrr90 = pygame.image.load('images/right/carr90.png')
caru = pygame.image.load('images/up/car.png')
carul30 = pygame.image.load('images/up/carl30.png')
carul60 = pygame.image.load('images/up/carl60.png')
carul90 = pygame.image.load('images/up/carl90.png')
carur30 = pygame.image.load('images/up/carr30.png')
carur60 = pygame.image.load('images/up/carr60.png')
carur90 = pygame.image.load('images/up/carr90.png')
card = pygame.image.load('images/down/car.png')
cardl30 = pygame.image.load('images/down/carl30.png')
cardl60 = pygame.image.load('images/down/carl60.png')
cardl90 = pygame.image.load('images/down/carl90.png')
cardr30 = pygame.image.load('images/down/carr30.png')
cardr60 = pygame.image.load('images/down/carr60.png')
cardr90 = pygame.image.load('images/down/carr90.png')
busl = pygame.image.load('images/left/bus.png')
busll30 = pygame.image.load('images/left/busl30.png')
busll60 = pygame.image.load('images/left/busl60.png')
```

```
busll90 = pygame.image.load('images/left/busl90.png')
buslr30 = pygame.image.load('images/left/busr30.png')
buslr60 = pygame.image.load('images/left/busr60.png')
buslr90 = pygame.image.load('images/left/busr90.png')
busr = pygame.image.load('images/right/bus.png')
busrl30 = pygame.image.load('images/right/busl30.png')
busrl60 = pygame.image.load('images/right/busl60.png')
busrl90 = pygame.image.load('images/right/busl90.png')
busrr30 = pygame.image.load('images/right/busr30.png')
busrr60 = pygame.image.load('images/right/busr60.png')
busrr90 = pygame.image.load('images/right/busr90.png')
busd = pygame.image.load('images/down/bus.png')
busdl30 = pygame.image.load('images/down/busl30.png')
busdl60 = pygame.image.load('images/down/busl60.png')
busdl90 = pygame.image.load('images/down/busl90.png')
busdr30 = pygame.image.load('images/down/busr30.png')
busdr60 = pygame.image.load('images/down/busr60.png')
busdr90 = pygame.image.load('images/down/busr90.png')
busu = pygame.image.load('images/up/bus.png')
busul30 = pygame.image.load('images/up/busl30.png')
busul60 = pygame.image.load('images/up/busl60.png')
busul90 = pygame.image.load('images/up/busl90.png')
busur30 = pygame.image.load('images/up/busr30.png')
busur60 = pygame.image.load('images/up/busr60.png')
busur90 = pygame.image.load('images/up/busr90.png')
truckl = pygame.image.load('images/left/truck.png')
truckll30 = pygame.image.load('images/left/truckl30.png')
truckll60 = pygame.image.load('images/left/truckl60.png')
truckll90 = pygame.image.load('images/left/truckl90.png')
trucklr30 = pygame.image.load('images/left/truckr30.png')
trucklr60 = pygame.image.load('images/left/truckr60.png')
trucklr90 = pygame.image.load('images/left/truckr90.png')
truckr = pygame.image.load('images/right/truck.png')
truckrl30 = pygame.image.load('images/right/truckl30.png')
truckrl60 = pygame.image.load('images/right/truckl60.png')
truckrl90 = pygame.image.load('images/right/truckl90.png')
truckrr30 = pygame.image.load('images/right/truckr30.png')
truckrr60 = pygame.image.load('images/right/truckr60.png')
truckrr90 = pygame.image.load('images/right/truckr90.png')
truckd = pygame.image.load('images/down/truck.png')
truckdl30 = pygame.image.load('images/down/truckl30.png')
truckdl60 = pygame.image.load('images/down/truckl60.png')
truckdl90 = pygame.image.load('images/down/truckl90.png')
truckdr30 = pygame.image.load('images/down/truckr30.png')
truckdr60 = pygame.image.load('images/down/truckr60.png')
truckdr90 = pygame.image.load('images/down/truckr90.png')
trucku = pygame.image.load('images/up/truck.png')
truckul30 = pygame.image.load('images/up/truckl30.png')
truckul60 = pygame.image.load('images/up/truckl60.png')
truckul90 = pygame.image.load('images/up/truckl90.png')
truckur30 = pygame.image.load('images/up/truckr30.png')
```

```
truckur60 = pygame.image.load('images/up/truckr60.png')
truckur90 = pygame.image.load('images/up/truckr90.png')
bikel = pygame.image.load('images/left/bike.png')
bikell30 = pygame.image.load('images/left/bikel30.png')
bikell60 = pygame.image.load('images/left/bikel60.png')
bikell90 = pygame.image.load('images/left/bikel90.png')
bikelr30 = pygame.image.load('images/left/biker30.png')
bikelr60 = pygame.image.load('images/left/biker60.png')
bikelr90 = pygame.image.load('images/left/biker90.png')
biker = pygame.image.load('images/right/bike.png')
bikerl30 = pygame.image.load('images/right/bikel30.png')
bikerl60 = pygame.image.load('images/right/bikel60.png')
bikerl90 = pygame.image.load('images/right/bikel90.png')
bikerr30 = pygame.image.load('images/right/biker30.png')
bikerr60 = pygame.image.load('images/right/biker60.png')
bikerr90 = pygame.image.load('images/right/biker90.png')
biked = pygame.image.load('images/down/bike.png')
bikedl30 = pygame.image.load('images/down/bikel30.png')
bikedl60 = pygame.image.load('images/down/bikel60.png')
bikedl90 = pygame.image.load('images/down/bikel90.png')
bikedr30 = pygame.image.load('images/down/biker30.png')
bikedr60 = pygame.image.load('images/down/biker60.png')
bikedr90 = pygame.image.load('images/down/biker90.png')
bikeu = pygame.image.load('images/up/bike.png')
bikeul30 = pygame.image.load('images/up/bikel30.png')
bikeul60 = pygame.image.load('images/up/bikel60.png')
bikeul90 = pygame.image.load('images/up/bikel90.png')
bikeur30 = pygame.image.load('images/up/biker30.png')
bikeur60 = pygame.image.load('images/up/biker60.png')
bikeur90 = pygame.image.load('images/up/biker90.png')
green = pygame.image.load('images/signals/green.png')
green2 = pygame.image.load('images/signals/green2.png')
red = pygame.image.load('images/signals/red.png')
yellow = pygame.image.load('images/signals/yellow.png')


# current_signals
signall = ""
signalr = ""
signalu = ""
signald = ""


def player(vehicle, x, y):
    screen.blit(vehicle, (x, y))


# generating vehicles
def VehicleGenerate():
    dir = ''
    vehicle = ''
```

```
x = 0
y = 0
turn = ''
num1 = random.randint(0, 1001)
ger = [150, 500, 650, 1000]
if (num1 < ger[0]):
    dir = 'left'
elif (num1 > ger[0] and num1 < ger[1]):
    dir = 'up'
elif (num1 > ger[1] and num1 < ger[2]):
    dir = 'right'
elif (num1 > ger[2] and num1 < ger[3]):
    dir = 'down'
if dir == 'left':
    num2 = random.randint(0, 3)
    if num2 == 0:
        vehicle = carl
    elif num2 == 1:
        vehicle = bikel
    elif num2 == 2:
        vehicle = busl
    else:
        vehicle = truckl
    x = 10
    y = 385
    num3 = random.randint(0, 6)
    if num3 == 0:
        turn = 'l'
    elif num3 == 1:
        turn = 'r'
    else:
        turn = 's'
elif dir == 'up':
    num2 = random.randint(0, 3)
    if num2 == 0:
        vehicle = caru
    elif num2 == 1:
        vehicle = bikeu
    elif num2 == 2:
        vehicle = busu
    else:
        vehicle = trucku
    num3 = random.randint(0, 1)
    if num3 == 0:
        x = 765
        y = 5
        num4 = random.randint(0, 4)
        if num4 == 0:
            turn = 'l'
        else:
            turn = 's'
```

```
    else:
        x = 700
        y = 5
        num4 = random.randint(0, 4)
        if num4 == 0:
            turn = 'r'
        else:
            turn = 's'
elif dir == 'right':
    num2 = random.randint(0, 3)
    if num2 == 0:
        vehicle = carr
    elif num2 == 1:
        vehicle = biker
    elif num2 == 2:
        vehicle = busr
    else:
        vehicle = truckr
    x = 1350
    y = 445
    num3 = random.randint(0, 6)
    if num3 == 0:
        turn = 'l'
    elif num3 == 1:
        turn = 'r'
    else:
        turn = 's'
else:
    num2 = random.randint(0, 3)
    if num2 == 0:
        vehicle = card
    elif num2 == 1:
        vehicle = biked
    elif num2 == 2:
        vehicle = busd
    else:
        vehicle = truckd
    num3 = random.randint(0, 1)
    if num3 == 0:
        x = 570
        y = 760
        num4 = random.randint(0, 4)
        if num4 == 0:
            turn = 'l'
        else:
            turn = 's'
    else:
        x = 640
        y = 760
        num4 = random.randint(0, 4)
        if num4 == 0:
```

```
            turn = 'r'
        else:
            turn = 's'
    return vehicle, x, y, dir, turn


clock = pygame.time.Clock()
pygame.time.set_timer(pygame.USEREVENT, 1000)
running = True
vehicles = []
xpos = []
ypos = []
dirs = []
turns = []
turned = []
turned2 = []
ql = []
vehiclegen = 2
sg = 'lr'
sg2 = 'l'
a = 0
fl = 1
fu = 1
fr = 1
fd = 1
temp = 0
vehiclecount = 0
vehiclel = []
vehiclelcount = 0
vehiclercount = 0
vehicleucount = 0
vehicledcount = 0
vehicleLeft = 0
vehicleRight = 0
vehicleUp = 0
vehicleDown = 0
max = 40
rmax = 5
counter = max
counter2 = 2
counter3 = rmax
counter4 = rmax
c1 = 0
c2 = 0
while running:
    flag = 0
    for event in pygame.event.get():
        if event.type == pygame.USEREVENT:
            simtime += 1
            vehiclegen -= 1
            counter -= 1
```

```
            counter2 -= 1
            counter3 -= 1
        if event.type == pygame.QUIT:
            vehiclePassed = vehicleDown + vehicleRight + vehicleUp + vehicleLeft
            stop = timeit.default_timer()
            timeRun = stop - start
            timeRun = int(timeRun)
            print("Constant Simulation:")
            print('Simulation time:', timeRun, "seconds")
            print('Total no of vehicles passed:', vehiclePassed)
            running = False
    screen.fill((0, 0, 0))
    screen.blit(background, (0, 0))
    simtimeText = font.render(
        ("Simulation Time: " + str(simtime)), True, black, white)
    screen.blit(simtimeText, (1160, 40))
    if counter < 0:
        if sg == 'lr':
            sg = 'ud'
            fl = 0
            if vehicledcount > vehicleucount:
                sg2 = 'd'
            else:
                sg2 = 'u'
            counter = max
            counter3 = rmax
            counter4 = rmax
        elif sg == 'ud':
            sg = 'lr'
            fd = 0
            if vehiclelcount > vehiclercount:
                sg2 = 'l'
            else:
                sg2 = 'r'
            counter = max
            counter3 = rmax
            counter4 = rmax
    if counter >= 0 or flag == 1:
        if sg == 'lr':
            if fd == 0:
                counter2 = 2
                fd = 1
            if counter2 >= 0:
                signald = 'yellow'
                signalu = 'yellow'
            else:
                signald = 'red'
                signalu = 'red'
            if signald == 'yellow' or signalu == 'yellow':
                screen.blit(yellow, (500, 525))
                screen.blit(yellow, (825, 250))
```

