UNIVERSITÉ DE BOURGOGNE

Software Engineering Project Report

TRAFFIC LIGHTS VIOLATION SYSTEM

By:

- Jaqueline Leal
- Muhammad Zain Amin
- Md. Imran Hossain

04 / December / 2023

Table of Contents

Abstract 3

1. Introduction 4

2. Objectives 5

3. Tools and Technologies Used 6

3.1 Python Programming Language 6

3.2 Pygame 6

3.3 Hardware Requirements 6

3.4 Software Requirements 6

4. Plan of Project Execution 7

4.1 Four-way traffic roads intersection with pedestrian crossings 7

4.2 Traffic Light Simulator 7

4.3 Pedestrian Light Simulator 8

4.4 Vehicle Movements 9

4.5 Simulation Timing 11

5. Design Specification 12

5.1 Generate Vehicle 12

5.2 Generate Pedestrians 13

6. Traffic Light Violation Detection 15

7. Pedestrian Light Violation Detection 17

8. Future Scope 19

Conclusions 19

References 20

Abstract

Traffic is increasing at a rapid pace on the roads proportional to the increase in population. The traffic problem is faced by all big cities in all the countries. As the traffic increases it causes violations both by vehicles and pedestrians on the roads. Due to the violations made by vehicles and pedestrians, the risk of accidents has been increasing. To counter this problem, we have proposed a traffic light and pedestrian light violations detection simulator program in python using the pygame library. Our project has the ability to detect the violations made by pedestrians and vehicles in real-time.

1. Introduction

People are using cars or other vehicles that basically use roads, for moving to their destinations; is increasing day by day. Current analysis of big cities shows that the number of cars on the roads is increasing tremendously and the available resources are limited. Proper handling of traffic seems to be the market demand. The traffic light problem is a well-known optimization problem for a decade or so. Traffic lights and pedestrian lights violations have caused serious accidents on the roads every year. According to a report by Road Safety in 2021, around 3,219 people were involved on accidents in roads, mainland in France and overseas. This figure, 8% lower than in 2019, is historically low (excluding the atypical year 2020). The upturn in travel in 2021 was tempered by a period of confinement in April, curfews throughout the first half of the year, and several periods when working from home was strongly recommended [1].

In 2021, there were 53,540 injury traffic accidents in mainland France. 2,944 people had road accidents within 30 days of their incident, including 414 pedestrians, 24 users of personal mobility devices, 227 cyclists, 96 moped riders, 572 motorcyclists, 1,414 motorists, 103 users of utility vehicles, 44 users of heavy goods vehicles [1].

2. Objectives

In this project, the main objective was to design a simple traffic light and pedestrian light-based controller system with violation detection and illustrate its operation by using pygame and python programming language.

1. Add a real time junction of four-way traffic
2. Simulate three signals for traffic control, i.e., Red, Green, Yellow
3. Simulate two signals for pedestrian control, i.e., Red and Green
4. Violation detection of traffic signals by vehicles
5. Violation detection of pedestrian signals by pedestrian

3. Tools and Technologies Used

3.1. *Python Programming Language*

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together [2]..

Why do we use Python?

- Easy to Read, Learn and Write.
- Improved Productivity.
- Interpreted Language.
- Dynamically Typed.
- Free and Open-Source.
- Vast Libraries Support.

3.2. ***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.

Pygame uses the Simple DirectMedia Layer (SDL) library, with the intention of allowing real-time computer game development without the low-level mechanics of the C programming language and its derivatives. 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 [3].

3.3. *Hardware Requirements*

- Processor: Intel i3 and above
- RAM: 4 GB
- Operating System: Windows

3.4. *Software Requirements*

- Python version 3.8.1 or above
- PyCharm

4. Plan of Project Execution

The basic components of our traffic light violation detection system are:

4.1. Four-way traffic roads Intersection with pedestrian crossings

The basic components of our traffic light simulator are Road1, Road2, Road3, and Road4 along with zebra crossings on all the roads as shown in Fig 1.

