VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belagavi - 590018

Mini Project Report
on

"GRAPHICAL REPRESENTATION OF TRAFFIC SIGNALS"

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

BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE AND ENGINEERING
by

SHAMITHA SHETTY	**4MT20CS145**
SHIKHA BALLAL	**4MT20CS150**

Under the Guidance of
MR. SHIVARAJ B G
Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

(Accredited by NBA)

MANGALORE INSTITUTE OF TECHNOLOGY & ENGINEERING

Accredited by NAAC with A+ Grade, An ISO 9001: 2015 Certified Institution
(*A Unit of Rajalaxmi Education Trust®, Mangalore - 575001*)
Affiliated to VTU, Belagavi, Approved by AICTE, New Delhi
Badaga Mijar, Moodabidri-574225, Karnataka

2022-23

MANGALORE INSTITUTE OF TECHNOLOGY &ENGINEERING

Accredited by NAAC with A+ Grade, An ISO 9001: 2015 Certified Institution
(*A Unit of Rajalaxmi Education Trust®, Mangalore - 575001*)
Affiliated to VTU, Belagavi, Approved by AICTE, New Delhi
Badaga Mijar, Moodabidri-574225, Karnataka

DEPARTMENT OF COMPUTERSCIENCE & ENGINEERING

(Accredited by NBA)

CERTIFICATE

This is to certify that Ms. **SHAMITHA SHETTY(4MT20CS145),** Ms. **SHIKHA BALLAL(4MT20CS150)** has satisfactorily completed the mini project entitled "**GRAPHICAL REPRESENTATION OF TRAFFIC SIGNALS"** for the **CG Laboratory with Mini Project (18CSL67)** lab as prescribed by the VTU for 6th semester B.E. Computer Science and Engineering branch for the academic year 2022 – 2023.

…….	**……**
Mr. Shivaraj B G	**Dr. Ravinarayana B**
Assistant Professor	**Associate Professor and HOD**
Dept. of CS&E	**Dept. of CS&E**
MITE Moodabidri	**MITE Moodabidri**

Name of Examiners	**Signature of the Examiners**
1. …….	**……..**
2. …….	**……..**

ACKNOWLEDGEMENT

Any achievement, be it scholastic or otherwise does not depend solely on the individual efforts but on the guidance, encouragement and cooperation of intellectuals, elders and friends. A number of personalities, in their own capacities have helped us in carrying out this project. we would like to take an opportunity to thank them all.

We would like to thank **Prof. Prashanth C M, Principal, Mangalore Institute of Technology and Engineering, Moodabidri**, for his valuable suggestions and expert advice.

We would like to thank **Dr. Ravinarayana B**, **Associate Professor and Head of the Department, Computer Science and Engineering**, Mangalore Institute of Technology and Engineering, Moodabidri, for constant encouragement and support extended towards completing our Project.

We deeply express our sincere gratitude to our guide **Mr. Shivaraj B G, Assistant professor, Department of CS&E**, Mangalore Institute of Technology and Engineering, Moodabidri, for his able guidance, regular source of encouragement and assistance throughout our project period.

Last, but not the least, we would like to thank our peers and friends who provided us with valuable suggestions to improve our project.

Shamitha Shetty **4MT20CS145**

Shikha Ballal **4MT20CS150**

ABSTRACT

Graphics and visualization is the best mean for interaction, understanding and interpretation of data. Computer graphics is basically the rendering of images using computers, usually referring to image data created by a computer specifically with the help from specialized graphical hardware and software. Open Graphics Library (OpenGL) is a cross-language, cross-platform application programming interface (API) for rendering 2D and 3D graphics. The main objective is to demonstrate the working of road traffic signal. As we all know that the traffic signals direct the flow of traffic with the exemption of signals with turning arrows which should be compulsory in accord with each other. From speed limit to directions on where and when to turn, traffic signs provide a wealth of information. Following traffic signs helps to keep everyone on the road safe by reducing the chance of drivers colliding with other vehicles, pedestrians, or cyclists. In this project we are implementing Traffic Signal. We have three lights in the signal red, green and yellow where each light has it's own meaning. Red light tells the driver to stop, green light tells the drivers can start or keep driving, Yellow light tells driver to stop when it is safe to, because the light is about to turn red. The same method we are implementing in our project which is used in real life. We have used this traffic signal in a cross to avoid, miss happening. When there is a green signal it means vehicle can move or run on the road but if this green signal turns off and the red signal is turned on then it indicates the driver to stop his vehicle and at the same the signal allows some other side vehicles to cross the road.