```
            else:
                screen.blit(red, (500, 525))
                screen.blit(red, (825, 250))
            fdb = 0
            for m in range(vehiclecount):
                if (ypos[m] in range(490, 310, -1) and dirs[m] == 'down' and turns[m] == 's') or (ypos[m] in
range(310, 500) and dirs[m] == 'up' and turns[m] == 's') or (ypos[m] <= 490 and turned[m] == 0 and
dirs[m] == 'down' and turns[m] == 'r') or (ypos[m] >= 310 and turned[m] == 0 and dirs[m] == 'up' and
turns[m] == 'l') or (ypos[m] <= 490 and turned[m] == 0 and dirs[m] == 'down' and turns[m] == 'l') or
(ypos[m] >= 310 and turned[m] == 0 and dirs[m] == 'up' and turns[m] == 'r'):
                    fdb = 1
                    break
            if fdb == 1:
                signall = 'red'
                signalr = 'red'
                screen.blit(red, (825, 525))
                screen.blit(red, (500, 250))
            else:
                if sg2 == 'l':
                    if counter3 > 0:
                        signall = 'green'
                        screen.blit(green, (500, 250))
                        signalr = 'red'
                        screen.blit(red, (825, 525))
                    if counter3 <= 0:
                        if counter <= counter4:
                            signalr = 'green'
                            screen.blit(green, (825, 525))
                            signall = 'red'
                            screen.blit(red, (500, 250))
                        else:
                            signall = 'green2'
                            screen.blit(green2, (500, 250))
                            signalr = 'green2'
                            screen.blit(green2, (825, 525))
                else:
                    if counter3 > 0:
                        signalr = 'green'
                        screen.blit(green, (825, 525))
                        signall = 'red'
                        screen.blit(red, (500, 250))
                    if counter3 <= 0:
                        if counter <= counter4:
                            signall = 'green'
                            screen.blit(green, (500, 250))
                            signalr = 'red'
                            screen.blit(red, (825, 525))
                        else:
                            frb = 0
                            for w in range(vehiclecount):
                                if (xpos[w] <= 780 and turned[w] == 0 and dirs[w] == 'right' and turns[w] == 'r'):
```

```
                        frb = 1
                        break
                if frb == 0:
                    signalr = 'green2'
                    screen.blit(green2, (825, 525))
                    signall = 'green2'
                    screen.blit(green2, (500, 250))
                else:
                    signalr = 'green'
                    screen.blit(green, (825, 525))
                    signall = 'red'
                    screen.blit(red, (500, 250))
        elif sg == 'ud':
            if fl == 0:
                counter2 = 2
                fl = 1
            if counter2 >= 0:
                signall = 'yellow'
                signalr = 'yellow'
            else:
                signall = 'red'
                signalr = 'red'
            if signall == 'yellow':
                screen.blit(yellow, (500, 250))
                screen.blit(yellow, (825, 525))
            else:
                screen.blit(red, (500, 250))
                screen.blit(red, (825, 525))
            flb = 0
            for w in range(vehiclecount):
                if (xpos[w] in range(495, 765) and dirs[w] == 'left' and turns[w] == 's') or (xpos[w] in
range(780, 520, -1) and dirs[w] == 'right' and turns[w] == 's') or (xpos[w] >= 495 and turned[w] == 0
and dirs[w] == 'left' and turns[w] == 'l') or (xpos[w] <= 780 and turned[w] == 0 and dirs[w] == 'right'
and turns[w] == 'l') or (xpos[w] >= 495 and turned[w] == 0 and dirs[w] == 'left' and turns[w] == 'r') or
(xpos[w] <= 780 and turned[w] == 0 and dirs[w] == 'right' and turns[w] == 'r'):
                    flb = 1
                    break
            if flb == 1:
                signalu = 'red'
                signald = 'red'
                screen.blit(red, (500, 525))
                screen.blit(red, (825, 250))
            else:
                if sg2 == 'u':
                    if counter3 > 0:
                        signalu = 'green'
                        screen.blit(green, (825, 250))
                        signald = 'red'
                        screen.blit(red, (500, 525))
                    if counter3 <= 0:
                        if counter <= counter4:
```

```
                    signald = 'green'
                    screen.blit(green, (500, 525))
                    signalu = 'red'
                    screen.blit(red, (825, 250))
                else:
                    signalu = 'green2'
                    screen.blit(green2, (825, 250))
                    signald = 'green2'
                    screen.blit(green2, (500, 525))
            else:
                if counter3 > 0:
                    signald = 'green'
                    screen.blit(green, (500, 525))
                    signalu = 'red'
                    screen.blit(red, (825, 250))
                if counter3 <= 0:
                    if counter <= counter4:
                        signalu = 'green'
                        screen.blit(green, (825, 250))
                        signald = 'red'
                        screen.blit(red, (500, 525))
                    else:
                        signald = 'green2'
                        screen.blit(green2, (500, 525))
                        signalu = 'green2'
                        screen.blit(green2, (825, 250))

v = ''
x = 0
y = 0
dir = ''
for i in range(vehiclecount):
    if dirs[i] == 'left':
        if signall == 'green2' and turns[i] == 'r' and xpos[i] > 530 and turned2[i] == 0:
            turns[i] = 's'
        if(xpos[i] == 495):
            vehicleLeft += 1
        if xpos[i] == 11:
            vehiclelcount += 1
        if xpos[i] == 530:
            vehiclelcount -= 1
        if signall == 'red' or signall == 'yellow':
            if xpos[i] > 496:
                if turns[i] == 'l':
                    if xpos[i] >= 560 and xpos[i] <= 580:
                        if xpos[i] == 560:
                            if vehicles[i] == carl:
                                vehicles[i] = carll30
                            elif vehicles[i] == busl:
                                vehicles[i] = busll30
                            elif vehicles[i] == truckl:
```

```
                vehicles[i] = truckll30
            elif vehicles[i] == bikel:
                vehicles[i] = bikell30
            turned2[i] = 1
        elif xpos[i] == 570:
            if vehicles[i] == carll30:
                vehicles[i] = carll60
            elif vehicles[i] == busll30:
                vehicles[i] = busll60
            elif vehicles[i] == truckll30:
                vehicles[i] = truckll60
            elif vehicles[i] == bikell30:
                vehicles[i] = bikell60
        elif xpos[i] == 580:
            if vehicles[i] == carll60:
                vehicles[i] = carll90
            elif vehicles[i] == busll60:
                vehicles[i] = busll90
            elif vehicles[i] == truckll60:
                vehicles[i] = truckll90
            elif vehicles[i] == bikell60:
                vehicles[i] = bikell90
        xpos[i] += 1
        ypos[i] -= 2
    else:
        if xpos[i] > 580:
            if ypos[i] <= 310:
                turned[i] = 1
            ypos[i] -= 1
        else:
            xpos[i] += 1
elif turns[i] == 'r':
    if xpos[i] >= 675 and xpos[i] <= 695:
        if xpos[i] == 675:
            if vehicles[i] == carl:
                vehicles[i] = carlr30
            elif vehicles[i] == busl:
                vehicles[i] = buslr30
            elif vehicles[i] == bikel:
                vehicles[i] = bikelr30
            elif vehicles[i] == truckl:
                vehicles[i] = trucklr30
            turned2[i] = 1
        elif xpos[i] == 685:
            if vehicles[i] == carlr30:
                vehicles[i] = carlr60
            elif vehicles[i] == buslr30:
                vehicles[i] = buslr60
            elif vehicles[i] == bikelr30:
                vehicles[i] = bikelr60
            elif vehicles[i] == trucklr30:
```

```
                    vehicles[i] = trucklr60
                elif xpos[i] == 695:
                    if vehicles[i] == carlr60:
                        vehicles[i] = carlr90
                    elif vehicles[i] == buslr60:
                        vehicles[i] = buslr90
                    elif vehicles[i] == bikelr60:
                        vehicles[i] = bikelr90
                    elif vehicles[i] == trucklr60:
                        vehicles[i] = trucklr90
                xpos[i] += 1
                ypos[i] += 2
            else:
                if xpos[i] > 695:
                    if ypos[i] >= 500:
                        turned[i] = 1
                    ypos[i] += 1
                else:
                    xpos[i] += 1
        else:
            xpos[i] += 1
    else:
        k = 0
        for j in range(vehiclecount):
            if xpos[j] in range(435-100*k, 495-100*k) and j != i and ypos[j] == 385:
                k += 1
        if xpos[i] >= 435-100*k and xpos[i] <= 495:
            xpos[i] = xpos[i]
        else:
            xpos[i] += 1
else:
    if turns[i] == 's':
        xpos[i] += 1
    elif turns[i] == 'l':
        if xpos[i] >= 560 and xpos[i] <= 580:
            if xpos[i] == 560:
                if vehicles[i] == carl:
                    vehicles[i] = carll30
                elif vehicles[i] == busl:
                    vehicles[i] = busll30
                elif vehicles[i] == truckl:
                    vehicles[i] = truckll30
                elif vehicles[i] == bikel:
                    vehicles[i] = bikell30
                turned2[i] = 1
            elif xpos[i] == 570:
                if vehicles[i] == carll30:
                    vehicles[i] = carll60
                elif vehicles[i] == busll30:
                    vehicles[i] = busll60
                elif vehicles[i] == truckll30:
```

```
                vehicles[i] = truckll60
            elif vehicles[i] == bikell30:
                vehicles[i] = bikell60
        elif xpos[i] == 580:
            if vehicles[i] == carll60:
                vehicles[i] = carll90
            elif vehicles[i] == busll60:
                vehicles[i] = busll90
            elif vehicles[i] == truckll60:
                vehicles[i] = truckll90
            elif vehicles[i] == bikell60:
                vehicles[i] = bikell90
        xpos[i] += 1
        ypos[i] -= 2
    else:
        if xpos[i] > 580:
            if ypos[i] <= 310:
                turned[i] = 1
            ypos[i] -= 1
        else:
            xpos[i] += 1
else:
    if xpos[i] >= 675 and xpos[i] <= 695:
        if xpos[i] == 675:
            if vehicles[i] == carl:
                vehicles[i] = carlr30
            elif vehicles[i] == busl:
                vehicles[i] = buslr30
            elif vehicles[i] == bikel:
                vehicles[i] = bikelr30
            elif vehicles[i] == truckl:
                vehicles[i] = trucklr30
            turned2[i] = 1
        elif xpos[i] == 685:
            if vehicles[i] == carlr30:
                vehicles[i] = carlr60
            elif vehicles[i] == buslr30:
                vehicles[i] = buslr60
            elif vehicles[i] == bikelr30:
                vehicles[i] = bikelr60
            elif vehicles[i] == trucklr30:
                vehicles[i] = trucklr60
        elif xpos[i] == 695:
            if vehicles[i] == carlr60:
                vehicles[i] = carlr90
            elif vehicles[i] == buslr60:
                vehicles[i] = buslr90
            elif vehicles[i] == bikelr60:
                vehicles[i] = bikelr90
            elif vehicles[i] == trucklr60:
                vehicles[i] = trucklr90
```

```
                xpos[i] += 1
                ypos[i] += 2
            else:
                if xpos[i] > 695:
                    if ypos[i] >= 500:
                        turned[i] = 1
                    ypos[i] += 1
                else:
                    xpos[i] += 1
    elif dirs[i] == 'up':
        if signalu == 'green2' and turns[i] == 'r' and ypos[i] > 320 and turned2[i] == 0:
            turns[i] = 's'
        if(ypos[i] == 299):
            vehicleUp += 1
        if ypos[i] == 6:
            vehicleucount += 1
        if ypos[i] == 320:
            vehicleucount -= 1
        if signalu == 'red' or signalu == 'yellow':
            if ypos[i] > 301:
                if turns[i] == 'l':
                    if ypos[i] >= 360 and ypos[i] <= 380:
                        if ypos[i] == 360:
                            if vehicles[i] == caru:
                                vehicles[i] = carul30
                            elif vehicles[i] == busu:
                                vehicles[i] = busul30
                            elif vehicles[i] == trucku:
                                vehicles[i] = truckul30
                            elif vehicles[i] == bikeu:
                                vehicles[i] = bikeul30
                            turned2[i] = 1
                        elif ypos[i] == 370:
                            if vehicles[i] == carul30:
                                vehicles[i] = carul60
                            elif vehicles[i] == busul30:
                                vehicles[i] = busul60
                            elif vehicles[i] == truckul30:
                                vehicles[i] = truckul60
                            elif vehicles[i] == bikeul30:
                                vehicles[i] = bikeul60
                        elif ypos[i] == 380:
                            if vehicles[i] == carul60:
                                vehicles[i] = carul90
                            elif vehicles[i] == busul60:
                                vehicles[i] = busul90
                            elif vehicles[i] == truckul60:
                                vehicles[i] = truckul90
                            elif vehicles[i] == bikeul60:
                                vehicles[i] = bikeul90
                        ypos[i] += 1
```