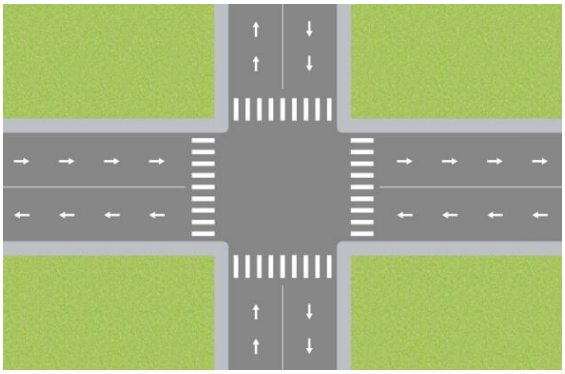

Figure 1. Four Way Traffic Roads Intersection

4.2. Traffic Light Simulator

Fig. 1 shows that there are four road intersections namely Roadd1, Road2, Road3, and Road4. Each intersection has three traffic signals: red, yellow, and green. Road 1 is the West one, Road 2 is the North one, Road 3 is the East one, and Road 4 is the South one and West W. Hence there are 12 traffic light signals in total. When the car(s) will reach close to any signal it will check the status of that signal. The status of the signal is red or green. If green cars continue their motion to reach their destination point else, it will stop at a red signal. We have also made some cars violate the traffic rules, in order to detect the violations made by the car. We have built a traffic signal class to simulate the operations of traffic control.

Code:

```
{
class TrafficSignal:
   def _init_(self, red, yellow, green):
      self.red = red
      self.yellow = yellow
```

self.green = green

self.signalText = ""

}

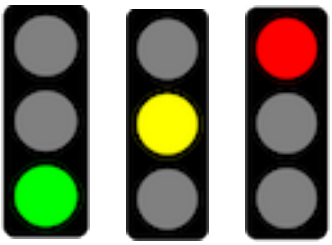

Figure 2. Traffic Light Simulations

4.3. Pedestrian Light Simulator

When the pedestrians reach close to any pedestrian signal it will check the status of that signal. The status of the signal is red or green. If it is green pedestrian will continue their motion to pass through the zebra crossing else, it will stop at a red pedestrian signal. We have also made some random pedestrians violate the pedestrian signal rules, to detect violations made by the pedestrians.

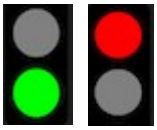

Figure 3. Pedestrian Light Simulations

Code:

{

display signal pedestrian

screen.blit(pedestriansredSignalH, pd.signalCoodsHorizontal[0])

screen.blit(pedestriansredSignal, pd.signalCoods[1])

screen.blit(pedestriansredSignalH, pd.signalCoodsHorizontal[2])

screen.blit(pedestriansredSignal, pd.signalCoods[3])

```
for i in range(0,noOfSignals):
    if  signals[i].red==0 and signals[i].yellow==5 :
      # messagebox.showinfo("note",str(signals[i]))
       if i==0 or i==2 :
          screen.blit(pedestriansgreenSignalH, pd.signalCoodsHorizontal[i -2])
       else:
          screen.blit(pedestriansgreenSignal, pd.signalCoods[i-2])
}
```

4.4. Vehicle Movements

For the car simulator, the highways are the roads with random generation as well as heavy load roads.

For the random vehicle generation, we have built a class of generate vehicles, to randomly generate the vehicles from all four roads. For vehicle movement, we have built a move function to turn the vehicles in three directions, right side, left side, and straight.