TABLES OF CONTENTS

LIST OF SNAPSHOTS

CHAPTER 1

INTRODUCTION

1.1 Computer Graphics

The term computer graphics has been used in a broad sense to describe “almost everything on computers that is not text or sound”. Typically, the term computer graphics refers to several different things:

- The representation and manipulation of image data by a computer.
- The various technologies used to create and manipulate images.
- The sub-field of computer science which studies methods for digitally synthesizing and manipulating visual content, see study of computer graphics.

Today, computer graphics is widespread. Such imagery is found in and on television, newspapers, weather reports, and in a variety of medical investigations and surgical procedures. A well- constructed graph can present complex statistics in a form that is easier to understand and interpret. In the media “such graphs are used to illustrate papers, reports, thesis”, and other presentation material.

Many tools have been developed to visualize data. Computer generated imagery can be categorized into several different types: two dimensional (2D), three dimensional (3D), and animated graphics. As technology has improved, 3D computer graphics have become more common, but 2D computer graphics are still widely used. Computer graphics has emerged as a sub-field of computer science with studies methods for digitally synthesizing and manipulating visual content. Over the past decade, other specialized fields have been developed like information visualization, and scientific visualization more concerned with “the visualization of three-dimensional phenomena (architectural, meteorological, medical, biological, etc.), where the emphasis is on realistic renderings of volumes, surfaces, illumination sources, and so forth, perhaps with a dynamic (time) component”.

OpenGL

OpenGL (Open Graphics Library) is a cross-platform, hardware-accelerated, language-independent, industrial standard API for producing 3D (including 2D) graphics. The

OpenGL specification describes an abstract API for drawing 2D and 3D graphics. OpenGL is the standard application program interface (API) for defining 2D and 3D images. Prior to OpenGL, any company developing a graphical application typically had to rewrite the graphics part of it for each operating system platform and had to be cognizant of the graphics hardware as well. With OpenGL, an application can create the same effects in any operating system using any OpenGL adhering graphics adapter.

OpenGL specifies a set of "commands" or immediately executed functions. Each command directs a drawing action or causes special effects. A list of these commands can be created for repetitive effects. OpenGL is independent of the windowing characteristics of each operating system, but provides special "glue" routines for each operating system that enable OpenGL to work in that system's windowing environment. OpenGL comes with a large number of built-in capabilities requestable through the API. These include hidden surface removal, alpha blending (transparency), antialiasing, texture mapping, pixel operations, viewing and modelling transformations, and atmospheric effects (fog, smoke, and haze).

Although the function definitions are superficially similar to those of the programming language C, they are language-independent. In addition to being language-independent, OpenGL is also cross-platform. The specification says nothing on the subject of obtaining, and managing an OpenGL context, leaving this as a detail of the underlying windowing system. For the same reason, OpenGL is purely concerned with rendering, providing no APIs related to input, audio or windowing. In addition to the features required by the core API, graphics processing unit (GPU) vendors may provide additional functionality in the form of extensions. Extensions may introduce new functions and new constants, and may relax or remove restrictions on existing OpenGL functions. All extensions are collected in, and defined by, the OpenGL Registry.

Summary: This chapter gives an overview of computer graphics.

CHAPTER 2

REQUIREMENT ANALYSIS

2.1 Non-Functional Requirements

In system engineering and requirements engineering, a non-functional requirement (NFR) is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. They are contrasted with functional requirements that define specific behavior or functions. Some of the non-functional requirements are:

- Performance: for example: response time, throughput, utilization, static volumetric.
- Scalability.
- Capacity.
- Availability.
- Reliability.
- Recoverability.
- Maintainability.
- Serviceability.

2.2 Functional Requirements

The functional requirements are the requirements which are needed to develop the software application. The functional requirement are broadly classified into 2 categories, they are,

1. Hardware Requirements.
2. Software Requirements.

2.2.1 Hardware Requirements

For the hardware requirements the SRS (Software Requirement Specification) specifies the logical characteristics of each interface between the software product and the hardware components. It specifies the hardware requirements like memory restrictions, cache size, the processor, RAM size etc. those are required for the software to run.

- Processor : Intel i3/i5/i7
- Processor speed : 1 GB or more
- Hard disk : 40 GB or more
- RAM size : 1 GB or more
- Display with 256 colors : 800x600 or higher resolution display.
- Mouse : Standard serial mouse.
- Keyboard : Standard QWERTY Keyboard.
- GPU : Intel HD Graphics 5000 or better.