```
            xpos[i] += 2
        else:
            if ypos[i] > 380:
                if xpos[i] >= 765:
                    turned[i] = 1
                xpos[i] += 1
            else:
                ypos[i] += 1
    elif turns[i] == 'r':
        if ypos[i] >= 430 and ypos[i] <= 450:
            if ypos[i] == 430:
                if vehicles[i] == caru:
                    vehicles[i] = carur30
                elif vehicles[i] == busu:
                    vehicles[i] = busur30
                elif vehicles[i] == trucku:
                    vehicles[i] = truckur30
                elif vehicles[i] == bikeu:
                    vehicles[i] = bikeur30
                turned2[i] = 1
            elif ypos[i] == 440:
                if vehicles[i] == carur30:
                    vehicles[i] = carur60
                elif vehicles[i] == busur30:
                    vehicles[i] = busur60
                elif vehicles[i] == truckur30:
                    vehicles[i] = truckur60
                elif vehicles[i] == bikeur30:
                    vehicles[i] = bikeur60
            elif ypos[i] == 450:
                if vehicles[i] == carur60:
                    vehicles[i] = carur90
                elif vehicles[i] == busur60:
                    vehicles[i] = busur90
                elif vehicles[i] == truckur60:
                    vehicles[i] = truckur90
                elif vehicles[i] == bikeur60:
                    vehicles[i] = bikeur90
            ypos[i] += 1
            xpos[i] -= 2
        else:
            if ypos[i] > 450:
                if xpos[i] <= 520:
                    turned[i] = 1
                xpos[i] -= 1
            else:
                ypos[i] += 1
    else:
        ypos[i] += 1
else:
    if xpos[i] == 765:
```

```
            k3 = 0
            for m in range(vehiclecount):
                if ypos[m] in range(240-100*k3, 300-100*k3) and m != i and xpos[m] == 765:
                    k3 += 1
            if ypos[i] >= 240-100*k3 and ypos[i] <= 300:
                ypos[i] = ypos[i]
            else:
                ypos[i] += 1
        else:
            k4 = 0
            for e in range(vehiclecount):
                if ypos[e] in range(240-100*k4, 300-100*k4) and e != i and xpos[e] == 700:
                    k4 += 1
            if ypos[i] >= 240-100*k4 and ypos[i] <= 300:
                ypos[i] = ypos[i]
            else:
                ypos[i] += 1
else:
    if turns[i] == 'l':
        if ypos[i] >= 360 and ypos[i] <= 380:
            if ypos[i] == 360:
                if vehicles[i] == caru:
                    vehicles[i] = carul30
                elif vehicles[i] == busu:
                    vehicles[i] = busul30
                elif vehicles[i] == trucku:
                    vehicles[i] = truckul30
                elif vehicles[i] == bikeu:
                    vehicles[i] = bikeul30
                turned2[i] = 1
            elif ypos[i] == 370:
                if vehicles[i] == carul30:
                    vehicles[i] = carul60
                elif vehicles[i] == busul30:
                    vehicles[i] = busul60
                elif vehicles[i] == truckul30:
                    vehicles[i] = truckul60
                elif vehicles[i] == bikeul30:
                    vehicles[i] = bikeul60
            elif ypos[i] == 380:
                if vehicles[i] == carul60:
                    vehicles[i] = carul90
                elif vehicles[i] == busul60:
                    vehicles[i] = busul90
                elif vehicles[i] == truckul60:
                    vehicles[i] = truckul90
                elif vehicles[i] == bikeul60:
                    vehicles[i] = bikeul90
            ypos[i] += 1
            xpos[i] += 2
        else:
```

```
                if ypos[i] > 380:
                    if xpos[i] >= 765:
                        turned[i] = 1
                    xpos[i] += 1
                else:
                    ypos[i] += 1
            elif turns[i] == 'r':
                if ypos[i] >= 430 and ypos[i] <= 450:
                    if ypos[i] == 430:
                        if vehicles[i] == caru:
                            vehicles[i] = carur30
                        elif vehicles[i] == busu:
                            vehicles[i] = busur30
                        elif vehicles[i] == trucku:
                            vehicles[i] = truckur30
                        elif vehicles[i] == bikeu:
                            vehicles[i] = bikeur30
                        turned2[i] = 1
                    elif ypos[i] == 440:
                        if vehicles[i] == carur30:
                            vehicles[i] = carur60
                        elif vehicles[i] == busur30:
                            vehicles[i] = busur60
                        elif vehicles[i] == truckur30:
                            vehicles[i] = truckur60
                        elif vehicles[i] == bikeur30:
                            vehicles[i] = bikeur60
                    elif ypos[i] == 450:
                        if vehicles[i] == carur60:
                            vehicles[i] = carur90
                        elif vehicles[i] == busur60:
                            vehicles[i] = busur90
                        elif vehicles[i] == truckur60:
                            vehicles[i] = truckur90
                        elif vehicles[i] == bikeur60:
                            vehicles[i] = bikeur90
                    ypos[i] += 1
                    xpos[i] -= 2
                else:
                    if ypos[i] > 450:
                        if xpos[i] <= 520:
                            turned[i] = 1
                        xpos[i] -= 1
                    else:
                        ypos[i] += 1
            else:
                ypos[i] += 1
        elif dirs[i] == 'right':
            if signalr == 'green2' and turns[i] == 'r' and xpos[i] < 765 and turned2[i] == 0:
                turns[i] = 's'
            if(xpos[i] == 797):
```

```
    vehicleRight += 1
if xpos[i] == 1349:
    vehiclercount += 1
if xpos[i] == 765:
    vehiclercount -= 1
if signalr == 'red' or signalr == 'yellow':
    if xpos[i] < 799:
        if turns[i] == 'r':
            if xpos[i] <= 660 and xpos[i] >= 640:
                if xpos[i] == 660:
                    turned2[i] = 1
                    if vehicles[i] == carr:
                        vehicles[i] = carrr30
                    elif vehicles[i] == busr:
                        vehicles[i] = busrr30
                    elif vehicles[i] == truckr:
                        vehicles[i] = truckrr30
                    elif vehicles[i] == biker:
                        vehicles[i] = bikerr30
                elif xpos[i] == 650:
                    if vehicles[i] == carrr30:
                        vehicles[i] = carrr60
                    elif vehicles[i] == busrr30:
                        vehicles[i] = busrr60
                    elif vehicles[i] == truckrr30:
                        vehicles[i] = truckrr60
                    elif vehicles[i] == bikerr30:
                        vehicles[i] = bikerr60
                elif xpos[i] == 640:
                    if vehicles[i] == carrr60:
                        vehicles[i] = carrr90
                    elif vehicles[i] == busrr60:
                        vehicles[i] = busrr90
                    elif vehicles[i] == truckrr60:
                        vehicles[i] = truckrr90
                    elif vehicles[i] == bikerr60:
                        vehicles[i] = bikerr90
                xpos[i] -= 1
                ypos[i] -= 2
            else:
                if xpos[i] < 640:
                    if ypos[i] <= 310:
                        turned[i] = 1
                    ypos[i] -= 1
                else:
                    xpos[i] -= 1
        elif turns[i] == 'l':
            if xpos[i] <= 780 and xpos[i] >= 760:
                if xpos[i] == 780:
                    turned2[i] = 1
                    if vehicles[i] == carr:
```

```
                    vehicles[i] = carrl30
                elif vehicles[i] == busr:
                    vehicles[i] = busrl30
                elif vehicles[i] == biker:
                    vehicles[i] = bikerl30
                elif vehicles[i] == truckr:
                    vehicles[i] = truckrl30
            elif xpos[i] == 770:
                if vehicles[i] == carrl30:
                    vehicles[i] = carrl60
                elif vehicles[i] == busrl30:
                    vehicles[i] = busrl60
                elif vehicles[i] == bikerl30:
                    vehicles[i] = bikerl60
                elif vehicles[i] == truckrl30:
                    vehicles[i] = truckrl60
            elif xpos[i] == 760:
                if vehicles[i] == carrl60:
                    vehicles[i] = carrl90
                elif vehicles[i] == busrl60:
                    vehicles[i] = busrl90
                elif vehicles[i] == bikerl60:
                    vehicles[i] = bikerl90
                elif vehicles[i] == truckrl60:
                    vehicles[i] = truckrl90
            xpos[i] -= 1
            ypos[i] += 2
        else:
            if xpos[i] < 760:
                if ypos[i] >= 500:
                    turned[i] = 1
                ypos[i] += 1
            else:
                xpos[i] -= 1
    else:
        xpos[i] -= 1
else:
    k2 = 0
    for l in range(vehiclecount):
        if xpos[l] in range(860+100*k2, 800+100*k2, -1) and l != i and ypos[l] == 445:
            k2 += 1
    if xpos[i] <= 860+100*k2 and xpos[i] >= 800:
        xpos[i] = xpos[i]
    else:
        xpos[i] -= 1
else:
    if turns[i] == 's':
        xpos[i] -= 1
    elif turns[i] == 'r':
        if xpos[i] <= 660 and xpos[i] >= 640:
            if xpos[i] == 660:
```

```
                turned2[i] = 1
                if vehicles[i] == carr:
                    vehicles[i] = carrr30
                elif vehicles[i] == busr:
                    vehicles[i] = busrr30
                elif vehicles[i] == truckr:
                    vehicles[i] = truckrr30
                elif vehicles[i] == biker:
                    vehicles[i] = bikerr30
            elif xpos[i] == 650:
                if vehicles[i] == carrr30:
                    vehicles[i] = carrr60
                elif vehicles[i] == busrr30:
                    vehicles[i] = busrr60
                elif vehicles[i] == truckrr30:
                    vehicles[i] = truckrr60
                elif vehicles[i] == bikerr30:
                    vehicles[i] = bikerr60
            elif xpos[i] == 640:
                if vehicles[i] == carrr60:
                    vehicles[i] = carrr90
                elif vehicles[i] == busrr60:
                    vehicles[i] = busrr90
                elif vehicles[i] == truckrr60:
                    vehicles[i] = truckrr90
                elif vehicles[i] == bikerr60:
                    vehicles[i] = bikerr90
            xpos[i] -= 1
            ypos[i] -= 2
        else:
            if xpos[i] < 640:
                if ypos[i] <= 310:
                    turned[i] = 1
                ypos[i] -= 1
            else:
                xpos[i] -= 1
    else:
        if xpos[i] <= 780 and xpos[i] >= 760:
            if xpos[i] == 780:
                turned2[i] = 1
                if vehicles[i] == carr:
                    vehicles[i] = carrl30
                elif vehicles[i] == busr:
                    vehicles[i] = busrl30
                elif vehicles[i] == biker:
                    vehicles[i] = bikerl30
                elif vehicles[i] == truckr:
                    vehicles[i] = truckrl30
            elif xpos[i] == 770:
                if vehicles[i] == carrl30:
                    vehicles[i] = carrl60
```