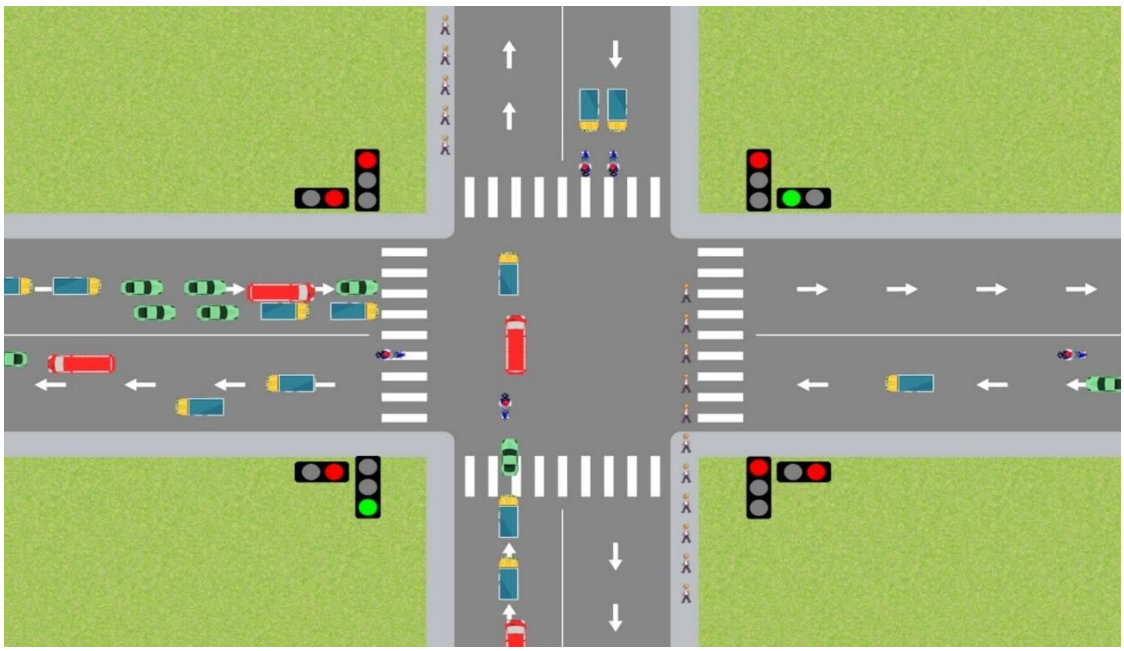

Figure 4. Traffic Movements Simulations

Code:

```
{
class Vehicle(pygame.sprite.Sprite):
    def _init_(self, lane, vehicleClass, direction_number, direction, will_turn):
        pygame.sprite.Sprite._init_(self)
        self.lane = lane
        self.vehicleClass = vehicleClass
        self.speed = speeds[vehicleClass]
        self.direction_number = direction_number
        self.direction = direction
        self.x = x[direction][lane]
        self.y = y[direction][lane]
        self.crossed = 0
        self.willTurn = will_turn
        self.turned = 0
        self.rotateAngle = 0
        vehicles[direction][lane].append(self)
        self.index = len(vehicles[direction][lane]) - 1
        self.crossedIndex = 0
        path = "images/" + direction + "/" + vehicleClass + ".png"
        self.originalImage = pygame.image.load(path)
        self.image = pygame.image.load(path)
}
```

4.5. Simulation Timing

The time for traffic, and pedestrian signals simulations are hard coded. A simulation timer is set to some value. we use the import time module. As the name suggests Python time module allows one to work with time in Python. It allows functionality like getting the current time, pausing the Program from executing, etc. So before starting with this module we need to import it. The timer is integrated into the System.Windows.Threading queue is processed at a specified interval of time and at a specified priority. The value is decremented on all the traffic and pedestrian signals. The particular traffic and pedestrian signal simulation changes to the next condition when the timer reaches to zero.

Code

```
{
# Update values of the signal timers after every second
def updateValues():
    for i in range(0, noOfSignals):
        if(i==currentGreen):
            if(currentYellow==0):
                signals[i].green-=1
            else:
                signals[i].yellow-=1
        else:
            signals[i].red-=1
}
```

5. Design Specification

In this part, we will dicussed the basic flow chart of our project design along with their particular specifications.