2.2.2 Software Requirements

- Operating System : Windows 10, 64 bits.
- Software used : Dev-C++.
- Programming Language : C language.
- Graphics Library : GL and GLU/GLUT.
- Compiler used : MinGW Compiler.

CHAPTER 3

DESIGN

3.1 Description

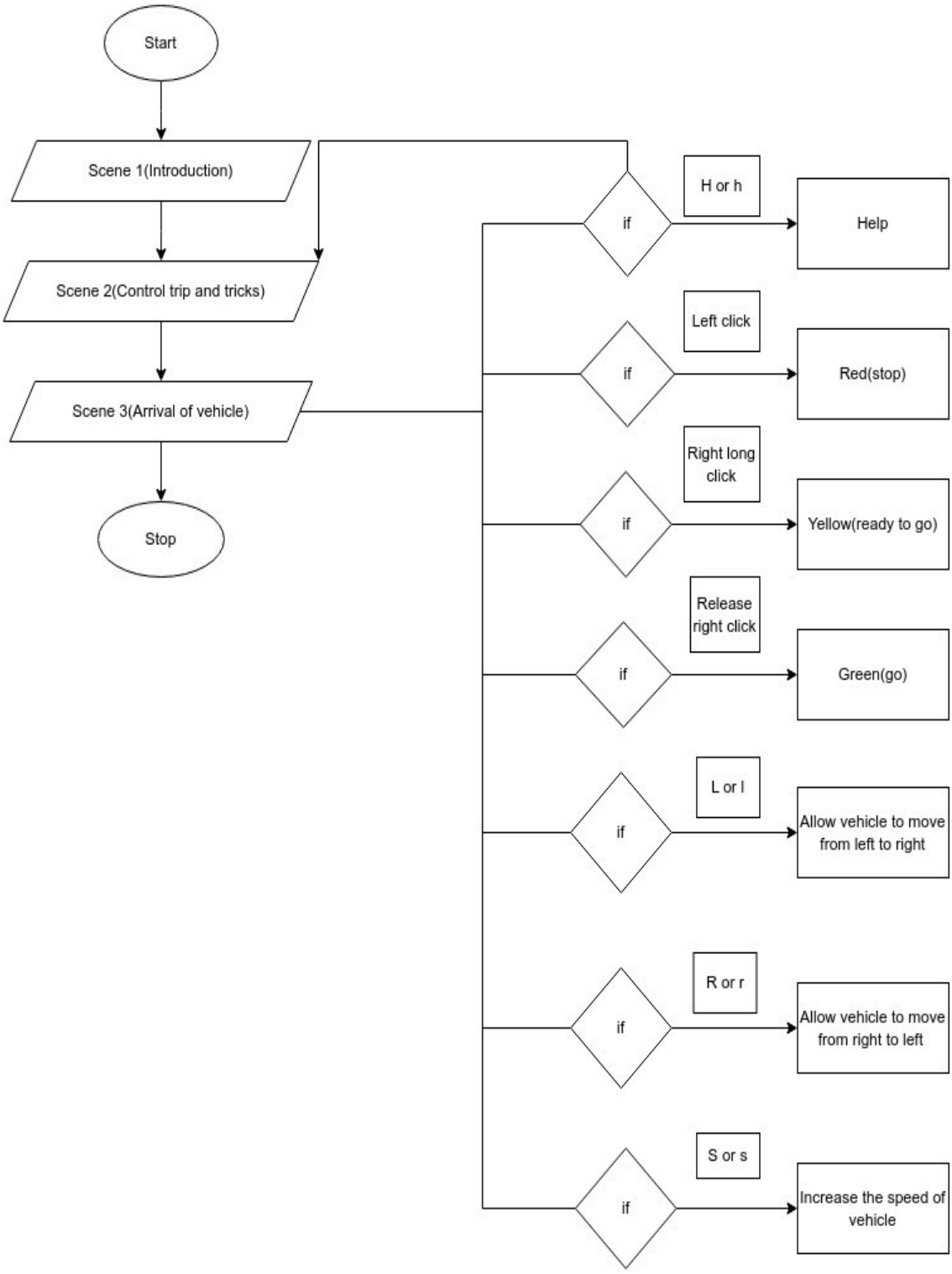
Traffic signal is a newly designed computer graphics mini project. In this mini project there will be objects like bus, cars and signals and a place called “Road”. In our mini project, the second scene describes the control of the projects like the key and mouse control. The third scene describes about the arrival of car and bus in opposite direction. Here when you press L or l allows vehicle to move from left to right and right vehicle will stop and when you press R or r allows vehicle to move from right to left and left vehicle will stop and S or s used to speed up the vehicle. Here in third scene when you press left mouse button red (stop) signal appears and vehicle stops and when you press right long mouse button yellow (ready to go) signal appears and vehicles moves and you release right button green (go) signal appears and vehicles moves.