```
                elif vehicles[i] == busrl30:
                    vehicles[i] = busrl60
                elif vehicles[i] == bikerl30:
                    vehicles[i] = bikerl60
                elif vehicles[i] == truckrl30:
                    vehicles[i] = truckrl60
            elif xpos[i] == 760:
                if vehicles[i] == carrl60:
                    vehicles[i] = carrl90
                elif vehicles[i] == busrl60:
                    vehicles[i] = busrl90
                elif vehicles[i] == bikerl60:
                    vehicles[i] = bikerl90
                elif vehicles[i] == truckrl60:
                    vehicles[i] = truckrl90
                xpos[i] -= 1
                ypos[i] += 2
            else:
                if xpos[i] < 760:
                    if ypos[i] >= 500:
                        turned[i] = 1
                    ypos[i] += 1
                else:
                    xpos[i] -= 1
    else:
        if signald == 'green2' and turns[i] == 'r' and ypos[i] < 490 and turned2[i] == 0:
            turns[i] = 's'
        if(ypos[i] == 517):
            vehicleDown += 1
        if ypos[i] == 759:
            vehicledcount += 1
        if ypos[i] == 490:
            vehicledcount -= 1
        if signald == 'red' or signald == 'yellow':
            if ypos[i] < 519:
                if turns[i] == 'l':
                    if ypos[i] <= 470 and ypos[i] >= 450:
                        if ypos[i] == 470:
                            turned2[i] = 1
                            if vehicles[i] == card:
                                vehicles[i] = cardl30
                            elif vehicles[i] == busd:
                                vehicles[i] = busdl30
                            elif vehicles[i] == truckd:
                                vehicles[i] = truckdl30
                            elif vehicles[i] == biked:
                                vehicles[i] = bikedl30
                        elif ypos[i] == 460:
                            if vehicles[i] == cardl30:
                                vehicles[i] = cardl60
                            elif vehicles[i] == busdl30:
```

```
                vehicles[i] = busdl60
            elif vehicles[i] == truckdl30:
                vehicles[i] = truckdl60
            elif vehicles[i] == bikedl30:
                vehicles[i] = bikedl60
        elif ypos[i] == 450:
            if vehicles[i] == cardl60:
                vehicles[i] = cardl90
            elif vehicles[i] == busdl60:
                vehicles[i] = busdl90
            elif vehicles[i] == truckdl60:
                vehicles[i] = truckdl90
            elif vehicles[i] == bikedl60:
                vehicles[i] = bikedl90
        ypos[i] -= 1
        xpos[i] -= 2
    else:
        if ypos[i] < 450:
            if xpos[i] <= 520:
                turned[i] = 1
            xpos[i] -= 1
        else:
            ypos[i] -= 1
elif turns[i] == 'r':
    if ypos[i] <= 400 and ypos[i] >= 380:
        if ypos[i] == 400:
            turned2[i] = 1
            if vehicles[i] == card:
                vehicles[i] = cardr30
            elif vehicles[i] == busd:
                vehicles[i] = busdr30
            elif vehicles[i] == truckd:
                vehicles[i] = truckdr30
            elif vehicles[i] == biked:
                vehicles[i] = bikedr30
        elif ypos[i] == 390:
            if vehicles[i] == cardr30:
                vehicles[i] = cardr60
            elif vehicles[i] == busdr30:
                vehicles[i] = busdr60
            elif vehicles[i] == truckdr30:
                vehicles[i] = truckdr60
            elif vehicles[i] == bikedr30:
                vehicles[i] = bikedr60
        elif ypos[i] == 380:
            if vehicles[i] == cardr60:
                vehicles[i] = cardr90
            elif vehicles[i] == busdr60:
                vehicles[i] = busdr90
            elif vehicles[i] == truckdr60:
                vehicles[i] = truckdr90
```

```
                elif vehicles[i] == bikedr60:
                    vehicles[i] = bikedr90
            ypos[i] -= 1
            xpos[i] += 2
        else:
            if ypos[i] < 380:
                if xpos[i] >= 765:
                    turned[i] = 1
                xpos[i] += 1
            else:
                ypos[i] -= 1
    else:
        ypos[i] -= 1
else:
    if xpos[i] == 640:
        k5 = 0
        for p in range(vehiclecount):
            if ypos[p] in range(580+100*k5, 520+100*k5, -1) and p != i and xpos[p] == 640:
                k5 += 1
        if ypos[i] <= 580+100*k5 and ypos[i] >= 520:
            ypos[i] = ypos[i]
        else:
            ypos[i] -= 1
    else:
        k6 = 0
        for q in range(vehiclecount):
            if ypos[q] in range(580+100*k6, 520+100*k6, -1) and q != i and xpos[q] == 570:
                k6 += 1
        if ypos[i] <= 580+100*k6 and ypos[i] >= 520:
            ypos[i] = ypos[i]
        else:
            ypos[i] -= 1
else:
    if turns[i] == 'l':
        if ypos[i] <= 470 and ypos[i] >= 450:
            if ypos[i] == 470:
                turned2[i] = 1
                if vehicles[i] == card:
                    vehicles[i] = cardl30
                elif vehicles[i] == busd:
                    vehicles[i] = busdl30
                elif vehicles[i] == truckd:
                    vehicles[i] = truckdl30
                elif vehicles[i] == biked:
                    vehicles[i] = bikedl30
            elif ypos[i] == 460:
                if vehicles[i] == cardl30:
                    vehicles[i] = cardl60
                elif vehicles[i] == busdl30:
                    vehicles[i] = busdl60
                elif vehicles[i] == truckdl30:
```

```
                vehicles[i] = truckdl60
            elif vehicles[i] == bikedl30:
                vehicles[i] = bikedl60
        elif ypos[i] == 450:
            if vehicles[i] == cardl60:
                vehicles[i] = cardl90
            elif vehicles[i] == busdl60:
                vehicles[i] = busdl90
            elif vehicles[i] == truckdl60:
                vehicles[i] = truckdl90
            elif vehicles[i] == bikedl60:
                vehicles[i] = bikedl90
        ypos[i] -= 1
        xpos[i] -= 2
    else:
        if ypos[i] < 450:
            if xpos[i] <= 520:
                turned[i] = 1
            xpos[i] -= 1
        else:
            ypos[i] -= 1
elif turns[i] == 'r':
    if ypos[i] <= 400 and ypos[i] >= 380:
        if ypos[i] == 400:
            turned2[i] = 1
            if vehicles[i] == card:
                vehicles[i] = cardr30
            elif vehicles[i] == busd:
                vehicles[i] = busdr30
            elif vehicles[i] == truckd:
                vehicles[i] = truckdr30
            elif vehicles[i] == biked:
                vehicles[i] = bikedr30
        elif ypos[i] == 390:
            if vehicles[i] == cardr30:
                vehicles[i] = cardr60
            elif vehicles[i] == busdr30:
                vehicles[i] = busdr60
            elif vehicles[i] == truckdr30:
                vehicles[i] = truckdr60
            elif vehicles[i] == bikedr30:
                vehicles[i] = bikedr60
        elif ypos[i] == 380:
            if vehicles[i] == cardr60:
                vehicles[i] = cardr90
            elif vehicles[i] == busdr60:
                vehicles[i] = busdr90
            elif vehicles[i] == truckdr60:
                vehicles[i] = truckdr90
            elif vehicles[i] == bikedr60:
                vehicles[i] = bikedr90
```

```
                ypos[i] -= 1
                xpos[i] += 2
            else:
                if ypos[i] < 380:
                    if xpos[i] >= 765:
                        turned[i] = 1
                    xpos[i] += 1
                else:
                    ypos[i] -= 1
        else:
            ypos[i] -= 1
if vehiclegen <= 0:
    v, x, y, dir, turn = VehicleGenerate()
    if dir == 'left':
        fo = 0
        for o in range(vehiclecount):
            if xpos[o] in range(10, 80) and ypos[o] == 385:
                fo = 1
                break
        if fo == 0:
            vehicles.append(v)
            xpos.append(x)
            ypos.append(y)
            dirs.append(dir)
            vehiclecount += 1
            vehiclegen = 2
            turns.append(turn)
            turned.append(0)
            turned2.append(0)
    elif dir == 'right':
        fu = 0
        for u in range(vehiclecount):
            if xpos[u] in range(1350, 1280, -1) and ypos[u] == 445:
                fu = 1
                break
        if fu == 0:
            vehicles.append(v)
            xpos.append(x)
            ypos.append(y)
            dirs.append(dir)
            vehiclecount += 1
            vehiclegen = 2
            turns.append(turn)
            turned.append(0)
            turned2.append(0)
    elif dir == 'up':
        ft = 0
        for t in range(vehiclecount):
            if ypos[t] in range(5, 75) and (xpos[t] == 765 or xpos[t] == 700):
                ft = 1
                break
```

```
        if ft == 0:
            vehicles.append(v)
            xpos.append(x)
            ypos.append(y)
            dirs.append(dir)
            vehiclecount += 1
            vehiclegen = 2
            turns.append(turn)
            turned.append(0)
            turned2.append(0)
    else:
        fw = 0
        for w in range(vehiclecount):
            if ypos[w] in range(762, 680, -1) and (xpos[w] == 570 or xpos[w] == 640):
                fw = 1
                break
        if fw == 0:
            vehicles.append(v)
            xpos.append(x)
            ypos.append(y)
            dirs.append(dir)
            vehiclecount += 1
            vehiclegen = 2
            turns.append(turn)
            turned.append(0)
            turned2.append(0)
for i in range(vehiclecount):
    player(vehicles[i], xpos[i], ypos[i])
clock.tick(60)

pygame.display.update()
```

Traffic control using dynamic timers

Dynamic.py

```
import pygame
import random
import time
import timeit

# initialize pygame
pygame.init()
start = timeit.default_timer()
black = (0, 0, 0)
white = (255, 255, 255)
simtime = 0
font = pygame.font.Font(None, 30)
# create game screen
screen = pygame.display.set_mode((1400, 800))
background = pygame.image.load('images/mod_int.png')

# players
carl = pygame.image.load('images/left/car.png')
carll30 = pygame.image.load('images/left/carl30.png')
carll60 = pygame.image.load('images/left/carl60.png')
carll90 = pygame.image.load('images/left/carl90.png')
carlr30 = pygame.image.load('images/left/carr30.png')
carlr60 = pygame.image.load('images/left/carr60.png')
carlr90 = pygame.image.load('images/left/carr90.png')
carr = pygame.image.load('images/right/car.png')
carrl30 = pygame.image.load('images/right/carl30.png')
carrl60 = pygame.image.load('images/right/carl60.png')
carrl90 = pygame.image.load('images/right/carl90.png')
carrr30 = pygame.image.load('images/right/carr30.png')
carrr60 = pygame.image.load('images/right/carr60.png')
carrr90 = pygame.image.load('images/right/carr90.png')
caru = pygame.image.load('images/up/car.png')
carul30 = pygame.image.load('images/up/carl30.png')
carul60 = pygame.image.load('images/up/carl60.png')
carul90 = pygame.image.load('images/up/carl90.png')
carur30 = pygame.image.load('images/up/carr30.png')
carur60 = pygame.image.load('images/up/carr60.png')
carur90 = pygame.image.load('images/up/carr90.png')
card = pygame.image.load('images/down/car.png')
cardl30 = pygame.image.load('images/down/carl30.png')
cardl60 = pygame.image.load('images/down/carl60.png')
cardl90 = pygame.image.load('images/down/carl90.png')
cardr30 = pygame.image.load('images/down/carr30.png')
cardr60 = pygame.image.load('images/down/carr60.png')
cardr90 = pygame.image.load('images/down/carr90.png')
busl = pygame.image.load('images/left/bus.png')
busll30 = pygame.image.load('images/left/busl30.png')
busll60 = pygame.image.load('images/left/busl60.png')
```