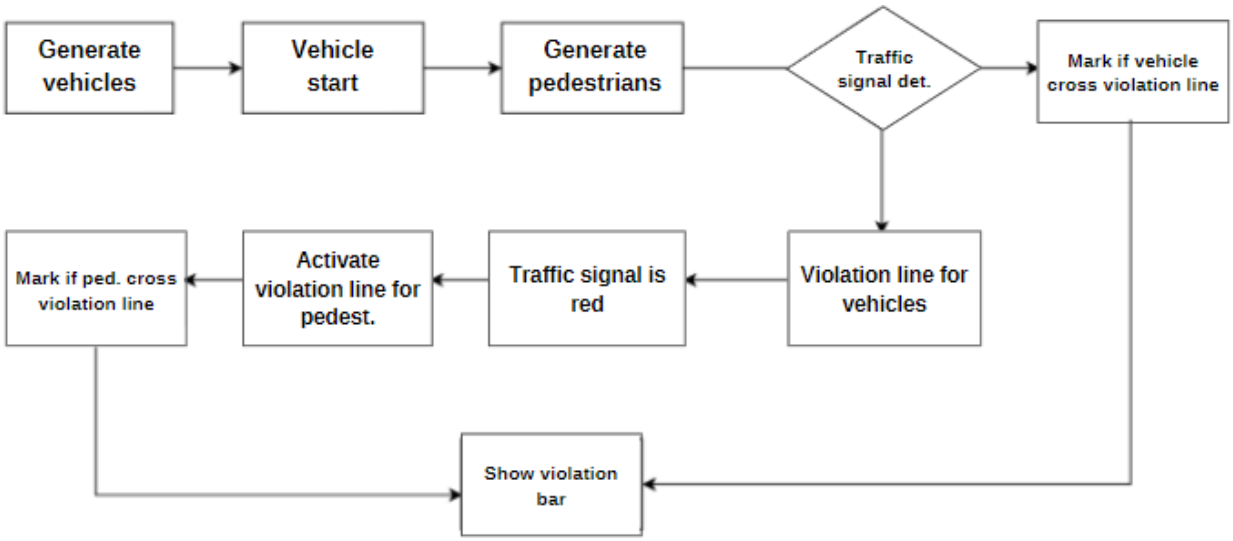

Figure 5. Flow chart of our design specifications

5.1. Generate Vehicle:

In the generate vehicle module, we have made the generate vehicle function randomly to insert vehicles on all the roads.

Code:

{

```
# Generating vehicles in the simulation
def generateVehicles():
    while(True):
        vehicle_type = random.choice(allowedVehicleTypesList)
        lane_number = random.randint(1,2)
        will_turn = 0
        if(lane_number == 1):
            temp = random.randint(0,99)
            if(temp<40):
                will_turn = 1
```

```
        elif(lane_number == 2):
            temp = random.randint(0,99)
            if(temp<40):
                will_turn = 1
        temp = random.randint(0,99)
        direction_number = 0
        dist = [25,50,75,100]
        if(temp<dist[0]):
            direction_number = 0
        elif(temp<dist[1]):
            direction_number = 1
        elif(temp<dist[2]):
            direction_number = 2
        elif(temp<dist[3]):
            direction_number = 3
        Vehicle(lane_number, vehicleTypes[vehicle_type], direction_number,
directionNumbers[direction_number], will_turn)
        time.sleep(1)

}
```

5.2. Generate Pedestrians:

In the generate pedestrain module, we have made the generate pedestrain function randomly to insert pedestrain on all the zabra crossings.

Code:

```
{
def generatePedestrains222():
    while (True):
        vehicle_type = random.randint(0, 3)
        lane_number = random.randint(1, 1)
        temp = random.randint(0, 99)
        direction_number = 0
        dist = [25, 50, 75, 100]
        if (temp < dist[0]):
            direction_number = 0
```

```
        elif (temp < dist[1]):
            direction_number = 1
        elif (temp < dist[2]):
            direction_number = 2
        elif (temp < dist[3]):
            direction_number = 3
        Pedestrains222(lane_number, Types222[Pedestrain_type], direction_number,
directionNumbers222[direction_number])
        time.sleep(1)

    }
```

The main working of our traffic violation detection system is based on the flowchart that we have explained above. The main principles of our coding logic is based on the traffic light signals and pedestrian light signals.

6. Traffic Light Violation Detection

In this project, the simulation was implemented, needed to observe and detect traffic red-light violations. Traffic violations mainly consist of two components: violation line, and vehicle detector.