```
busll90 = pygame.image.load('images/left/busl90.png')
buslr30 = pygame.image.load('images/left/busr30.png')
buslr60 = pygame.image.load('images/left/busr60.png')
buslr90 = pygame.image.load('images/left/busr90.png')
busr = pygame.image.load('images/right/bus.png')
busrl30 = pygame.image.load('images/right/busl30.png')
busrl60 = pygame.image.load('images/right/busl60.png')
busrl90 = pygame.image.load('images/right/busl90.png')
busrr30 = pygame.image.load('images/right/busr30.png')
busrr60 = pygame.image.load('images/right/busr60.png')
busrr90 = pygame.image.load('images/right/busr90.png')
busd = pygame.image.load('images/down/bus.png')
busdl30 = pygame.image.load('images/down/busl30.png')
busdl60 = pygame.image.load('images/down/busl60.png')
busdl90 = pygame.image.load('images/down/busl90.png')
busdr30 = pygame.image.load('images/down/busr30.png')
busdr60 = pygame.image.load('images/down/busr60.png')
busdr90 = pygame.image.load('images/down/busr90.png')
busu = pygame.image.load('images/up/bus.png')
busul30 = pygame.image.load('images/up/busl30.png')
busul60 = pygame.image.load('images/up/busl60.png')
busul90 = pygame.image.load('images/up/busl90.png')
busur30 = pygame.image.load('images/up/busr30.png')
busur60 = pygame.image.load('images/up/busr60.png')
busur90 = pygame.image.load('images/up/busr90.png')
truckl = pygame.image.load('images/left/truck.png')
truckll30 = pygame.image.load('images/left/truckl30.png')
truckll60 = pygame.image.load('images/left/truckl60.png')
truckll90 = pygame.image.load('images/left/truckl90.png')
trucklr30 = pygame.image.load('images/left/truckr30.png')
trucklr60 = pygame.image.load('images/left/truckr60.png')
trucklr90 = pygame.image.load('images/left/truckr90.png')
truckr = pygame.image.load('images/right/truck.png')
truckrl30 = pygame.image.load('images/right/truckl30.png')
truckrl60 = pygame.image.load('images/right/truckl60.png')
truckrl90 = pygame.image.load('images/right/truckl90.png')
truckrr30 = pygame.image.load('images/right/truckr30.png')
truckrr60 = pygame.image.load('images/right/truckr60.png')
truckrr90 = pygame.image.load('images/right/truckr90.png')
truckd = pygame.image.load('images/down/truck.png')
truckdl30 = pygame.image.load('images/down/truckl30.png')
truckdl60 = pygame.image.load('images/down/truckl60.png')
truckdl90 = pygame.image.load('images/down/truckl90.png')
truckdr30 = pygame.image.load('images/down/truckr30.png')
truckdr60 = pygame.image.load('images/down/truckr60.png')
truckdr90 = pygame.image.load('images/down/truckr90.png')
trucku = pygame.image.load('images/up/truck.png')
truckul30 = pygame.image.load('images/up/truckl30.png')
truckul60 = pygame.image.load('images/up/truckl60.png')
truckul90 = pygame.image.load('images/up/truckl90.png')
truckur30 = pygame.image.load('images/up/truckr30.png')
```

```
truckur60 = pygame.image.load('images/up/truckr60.png')
truckur90 = pygame.image.load('images/up/truckr90.png')
bikel = pygame.image.load('images/left/bike.png')
bikell30 = pygame.image.load('images/left/bikel30.png')
bikell60 = pygame.image.load('images/left/bikel60.png')
bikell90 = pygame.image.load('images/left/bikel90.png')
bikelr30 = pygame.image.load('images/left/biker30.png')
bikelr60 = pygame.image.load('images/left/biker60.png')
bikelr90 = pygame.image.load('images/left/biker90.png')
biker = pygame.image.load('images/right/bike.png')
bikerl30 = pygame.image.load('images/right/bikel30.png')
bikerl60 = pygame.image.load('images/right/bikel60.png')
bikerl90 = pygame.image.load('images/right/bikel90.png')
bikerr30 = pygame.image.load('images/right/biker30.png')
bikerr60 = pygame.image.load('images/right/biker60.png')
bikerr90 = pygame.image.load('images/right/biker90.png')
biked = pygame.image.load('images/down/bike.png')
bikedl30 = pygame.image.load('images/down/bikel30.png')
bikedl60 = pygame.image.load('images/down/bikel60.png')
bikedl90 = pygame.image.load('images/down/bikel90.png')
bikedr30 = pygame.image.load('images/down/biker30.png')
bikedr60 = pygame.image.load('images/down/biker60.png')
bikedr90 = pygame.image.load('images/down/biker90.png')
bikeu = pygame.image.load('images/up/bike.png')
bikeul30 = pygame.image.load('images/up/bikel30.png')
bikeul60 = pygame.image.load('images/up/bikel60.png')
bikeul90 = pygame.image.load('images/up/bikel90.png')
bikeur30 = pygame.image.load('images/up/biker30.png')
bikeur60 = pygame.image.load('images/up/biker60.png')
bikeur90 = pygame.image.load('images/up/biker90.png')
green = pygame.image.load('images/signals/green.png')
green2 = pygame.image.load('images/signals/green2.png')
red = pygame.image.load('images/signals/red.png')
yellow = pygame.image.load('images/signals/yellow.png')


# currentsignals
signall = ""
signalr = ""
signalu = ""
signald = ""


def player(vehicle, x, y):
    screen.blit(vehicle, (x, y))


# generating vehciles
def VehicleGenerate():
    dir = ''
    vehicle = ''
```

```
x = 0
y = 0
turn = ''
num1 = random.randint(0, 1001)
ger = [150, 500, 650, 1000]
if (num1 < ger[0]):
    dir = 'left'
elif (num1 > ger[0] and num1 < ger[1]):
    dir = 'up'
elif (num1 > ger[1] and num1 < ger[2]):
    dir = 'right'
elif (num1 > ger[2] and num1 < ger[3]):
    dir = 'down'
if dir == 'left':
    num2 = random.randint(0, 3)
    if num2 == 0:
        vehicle = carl
    elif num2 == 1:
        vehicle = bikel
    elif num2 == 2:
        vehicle = busl
    else:
        vehicle = truckl
    x = 10
    y = 385
    num3 = random.randint(0, 6)
    if num3 == 0:
        turn = 'l'
    elif num3 == 1:
        turn = 'r'
    else:
        turn = 's'
elif dir == 'up':
    num2 = random.randint(0, 3)
    if num2 == 0:
        vehicle = caru
    elif num2 == 1:
        vehicle = bikeu
    elif num2 == 2:
        vehicle = busu
    else:
        vehicle = trucku
    num3 = random.randint(0, 1)
    if num3 == 0:
        x = 765
        y = 5
        num4 = random.randint(0, 4)
        if num4 == 0:
            turn = 'l'
        else:
            turn = 's'
```

```
    else:
        x = 700
        y = 5
        num4 = random.randint(0, 4)
        if num4 == 0:
            turn = 'r'
        else:
            turn = 's'
elif dir == 'right':
    num2 = random.randint(0, 3)
    if num2 == 0:
        vehicle = carr
    elif num2 == 1:
        vehicle = biker
    elif num2 == 2:
        vehicle = busr
    else:
        vehicle = truckr
    x = 1350
    y = 445
    num3 = random.randint(0, 6)
    if num3 == 0:
        turn = 'l'
    elif num3 == 1:
        turn = 'r'
    else:
        turn = 's'
else:
    num2 = random.randint(0, 3)
    if num2 == 0:
        vehicle = card
    elif num2 == 1:
        vehicle = biked
    elif num2 == 2:
        vehicle = busd
    else:
        vehicle = truckd
    num3 = random.randint(0, 1)
    if num3 == 0:
        x = 570
        y = 760
        num4 = random.randint(0, 4)
        if num4 == 0:
            turn = 'l'
        else:
            turn = 's'
    else:
        x = 640
        y = 760
        num4 = random.randint(0, 4)
        if num4 == 0:
```

```
            turn = 'r'
        else:
            turn = 's'
    return vehicle, x, y, dir, turn


clock = pygame.time.Clock()
pygame.time.set_timer(pygame.USEREVENT, 1000)
running = True
vehicles = []
xpos = []
ypos = []
dirs = []
turns = []
turned = []
turned2 = []
ql = []
vehiclegen = 2
sg = 'lr'
sg2 = 'l'
a = 0
fl = 1
fu = 1
fr = 1
fd = 1
temp = 0
vehiclecount = 0
vehiclel = []
vehiclelcount = 0
vehiclercount = 0
vehicleucount = 0
vehicledcount = 0
vehicleLeft = 0
vehicleRight = 0
vehicleUp = 0
vehicleDown = 0
max = 40
min = 10
rmax = 5
rmin = 2
counter = 2*(vehiclecount+vehiclercount)+1
if counter < min:
    counter = min
if counter > max:
    counter = max
counter2 = 2
counter3 = (20/100)*counter
if counter3 < rmin:
    counter3 = rmin
if counter3 > rmax:
    counter3 = rmax
```

```
counter4 = (20/100)*counter
if counter4 < rmin:
    counter4 = rmin
if counter4 > rmax:
    counter4 = rmax
c1 = 0
c2 = 0
while running:
    flag = 0
    for event in pygame.event.get():
        if event.type == pygame.USEREVENT:
            simtime += 1
            vehiclegen -= 1
            counter -= 1
            counter2 -= 1
            counter3 -= 1
        if event.type == pygame.QUIT:
            vehiclePassed = vehicleDown + vehicleRight + vehicleUp + vehicleLeft
            stop = timeit.default_timer()
            timeRun = stop - start
            timeRun = int(timeRun)
            print("Dynamic Simulation:")
            print('Simulation time:', timeRun, "seconds")
            print('Total no of vehicles passed:', vehiclePassed)
            running = False
    screen.fill((0, 0, 0))
    screen.blit(background, (0, 0))
    simtimeText = font.render(
        ("Simulation Time: " + str(simtime)), True, black, white)
    screen.blit(simtimeText, (1160, 40))
    if counter < 0:
        if vehiclelcount+vehiclercount > vehicleucount+vehicledcount:
            c1 += 1
            if c1 > 4:
                sg = 'ud'
            else:
                sg = 'lr'
                c1 = 0
        else:
            c2 += 1
            if c2 > 4:
                sg = 'lr'
            else:
                sg = 'ud'
                c2 = 0
        if vehiclelcount+vehiclercount == 0:
            sg = 'ud'
        if vehicleucount+vehicledcount == 0:
            sg = 'lr'
        if sg == 'ud':
            fl = 0
```

```
            if vehicledcount > vehicleucount:
                sg2 = 'd'
            else:
                sg2 = 'u'
            counter = 2*(vehicledcount+vehicleucount)+1
            if counter < min:
                counter = min
            if counter > max:
                counter = max
            counter3 = (20/100)*counter
            if counter3 < rmin:
                counter3 = rmin
            if counter3 > rmax:
                counter3 = rmax
            counter4 = (20/100)*counter
            if counter4 < rmin:
                counter4 = rmin
            if counter4 > rmax:
                counter4 = rmax
        elif sg == 'lr':
            fd = 0
            if vehiclelcount > vehiclercount:
                sg2 = 'l'
            else:
                sg2 = 'r'
            counter = 2*(vehiclelcount+vehiclercount)+1
            if counter < min:
                counter = min
            if counter > max:
                counter = max
            counter3 = (20/100)*counter
            if counter3 < rmin:
                counter3 = rmin
            if counter3 > rmax:
                counter3 = rmax
            counter4 = (20/100)*counter
            if counter4 < rmin:
                counter4 = rmin
            if counter4 > rmax:
                counter4 = rmax
    if counter >= 0 or flag == 1:
        if sg == 'lr':
            if fd == 0:
                counter2 = 2
                fd = 1
            if counter2 >= 0:
                signald = 'yellow'
                signalu = 'yellow'
            else:
                signald = 'red'
                signalu = 'red'
```

```
        if signald == 'yellow' or signalu == 'yellow':
            screen.blit(yellow, (500, 525))
            screen.blit(yellow, (825, 250))
        else:
            screen.blit(red, (500, 525))
            screen.blit(red, (825, 250))
        fdb = 0
        for m in range(vehiclecount):
            if (ypos[m] in range(490, 310, -1) and dirs[m] == 'down' and turns[m] == 's') or (ypos[m] in
range(310, 500) and dirs[m] == 'up' and turns[m] == 's') or (ypos[m] <= 490 and turned[m] == 0 and
dirs[m] == 'down' and turns[m] == 'r') or (ypos[m] >= 310 and turned[m] == 0 and dirs[m] == 'up' and
turns[m] == 'l') or (ypos[m] <= 490 and turned[m] == 0 and dirs[m] == 'down' and turns[m] == 'l') or
(ypos[m] >= 310 and turned[m] == 0 and dirs[m] == 'up' and turns[m] == 'r'):
                fdb = 1
                break
        if fdb == 1:
            signall = 'red'
            signalr = 'red'
            screen.blit(red, (825, 525))
            screen.blit(red, (500, 250))
        else:
            if sg2 == 'l':
                if counter3 > 0:
                    signall = 'green'
                    screen.blit(green, (500, 250))
                    signalr = 'red'
                    screen.blit(red, (825, 525))
                if counter3 <= 0:
                    if counter <= counter4:
                        signalr = 'green'
                        screen.blit(green, (825, 525))
                        signall = 'red'
                        screen.blit(red, (500, 250))
                    else:
                        signall = 'green2'
                        screen.blit(green2, (500, 250))
                        signalr = 'green2'
                        screen.blit(green2, (825, 525))
            else:
                if counter3 > 0:
                    signalr = 'green'
                    screen.blit(green, (825, 525))
                    signall = 'red'
                    screen.blit(red, (500, 250))
                if counter3 <= 0:
                    if counter <= counter4:
                        signall = 'green'
                        screen.blit(green, (500, 250))
                        signalr = 'red'
                        screen.blit(red, (825, 525))
                    else:
```

```
                    frb = 0
                    for w in range(vehiclecount):
                        if (xpos[w] <= 780 and turned[w] == 0 and dirs[w] == 'right' and turns[w] == 'r'):
                            frb = 1
                            break
                    if frb == 0:
                        signalr = 'green2'
                        screen.blit(green2, (825, 525))
                        signall = 'green2'
                        screen.blit(green2, (500, 250))
                    else:
                        signalr = 'green'
                        screen.blit(green, (825, 525))
                        signall = 'red'
                        screen.blit(red, (500, 250))
        elif sg == 'ud':
            if fl == 0:
                counter2 = 2
                fl = 1
            if counter2 >= 0:
                signall = 'yellow'
                signalr = 'yellow'
            else:
                signall = 'red'
                signalr = 'red'
            if signall == 'yellow':
                screen.blit(yellow, (500, 250))
                screen.blit(yellow, (825, 525))
            else:
                screen.blit(red, (500, 250))
                screen.blit(red, (825, 525))
            flb = 0
            for w in range(vehiclecount):
                if (xpos[w] in range(495, 765) and dirs[w] == 'left' and turns[w] == 's') or (xpos[w] in
range(780, 520, -1) and dirs[w] == 'right' and turns[w] == 's') or (xpos[w] >= 495 and turned[w] == 0
and dirs[w] == 'left' and turns[w] == 'l') or (xpos[w] <= 780 and turned[w] == 0 and dirs[w] == 'right'
and turns[w] == 'l') or (xpos[w] >= 495 and turned[w] == 0 and dirs[w] == 'left' and turns[w] == 'r') or
(xpos[w] <= 780 and turned[w] == 0 and dirs[w] == 'right' and turns[w] == 'r'):
                    flb = 1
                    break
            if flb == 1:
                signalu = 'red'
                signald = 'red'
                screen.blit(red, (500, 525))
                screen.blit(red, (825, 250))
            else:
                if sg2 == 'u':
                    if counter3 > 0:
                        signalu = 'green'
                        screen.blit(green, (825, 250))
                        signald = 'red'
```

```
                screen.blit(red, (500, 525))
            if counter3 <= 0:
                if counter <= counter4:
                    signald = 'green'
                    screen.blit(green, (500, 525))
                    signalu = 'red'
                    screen.blit(red, (825, 250))
                else:
                    signalu = 'green2'
                    screen.blit(green2, (825, 250))
                    signald = 'green2'
                    screen.blit(green2, (500, 525))
        else:
            if counter3 > 0:
                signald = 'green'
                screen.blit(green, (500, 525))
                signalu = 'red'
                screen.blit(red, (825, 250))
            if counter3 <= 0:
                if counter <= counter4:
                    signalu = 'green'
                    screen.blit(green, (825, 250))
                    signald = 'red'
                    screen.blit(red, (500, 525))
                else:
                    signald = 'green2'
                    screen.blit(green2, (500, 525))
                    signalu = 'green2'
                    screen.blit(green2, (825, 250))

v = ''
x = 0
y = 0
dir = ''
for i in range(vehiclecount):
    if dirs[i] == 'left':
        if signall == 'green2' and turns[i] == 'r' and xpos[i] > 530 and turned2[i] == 0:
            turns[i] = 's'
        if(xpos[i] == 495):
            vehicleLeft += 1
        if xpos[i] == 11:
            vehiclelcount += 1
        if xpos[i] == 530:
            vehiclelcount -= 1
        if signall == 'red' or signall == 'yellow':
            if xpos[i] > 496:
                if turns[i] == 'l':
                    if xpos[i] >= 560 and xpos[i] <= 580:
                        if xpos[i] == 560:
                            if vehicles[i] == carl:
                                vehicles[i] = carll30
```

```
                elif vehicles[i] == busl:
                    vehicles[i] = busll30
                elif vehicles[i] == truckl:
                    vehicles[i] = truckll30
                elif vehicles[i] == bikel:
                    vehicles[i] = bikell30
                turned2[i] = 1
            elif xpos[i] == 570:
                if vehicles[i] == carll30:
                    vehicles[i] = carll60
                elif vehicles[i] == busll30:
                    vehicles[i] = busll60
                elif vehicles[i] == truckll30:
                    vehicles[i] = truckll60
                elif vehicles[i] == bikell30:
                    vehicles[i] = bikell60
            elif xpos[i] == 580:
                if vehicles[i] == carll60:
                    vehicles[i] = carll90
                elif vehicles[i] == busll60:
                    vehicles[i] = busll90
                elif vehicles[i] == truckll60:
                    vehicles[i] = truckll90
                elif vehicles[i] == bikell60:
                    vehicles[i] = bikell90
            xpos[i] += 1
            ypos[i] -= 2
        else:
            if xpos[i] > 580:
                if ypos[i] <= 310:
                    turned[i] = 1
                ypos[i] -= 1
            else:
                xpos[i] += 1
    elif turns[i] == 'r':
        if xpos[i] >= 675 and xpos[i] <= 695:
            if xpos[i] == 675:
                if vehicles[i] == carl:
                    vehicles[i] = carlr30
                elif vehicles[i] == busl:
                    vehicles[i] = buslr30
                elif vehicles[i] == bikel:
                    vehicles[i] = bikelr30
                elif vehicles[i] == truckl:
                    vehicles[i] = trucklr30
                turned2[i] = 1
            elif xpos[i] == 685:
                if vehicles[i] == carlr30:
                    vehicles[i] = carlr60
                elif vehicles[i] == buslr30:
                    vehicles[i] = buslr60
```

```
                elif vehicles[i] == bikelr30:
                    vehicles[i] = bikelr60
                elif vehicles[i] == trucklr30:
                    vehicles[i] = trucklr60
            elif xpos[i] == 695:
                if vehicles[i] == carlr60:
                    vehicles[i] = carlr90
                elif vehicles[i] == buslr60:
                    vehicles[i] = buslr90
                elif vehicles[i] == bikelr60:
                    vehicles[i] = bikelr90
                elif vehicles[i] == trucklr60:
                    vehicles[i] = trucklr90
                xpos[i] += 1
                ypos[i] += 2
            else:
                if xpos[i] > 695:
                    if ypos[i] >= 500:
                        turned[i] = 1
                    ypos[i] += 1
                else:
                    xpos[i] += 1
        else:
            xpos[i] += 1
    else:
        k = 0
        for j in range(vehiclecount):
            if xpos[j] in range(435-100*k, 495-100*k) and j != i and ypos[j] == 385:
                k += 1
        if xpos[i] >= 435-100*k and xpos[i] <= 495:
            xpos[i] = xpos[i]
        else:
            xpos[i] += 1
else:
    if turns[i] == 's':
        xpos[i] += 1
    elif turns[i] == 'l':
        if xpos[i] >= 560 and xpos[i] <= 580:
            if xpos[i] == 560:
                if vehicles[i] == carl:
                    vehicles[i] = carll30
                elif vehicles[i] == busl:
                    vehicles[i] = busll30
                elif vehicles[i] == truckl:
                    vehicles[i] = truckll30
                elif vehicles[i] == bikel:
                    vehicles[i] = bikell30
                turned2[i] = 1
            elif xpos[i] == 570:
                if vehicles[i] == carll30:
                    vehicles[i] = carll60
```

```
            elif vehicles[i] == busll30:
                vehicles[i] = busll60
            elif vehicles[i] == truckll30:
                vehicles[i] = truckll60
            elif vehicles[i] == bikell30:
                vehicles[i] = bikell60
        elif xpos[i] == 580:
            if vehicles[i] == carll60:
                vehicles[i] = carll90
            elif vehicles[i] == busll60:
                vehicles[i] = busll90
            elif vehicles[i] == truckll60:
                vehicles[i] = truckll90
            elif vehicles[i] == bikell60:
                vehicles[i] = bikell90
        xpos[i] += 1
        ypos[i] -= 2
    else:
        if xpos[i] > 580:
            if ypos[i] <= 310:
                turned[i] = 1
            ypos[i] -= 1
        else:
            xpos[i] += 1
else:
    if xpos[i] >= 675 and xpos[i] <= 695:
        if xpos[i] == 675:
            if vehicles[i] == carl:
                vehicles[i] = carlr30
            elif vehicles[i] == busl:
                vehicles[i] = buslr30
            elif vehicles[i] == bikel:
                vehicles[i] = bikelr30
            elif vehicles[i] == truckl:
                vehicles[i] = trucklr30
            turned2[i] = 1
        elif xpos[i] == 685:
            if vehicles[i] == carlr30:
                vehicles[i] = carlr60
            elif vehicles[i] == buslr30:
                vehicles[i] = buslr60
            elif vehicles[i] == bikelr30:
                vehicles[i] = bikelr60
            elif vehicles[i] == trucklr30:
                vehicles[i] = trucklr60
        elif xpos[i] == 695:
            if vehicles[i] == carlr60:
                vehicles[i] = carlr90
            elif vehicles[i] == buslr60:
                vehicles[i] = buslr90
            elif vehicles[i] == bikelr60:
```

```
                    vehicles[i] = bikelr90
                elif vehicles[i] == trucklr60:
                    vehicles[i] = trucklr90
            xpos[i] += 1
            ypos[i] += 2
        else:
            if xpos[i] > 695:
                if ypos[i] >= 500:
                    turned[i] = 1
                ypos[i] += 1
            else:
                xpos[i] += 1
elif dirs[i] == 'up':
    if signalu == 'green2' and turns[i] == 'r' and ypos[i] > 320 and turned2[i] == 0:
        turns[i] = 's'
    if(ypos[i] == 299):
        vehicleUp += 1
    if ypos[i] == 6:
        vehicleucount += 1
    if ypos[i] == 320:
        vehicleucount -= 1
    if signalu == 'red' or signalu == 'yellow':
        if ypos[i] > 301:
            if turns[i] == 'l':
                if ypos[i] >= 360 and ypos[i] <= 380:
                    if ypos[i] == 360:
                        if vehicles[i] == caru:
                            vehicles[i] = carul30
                        elif vehicles[i] == busu:
                            vehicles[i] = busul30
                        elif vehicles[i] == trucku:
                            vehicles[i] = truckul30
                        elif vehicles[i] == bikeu:
                            vehicles[i] = bikeul30
                        turned2[i] = 1
                    elif ypos[i] == 370:
                        if vehicles[i] == carul30:
                            vehicles[i] = carul60
                        elif vehicles[i] == busul30:
                            vehicles[i] = busul60
                        elif vehicles[i] == truckul30:
                            vehicles[i] = truckul60
                        elif vehicles[i] == bikeul30:
                            vehicles[i] = bikeul60
                    elif ypos[i] == 380:
                        if vehicles[i] == carul60:
                            vehicles[i] = carul90
                        elif vehicles[i] == busul60:
                            vehicles[i] = busul90
                        elif vehicles[i] == truckul60:
                            vehicles[i] = truckul90
```