The figure below illustrates our proposed flow:

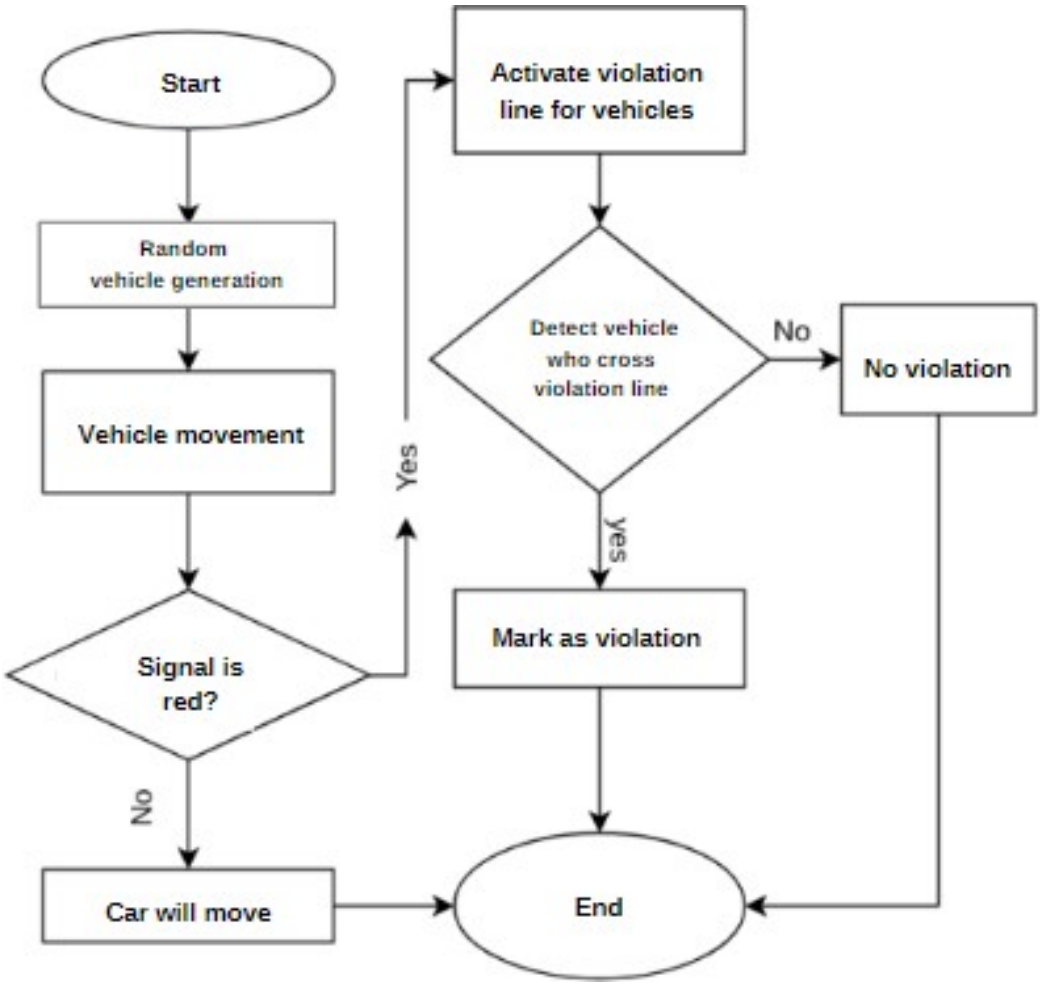

Figure 6. Traffic Light Violation Detection Flow Chart Scenario

Code:

#Trafic line function

Coordinates of stop lines

stopLines = {'right': 480, 'down': 250, 'left': 920, 'up': 675}

defaultStop = {'right': 470, 'down': 240, 'left': 930, 'up': 685}

Output:

In the output below, there is a clear indication in the message box of traffic signal violation occurrence.