```
            elif vehicles[i] == bikeul60:
                vehicles[i] = bikeul90
        ypos[i] += 1
        xpos[i] += 2
    else:
        if ypos[i] > 380:
            if xpos[i] >= 765:
                turned[i] = 1
            xpos[i] += 1
        else:
            ypos[i] += 1
elif turns[i] == 'r':
    if ypos[i] >= 430 and ypos[i] <= 450:
        if ypos[i] == 430:
            if vehicles[i] == caru:
                vehicles[i] = carur30
            elif vehicles[i] == busu:
                vehicles[i] = busur30
            elif vehicles[i] == trucku:
                vehicles[i] = truckur30
            elif vehicles[i] == bikeu:
                vehicles[i] = bikeur30
            turned2[i] = 1
        elif ypos[i] == 440:
            if vehicles[i] == carur30:
                vehicles[i] = carur60
            elif vehicles[i] == busur30:
                vehicles[i] = busur60
            elif vehicles[i] == truckur30:
                vehicles[i] = truckur60
            elif vehicles[i] == bikeur30:
                vehicles[i] = bikeur60
        elif ypos[i] == 450:
            if vehicles[i] == carur60:
                vehicles[i] = carur90
            elif vehicles[i] == busur60:
                vehicles[i] = busur90
            elif vehicles[i] == truckur60:
                vehicles[i] = truckur90
            elif vehicles[i] == bikeur60:
                vehicles[i] = bikeur90
        ypos[i] += 1
        xpos[i] -= 2
    else:
        if ypos[i] > 450:
            if xpos[i] <= 520:
                turned[i] = 1
            xpos[i] -= 1
        else:
            ypos[i] += 1
else:
```

```
            ypos[i] += 1
        else:
            if xpos[i] == 765:
                k3 = 0
                for m in range(vehiclecount):
                    if ypos[m] in range(240-100*k3, 300-100*k3) and m != i and xpos[m] == 765:
                        k3 += 1
                if ypos[i] >= 240-100*k3 and ypos[i] <= 300:
                    ypos[i] = ypos[i]
                else:
                    ypos[i] += 1
            else:
                k4 = 0
                for e in range(vehiclecount):
                    if ypos[e] in range(240-100*k4, 300-100*k4) and e != i and xpos[e] == 700:
                        k4 += 1
                if ypos[i] >= 240-100*k4 and ypos[i] <= 300:
                    ypos[i] = ypos[i]
                else:
                    ypos[i] += 1
    else:
        if turns[i] == 'l':
            if ypos[i] >= 360 and ypos[i] <= 380:
                if ypos[i] == 360:
                    if vehicles[i] == caru:
                        vehicles[i] = carul30
                    elif vehicles[i] == busu:
                        vehicles[i] = busul30
                    elif vehicles[i] == trucku:
                        vehicles[i] = truckul30
                    elif vehicles[i] == bikeu:
                        vehicles[i] = bikeul30
                    turned2[i] = 1
                elif ypos[i] == 370:
                    if vehicles[i] == carul30:
                        vehicles[i] = carul60
                    elif vehicles[i] == busul30:
                        vehicles[i] = busul60
                    elif vehicles[i] == truckul30:
                        vehicles[i] = truckul60
                    elif vehicles[i] == bikeul30:
                        vehicles[i] = bikeul60
                elif ypos[i] == 380:
                    if vehicles[i] == carul60:
                        vehicles[i] = carul90
                    elif vehicles[i] == busul60:
                        vehicles[i] = busul90
                    elif vehicles[i] == truckul60:
                        vehicles[i] = truckul90
                    elif vehicles[i] == bikeul60:
                        vehicles[i] = bikeul90
```

```
                ypos[i] += 1
                xpos[i] += 2
            else:
                if ypos[i] > 380:
                    if xpos[i] >= 765:
                        turned[i] = 1
                    xpos[i] += 1
                else:
                    ypos[i] += 1
        elif turns[i] == 'r':
            if ypos[i] >= 430 and ypos[i] <= 450:
                if ypos[i] == 430:
                    if vehicles[i] == caru:
                        vehicles[i] = carur30
                    elif vehicles[i] == busu:
                        vehicles[i] = busur30
                    elif vehicles[i] == trucku:
                        vehicles[i] = truckur30
                    elif vehicles[i] == bikeu:
                        vehicles[i] = bikeur30
                    turned2[i] = 1
                elif ypos[i] == 440:
                    if vehicles[i] == carur30:
                        vehicles[i] = carur60
                    elif vehicles[i] == busur30:
                        vehicles[i] = busur60
                    elif vehicles[i] == truckur30:
                        vehicles[i] = truckur60
                    elif vehicles[i] == bikeur30:
                        vehicles[i] = bikeur60
                elif ypos[i] == 450:
                    if vehicles[i] == carur60:
                        vehicles[i] = carur90
                    elif vehicles[i] == busur60:
                        vehicles[i] = busur90
                    elif vehicles[i] == truckur60:
                        vehicles[i] = truckur90
                    elif vehicles[i] == bikeur60:
                        vehicles[i] = bikeur90
                ypos[i] += 1
                xpos[i] -= 2
            else:
                if ypos[i] > 450:
                    if xpos[i] <= 520:
                        turned[i] = 1
                    xpos[i] -= 1
                else:
                    ypos[i] += 1
        else:
            ypos[i] += 1
    elif dirs[i] == 'right':
```

```
if signalr == 'green2' and turns[i] == 'r' and xpos[i] < 765 and turned2[i] == 0:
    turns[i] = 's'
if(xpos[i] == 797):
    vehicleRight += 1
if xpos[i] == 1349:
    vehiclercount += 1
if xpos[i] == 765:
    vehiclercount -= 1
if signalr == 'red' or signalr == 'yellow':
    if xpos[i] < 799:
        if turns[i] == 'r':
            if xpos[i] <= 660 and xpos[i] >= 640:
                if xpos[i] == 660:
                    turned2[i] = 1
                    if vehicles[i] == carr:
                        vehicles[i] = carrr30
                    elif vehicles[i] == busr:
                        vehicles[i] = busrr30
                    elif vehicles[i] == truckr:
                        vehicles[i] = truckrr30
                    elif vehicles[i] == biker:
                        vehicles[i] = bikerr30
                elif xpos[i] == 650:
                    if vehicles[i] == carrr30:
                        vehicles[i] = carrr60
                    elif vehicles[i] == busrr30:
                        vehicles[i] = busrr60
                    elif vehicles[i] == truckrr30:
                        vehicles[i] = truckrr60
                    elif vehicles[i] == bikerr30:
                        vehicles[i] = bikerr60
                elif xpos[i] == 640:
                    if vehicles[i] == carrr60:
                        vehicles[i] = carrr90
                    elif vehicles[i] == busrr60:
                        vehicles[i] = busrr90
                    elif vehicles[i] == truckrr60:
                        vehicles[i] = truckrr90
                    elif vehicles[i] == bikerr60:
                        vehicles[i] = bikerr90
                xpos[i] -= 1
                ypos[i] -= 2
            else:
                if xpos[i] < 640:
                    if ypos[i] <= 310:
                        turned[i] = 1
                    ypos[i] -= 1
                else:
                    xpos[i] -= 1
        elif turns[i] == 'l':
            if xpos[i] <= 780 and xpos[i] >= 760:
```

```
                if xpos[i] == 780:
                    turned2[i] = 1
                    if vehicles[i] == carr:
                        vehicles[i] = carrl30
                    elif vehicles[i] == busr:
                        vehicles[i] = busrl30
                    elif vehicles[i] == biker:
                        vehicles[i] = bikerl30
                    elif vehicles[i] == truckr:
                        vehicles[i] = truckrl30
                elif xpos[i] == 770:
                    if vehicles[i] == carrl30:
                        vehicles[i] = carrl60
                    elif vehicles[i] == busrl30:
                        vehicles[i] = busrl60
                    elif vehicles[i] == bikerl30:
                        vehicles[i] = bikerl60
                    elif vehicles[i] == truckrl30:
                        vehicles[i] = truckrl60
                elif xpos[i] == 760:
                    if vehicles[i] == carrl60:
                        vehicles[i] = carrl90
                    elif vehicles[i] == busrl60:
                        vehicles[i] = busrl90
                    elif vehicles[i] == bikerl60:
                        vehicles[i] = bikerl90
                    elif vehicles[i] == truckrl60:
                        vehicles[i] = truckrl90
                xpos[i] -= 1
                ypos[i] += 2
            else:
                if xpos[i] < 760:
                    if ypos[i] >= 500:
                        turned[i] = 1
                    ypos[i] += 1
                else:
                    xpos[i] -= 1
        else:
            xpos[i] -= 1
    else:
        k2 = 0
        for l in range(vehiclecount):
            if xpos[l] in range(860+100*k2, 800+100*k2, -1) and l != i and ypos[l] == 445:
                k2 += 1
        if xpos[i] <= 860+100*k2 and xpos[i] >= 800:
            xpos[i] = xpos[i]
        else:
            xpos[i] -= 1
else:
    if turns[i] == 's':
        xpos[i] -= 1
```

```
elif turns[i] == 'r':
    if xpos[i] <= 660 and xpos[i] >= 640:
        if xpos[i] == 660:
            turned2[i] = 1
            if vehicles[i] == carr:
                vehicles[i] = carrr30
            elif vehicles[i] == busr:
                vehicles[i] = busrr30
            elif vehicles[i] == truckr:
                vehicles[i] = truckrr30
            elif vehicles[i] == biker:
                vehicles[i] = bikerr30
        elif xpos[i] == 650:
            if vehicles[i] == carrr30:
                vehicles[i] = carrr60
            elif vehicles[i] == busrr30:
                vehicles[i] = busrr60
            elif vehicles[i] == truckrr30:
                vehicles[i] = truckrr60
            elif vehicles[i] == bikerr30:
                vehicles[i] = bikerr60
        elif xpos[i] == 640:
            if vehicles[i] == carrr60:
                vehicles[i] = carrr90
            elif vehicles[i] == busrr60:
                vehicles[i] = busrr90
            elif vehicles[i] == truckrr60:
                vehicles[i] = truckrr90
            elif vehicles[i] == bikerr60:
                vehicles[i] = bikerr90
        xpos[i] -= 1
        ypos[i] -= 2
    else:
        if xpos[i] < 640:
            if ypos[i] <= 310:
                turned[i] = 1
            ypos[i] -= 1
        else:
            xpos[i] -= 1
else:
    if xpos[i] <= 780 and xpos[i] >= 760:
        if xpos[i] == 780:
            turned2[i] = 1
            if vehicles[i] == carr:
                vehicles[i] = carrl30
            elif vehicles[i] == busr:
                vehicles[i] = busrl30
            elif vehicles[i] == biker:
                vehicles[i] = bikerl30
            elif vehicles[i] == truckr:
                vehicles[i] = truckrl30
```

```
                elif xpos[i] == 770:
                    if vehicles[i] == carrl30:
                        vehicles[i] = carrl60
                    elif vehicles[i] == busrl30:
                        vehicles[i] = busrl60
                    elif vehicles[i] == bikerl30:
                        vehicles[i] = bikerl60
                    elif vehicles[i] == truckrl30:
                        vehicles[i] = truckrl60
                elif xpos[i] == 760:
                    if vehicles[i] == carrl60:
                        vehicles[i] = carrl90
                    elif vehicles[i] == busrl60:
                        vehicles[i] = busrl90
                    elif vehicles[i] == bikerl60:
                        vehicles[i] = bikerl90
                    elif vehicles[i] == truckrl60:
                        vehicles[i] = truckrl90
                xpos[i] -= 1
                ypos[i] += 2
            else:
                if xpos[i] < 760:
                    if ypos[i] >= 500:
                        turned[i] = 1
                    ypos[i] += 1
                else:
                    xpos[i] -= 1
    else:
        if signald == 'green2' and turns[i] == 'r' and ypos[i] < 490 and turned2[i] == 0:
            turns[i] = 's'
        if(ypos[i] == 517):
            vehicleDown += 1
        if ypos[i] == 759:
            vehicledcount += 1
        if ypos[i] == 490:
            vehicledcount -= 1
        if signald == 'red' or signald == 'yellow':
            if ypos[i] < 519:
                if turns[i] == 'l':
                    if ypos[i] <= 470 and ypos[i] >= 450:
                        if ypos[i] == 470:
                            turned2[i] = 1
                            if vehicles[i] == card:
                                vehicles[i] = cardl30
                            elif vehicles[i] == busd:
                                vehicles[i] = busdl30
                            elif vehicles[i] == truckd:
                                vehicles[i] = truckdl30
                            elif vehicles[i] == biked:
                                vehicles[i] = bikedl30
                        elif ypos[i] == 460:
```

```
            if vehicles[i] == cardl30:
                vehicles[i] = cardl60
            elif vehicles[i] == busdl30:
                vehicles[i] = busdl60
            elif vehicles[i] == truckdl30:
                vehicles[i] = truckdl60
            elif vehicles[i] == bikedl30:
                vehicles[i] = bikedl60
        elif ypos[i] == 450:
            if vehicles[i] == cardl60:
                vehicles[i] = cardl90
            elif vehicles[i] == busdl60:
                vehicles[i] = busdl90
            elif vehicles[i] == truckdl60:
                vehicles[i] = truckdl90
            elif vehicles[i] == bikedl60:
                vehicles[i] = bikedl90
        ypos[i] -= 1
        xpos[i] -= 2
    else:
        if ypos[i] < 450:
            if xpos[i] <= 520:
                turned[i] = 1
            xpos[i] -= 1
        else:
            ypos[i] -= 1
elif turns[i] == 'r':
    if ypos[i] <= 400 and ypos[i] >= 380:
        if ypos[i] == 400:
            turned2[i] = 1
            if vehicles[i] == card:
                vehicles[i] = cardr30
            elif vehicles[i] == busd:
                vehicles[i] = busdr30
            elif vehicles[i] == truckd:
                vehicles[i] = truckdr30
            elif vehicles[i] == biked:
                vehicles[i] = bikedr30
        elif ypos[i] == 390:
            if vehicles[i] == cardr30:
                vehicles[i] = cardr60
            elif vehicles[i] == busdr30:
                vehicles[i] = busdr60
            elif vehicles[i] == truckdr30:
                vehicles[i] = truckdr60
            elif vehicles[i] == bikedr30:
                vehicles[i] = bikedr60
        elif ypos[i] == 380:
            if vehicles[i] == cardr60:
                vehicles[i] = cardr90
            elif vehicles[i] == busdr60:
```

```
                    vehicles[i] = busdr90
                elif vehicles[i] == truckdr60:
                    vehicles[i] = truckdr90
                elif vehicles[i] == bikedr60:
                    vehicles[i] = bikedr90
            ypos[i] -= 1
            xpos[i] += 2
        else:
            if ypos[i] < 380:
                if xpos[i] >= 765:
                    turned[i] = 1
                xpos[i] += 1
            else:
                ypos[i] -= 1
    else:
        ypos[i] -= 1
else:
    if xpos[i] == 640:
        k5 = 0
        for p in range(vehiclecount):
            if ypos[p] in range(580+100*k5, 520+100*k5, -1) and p != i and xpos[p] == 640:
                k5 += 1
        if ypos[i] <= 580+100*k5 and ypos[i] >= 520:
            ypos[i] = ypos[i]
        else:
            ypos[i] -= 1
    else:
        k6 = 0
        for q in range(vehiclecount):
            if ypos[q] in range(580+100*k6, 520+100*k6, -1) and q != i and xpos[q] == 570:
                k6 += 1
        if ypos[i] <= 580+100*k6 and ypos[i] >= 520:
            ypos[i] = ypos[i]
        else:
            ypos[i] -= 1
else:
    if turns[i] == 'l':
        if ypos[i] <= 470 and ypos[i] >= 450:
            if ypos[i] == 470:
                turned2[i] = 1
                if vehicles[i] == card:
                    vehicles[i] = cardl30
                elif vehicles[i] == busd:
                    vehicles[i] = busdl30
                elif vehicles[i] == truckd:
                    vehicles[i] = truckdl30
                elif vehicles[i] == biked:
                    vehicles[i] = bikedl30
            elif ypos[i] == 460:
                if vehicles[i] == cardl30:
                    vehicles[i] = cardl60
```

```
            elif vehicles[i] == busdl30:
                vehicles[i] = busdl60
            elif vehicles[i] == truckdl30:
                vehicles[i] = truckdl60
            elif vehicles[i] == bikedl30:
                vehicles[i] = bikedl60
        elif ypos[i] == 450:
            if vehicles[i] == cardl60:
                vehicles[i] = cardl90
            elif vehicles[i] == busdl60:
                vehicles[i] = busdl90
            elif vehicles[i] == truckdl60:
                vehicles[i] = truckdl90
            elif vehicles[i] == bikedl60:
                vehicles[i] = bikedl90
        ypos[i] -= 1
        xpos[i] -= 2
    else:
        if ypos[i] < 450:
            if xpos[i] <= 520:
                turned[i] = 1
            xpos[i] -= 1
        else:
            ypos[i] -= 1
elif turns[i] == 'r':
    if ypos[i] <= 400 and ypos[i] >= 380:
        if ypos[i] == 400:
            turned2[i] = 1
            if vehicles[i] == card:
                vehicles[i] = cardr30
            elif vehicles[i] == busd:
                vehicles[i] = busdr30
            elif vehicles[i] == truckd:
                vehicles[i] = truckdr30
            elif vehicles[i] == biked:
                vehicles[i] = bikedr30
        elif ypos[i] == 390:
            if vehicles[i] == cardr30:
                vehicles[i] = cardr60
            elif vehicles[i] == busdr30:
                vehicles[i] = busdr60
            elif vehicles[i] == truckdr30:
                vehicles[i] = truckdr60
            elif vehicles[i] == bikedr30:
                vehicles[i] = bikedr60
        elif ypos[i] == 380:
            if vehicles[i] == cardr60:
                vehicles[i] = cardr90
            elif vehicles[i] == busdr60:
                vehicles[i] = busdr90
            elif vehicles[i] == truckdr60:
```

```
                        vehicles[i] = truckdr90
                    elif vehicles[i] == bikedr60:
                        vehicles[i] = bikedr90
                ypos[i] -= 1
                xpos[i] += 2
            else:
                if ypos[i] < 380:
                    if xpos[i] >= 765:
                        turned[i] = 1
                    xpos[i] += 1
                else:
                    ypos[i] -= 1
        else:
            ypos[i] -= 1
if vehiclegen <= 0:
    v, x, y, dir, turn = VehicleGenerate()
    if dir == 'left':
        fo = 0
        for o in range(vehiclecount):
            if xpos[o] in range(10, 80) and ypos[o] == 385:
                fo = 1
                break
        if fo == 0:
            vehicles.append(v)
            xpos.append(x)
            ypos.append(y)
            dirs.append(dir)
            vehiclecount += 1
            vehiclegen = 2
            turns.append(turn)
            turned.append(0)
            turned2.append(0)
    elif dir == 'right':
        fu = 0
        for u in range(vehiclecount):
            if xpos[u] in range(1350, 1280, -1) and ypos[u] == 445:
                fu = 1
                break
        if fu == 0:
            vehicles.append(v)
            xpos.append(x)
            ypos.append(y)
            dirs.append(dir)
            vehiclecount += 1
            vehiclegen = 2
            turns.append(turn)
            turned.append(0)
            turned2.append(0)
    elif dir == 'up':
        ft = 0
        for t in range(vehiclecount):
```

```
                if ypos[t] in range(5, 75) and (xpos[t] == 765 or xpos[t] == 700):
                    ft = 1
                    break
            if ft == 0:
                vehicles.append(v)
                xpos.append(x)
                ypos.append(y)
                dirs.append(dir)
                vehiclecount += 1
                vehiclegen = 2
                turns.append(turn)
                turned.append(0)
                turned2.append(0)
        else:
            fw = 0
            for w in range(vehiclecount):
                if ypos[w] in range(762, 680, -1) and (xpos[w] == 570 or xpos[w] == 640):
                    fw = 1
                    break
            if fw == 0:
                vehicles.append(v)
                xpos.append(x)
                ypos.append(y)
                dirs.append(dir)
                vehiclecount += 1
                vehiclegen = 2
                turns.append(turn)
                turned.append(0)
                turned2.append(0)
    for i in range(vehiclecount):
        player(vehicles[i], xpos[i], ypos[i])
    clock.tick(60)
pygame.display.update()
```

Appendix D
CO-PO and CO-PSO Mapping

CO-PO and CO-PSO Mapping

SI No.	Description	Blooms Taxonomy Level
CS451.1	Analyze a current topic of professional interest and present it before an audience	Knowledge(level 11) Analyze (level 4)
CS451.2	Identify an engineering problem, analyze it and propose a work plan to solve it	Evaluate(level5) Understand (level 2)

Course Outcomes

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	P10	P11	P12
CS451.1	-	3	-	2	-	-	1	-	3	3	2	3
CS451.2	3	3	3	2	1	-	-	3	3	3	-	1

CO-PO Mapping

	PSO1	PSO2	PSO3
CS451.1	3	-	1
CS451.2	3	3	2

CO-PSO Mapping

Mapping	Low/Medium/High	Justification
CS451.1–PO2	H	Problem analysis : I was able to identify, formulate, review research literature, and analyze problems with the traditional learning environment, reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
CS451.1–PO4	M	Conduct investigations of complex problems : I used research based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
CS451.1–PO7	L	Environment and sustainability : I understood the impact of the professional engineering solutions in societal and environmen-tal contexts, and demonstrated the knowledge of- and the need for- sustainable developments.
CS451.1–PO9	H	Individual: I was able to function effectively as an individual, in multi- disciplinary settings.

CS451.1–PO10	H	Communication : I was able to communicate effectively on complex Engineering activities with the Engineer- ing Community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
CS451.1–PO11	M	Project Management and finance: Demonstrated knowledge and understanding of the Engineering and management principles and apply these to one's own work, to manage projects and in multi-disciplinary environments.
CS451.1–PO12	H	Life-long learning : Recognized the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest contextof technological change.

CS451.1–PSO3	L	Professional Skills : Was able to apply the fundamentals of com- puter science to formulate competitive research proposals and to develop innovative products to meet the societal needs thereby evolving as an eminent researcher and entrepreneur.
CS451.2–PO1	H	Engineering Knowledge : Applied the knowledge of Mathematics, Science, Engineering fundamentals, and an Engineering discipline to the solution of complex engineering problems.
CS451.2–PO2	H	Problem analysis : I was able to identify, formulate, review research literature, and analyze complex Engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and Engineering sciences.
CS451.2–PO3	H	Design/Development of solutions: Designed solutions for complex Engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

CS451.2–PO4	M	Conduct investigations of complex problems: Used research based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
CS451.2–PO5	L	Modern Tool usage: Created, selected, and applied appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex Engineering activities with an understanding of the limitations.
CS451.2–PO8	H	Ethics: Applied ethical principles and commit to professional ethics and responsibilities and norms of the Engineering practice.
CS451.2–PO9	H	Individual: I was able to function effectively as an individual, and in multi- disciplinary settings.

CS451.2–PO10	H	Communication : Communicated effectively on complex Engineering activities with the Engineering Community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
CS451.2–PO12	L	Life-long learning: Recognized the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest contextof technological change.
CS451.2–PSO1	H	Computer Science Specific Skills: I was able to identify, analyze and design solutions for complex engineering problems in multi- disciplinary areas by under- standing the core principles and concepts of computer science.
CS451.2–PSO2	H	Programming and Software Development Skills: Acquired programming efficiency by designing algorithms and applying standard practices in software project development to deliver quality software products.

CS451.2–PSO3	M	Professional Skills : Applied the fundamentals of computer scienceto formulate competitive research proposals and to develop innovative products to meet the so- cietal needs thereby evolving as an eminent re- searcher and entrepreneur.

Justifications for CO-PO Mapping