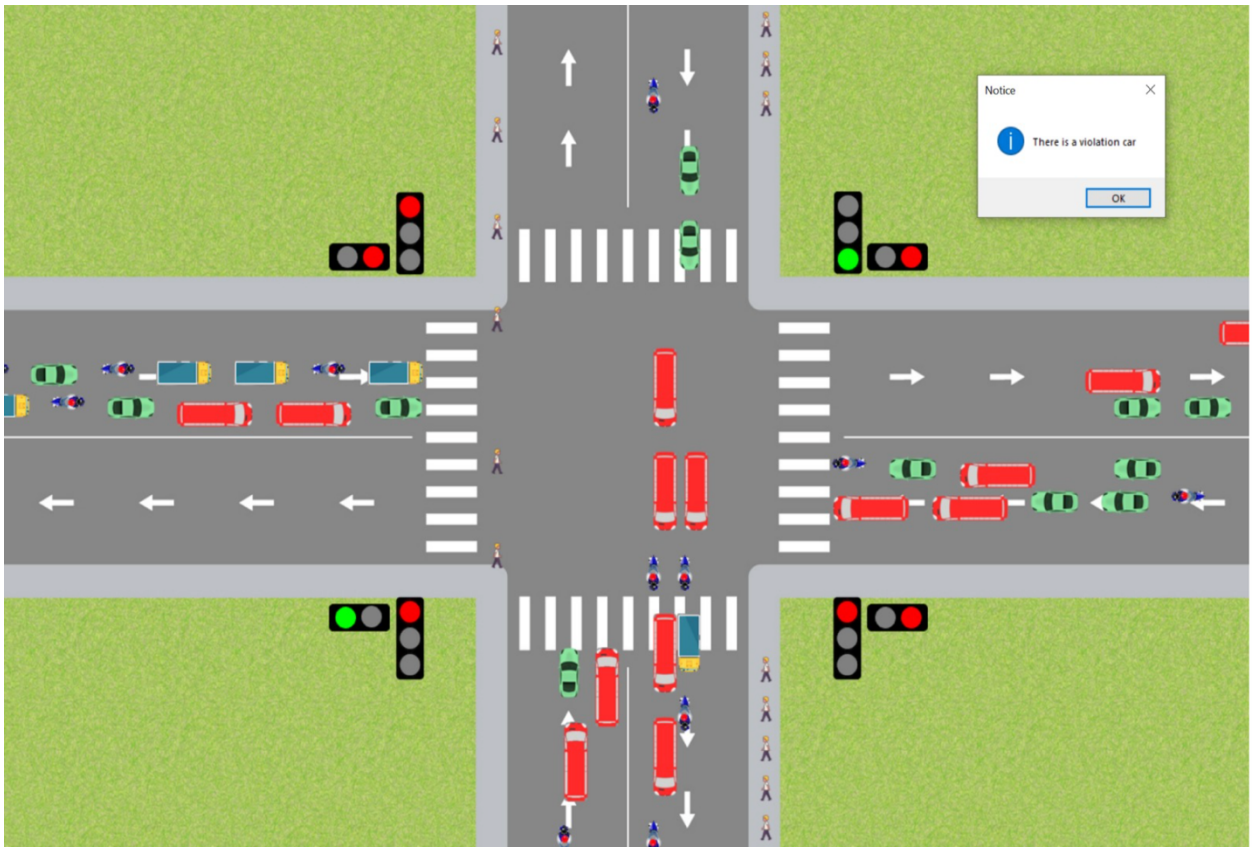

Figure 7. Output of our Traffic Light Violation Detection

7. Pedestrian Light Violation Detection

We have also added the pedestrian light violation system in our project, which primarily needed to observe pedestrian violation movements and detection. Pedestrian violations are also composed of two components: zebra crossing violation line, and pedestrian movements. detector.

The flowchart below illustrates our proposed flow for the detection of pedestrian's violation detection:

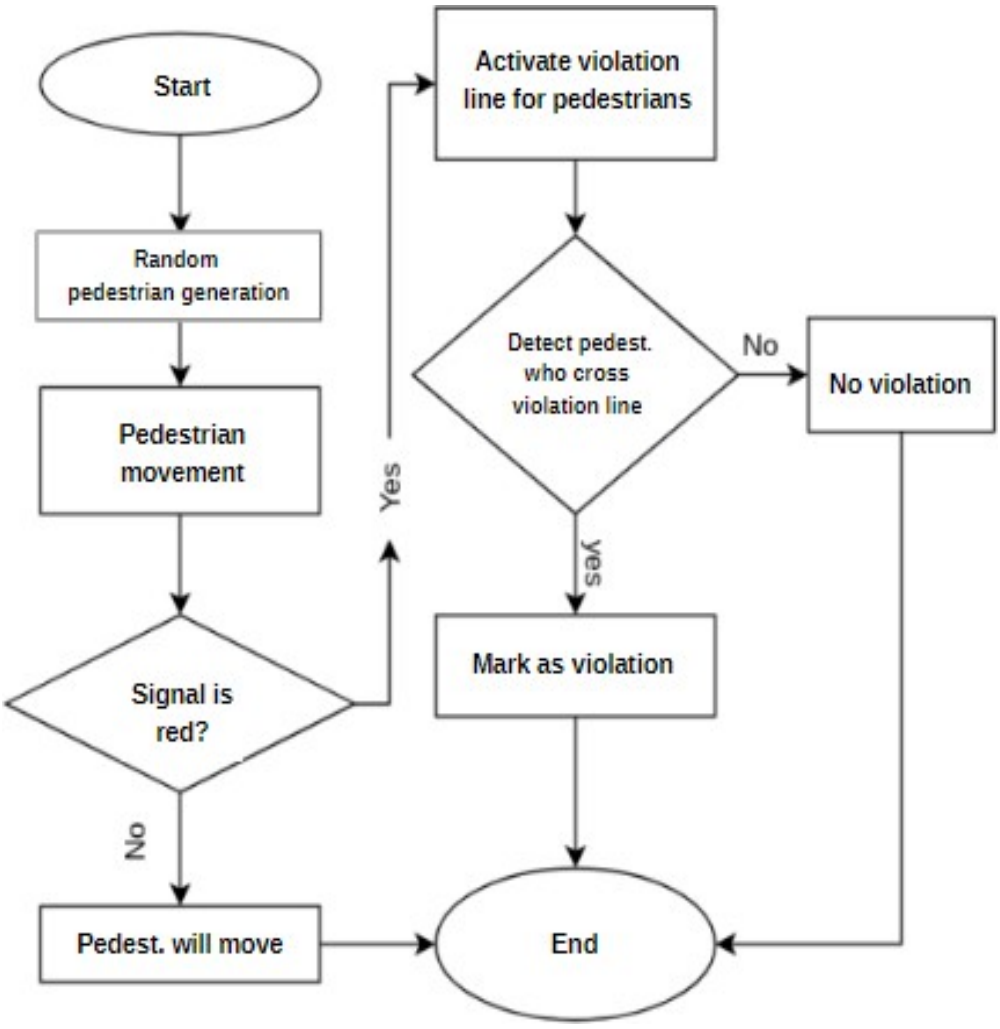

Figure 8. Pedestrian Light Violation Detection Flow Chart Scenario

Code:

```
#Pedestrian line  function

# Coordinates of stop lines

stopLines222 = {'right': 480, 'down': 250, 'left': 920, 'up': 675}

defaultStop222 = {'right': 470, 'down': 240, 'left': 930, 'up': 685
```

Output:

In the output below, there is a clear indication in the message box of pedestrian violation occurrence.

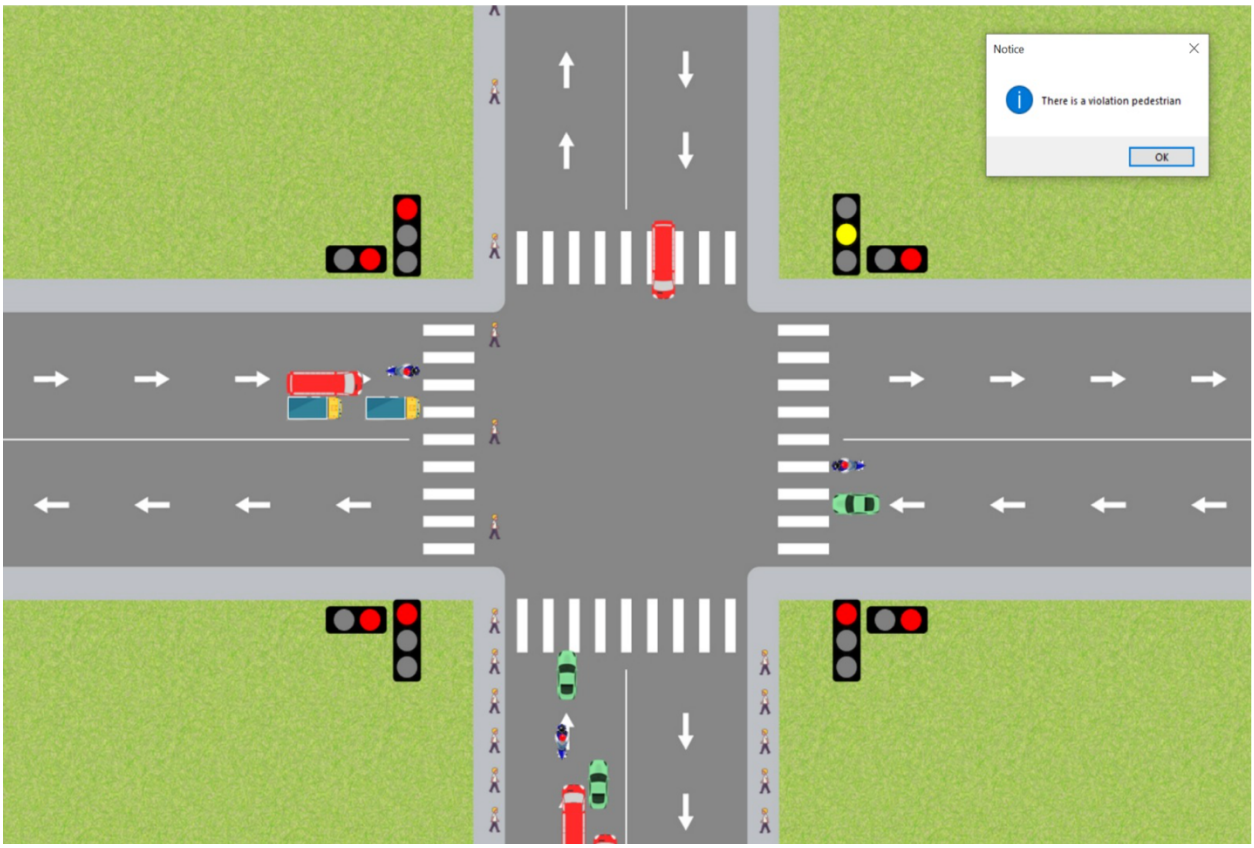

Figure 9. Output of our Pedestrian Light Violation Detection Scenario

8. Future Scope

Our proposed traffic violation detection system can be used for the real-time violation detection of vehicles on the traffic signal, monitoring the vehicles entering the roads and crossing the specific signals. This system can be used in the city or in a specific area. It helps to manage traffic management, and transport management decision logic. Ultimately it will detect pedestrians and traffic light violations and gives us a warning in the notice bar.

Conclusion

Our designed traffic violation detection system was effectively able to detect the type of violation specified in this project which are denying traffic signals and pedestrian signals. The convergence of detection for the traffic violation mentioned will depend on the violation made by vehicles. In the future, more work can be done by adding eight-way road junctions. Since our project has only a four-way traffic control condition and pedestrian signals control conditions. I think you'll get the hang of it. The system provides detection for traffic signal violations.

References

[1] https://www.onisr.securite-routiere.gouv.fr/en/road-safety-performance/annual-road-safety-reports/2021-road-safety-annual-report

[2] https://www.python.org/

[3] https://www.pygame.org/news