3.2 Algorithm

An algorithm is a step-by- step instruction to execute the program.

```
Step 1: Initialize the graphics library.
Step 2: Display the starting page.
Step 3: Take the user input.
If input = ENTER
Scene 1().
end if
Step 4: Press enter for the
Scene 2.
If input = h
Scene 2().
Step 5: Press enter for the
Scene 3.
If input = e
Scene 3().
```

3.3 Flowchart



CHAPTER 4

IMPLEMENTATION

4.1 Overview

Implementation is the carrying out, execution, or practice of a plan, a method, or any design for doing something. As such, implementation is the action that must follow any preliminary thinking in order for something to actually happen. In an information technology context, implementation encompasses all the processes involved in getting new software or hardware operating properly in its environment, including installation, configuration, running, testing, and making necessary changes. The word deployment is sometimes used to mean the same thing. This project has been developed using OpenGL provided by Visual C++.

4.2 Code Snippets

Code Snippet-1

```
glPushMatrix();
glScaled(40.0,40.0,0.0);
glColor3f(0.1,0.1,0.1);
glBegin(GL_POLYGON);
//straight road
glVertex2f(0,5);
glVertex2f(40,5);
glVertex2f(40,10);
glVertex2f(0,10);
glEnd();
//Side lines
glBegin(GL_LINES);
glColor3f(1.0,1.0,1.0);
glVertex2f(0,7);
glVertex2f(7,7);
glEnd();
//side right line
glBegin(GL_LINES);
glColor3f(1.0,1.0,1.0);
glVertex2f(22,7);
glVertex2f(35,7);
glEnd();
//up line not proper
```

```
glBegin(GL_LINES);
glColor3f(1.0,1.0,1.0);
glVertex2f(11,14);
glVertex2f(8,30);
glEnd();
//green edge
glBegin(GL_POLYGON);
glColor3f(0.1,0.2,0.1);
glVertex2f(0,5);
glVertex2f(40,5);
glVertex2f(40,4);
glVertex2f(0,4);
glEnd();
//cross road
glColor3f(0.1,0.1,0.1);
glBegin(GL_POLYGON);
glVertex2f(10,10);
glVertex2f(15,10);
glVertex2f(0,40);
glVertex2f(4,40);
glEnd();
glPopMatrix();
```

Code Snippet-2

```
void signal()
{
glPushMatrix();
glScaled(40.0,40.0,0.0);
//stand
glColor3f(0.1,0.2,0.1);
glBegin(GL_POLYGON);
glVertex2f(15,7);
glVertex2f(15,8);
glVertex2f(18,8);
glVertex2f(18,7);
glEnd();
//pole
glBegin(GL_POLYGON);
glVertex2f(16,7);
glVertex2f(17,8);
glVertex2f(17,15);
glVertex2f(16,15);
glEnd();
//

glBegin(GL_LINES);

glVertex2f(16,8.3);
glVertex2f(14.5,8.3);
glEnd();
glBegin(GL_LINES);
```

```
glVertex2f(19.5,7);
glVertex2f(20,8.3);
glEnd();
glBegin(GL_LINES);

glVertex2f(20,8.3);
glVertex2f(17,8.3);
glEnd();
//double
glBegin(GL_LINES);

glVertex2f(16,8.5);
glVertex2f(14.3,8.5);
glEnd();
glBegin(GL_LINES);

glVertex2f(14.3,8.5);
glVertex2f(13.2,6.8);
glEnd();
glBegin(GL_LINES);

glVertex2f(13.2,6.8);
glVertex2f(19.7,6.8);
glEnd();
glBegin(GL_LINES);

glVertex2f(19.7,6.8);
glVertex2f(20.2,8.5);
glEnd();
glBegin(GL_LINES);

glVertex2f(20.2,8.5);
glVertex2f(17,8.5);
glEnd();
//board
glBegin(GL_POLYGON);
glVertex2f(15.5,15);
glVertex2f(17.5,15);
glVertex2f(17.5,10);
glVertex2f(15.5,10);
glEnd();
```

```
//red
glColor3f(p,0.0,0.0);
glBegin(GL_POLYGON);
glVertex2f(16,14.5);
glVertex2f(17,14.5);
glVertex2f(17,14);
glVertex2f(16,14);
glEnd();
//yellow
glColor3f(q,q,0.0);
glBegin(GL_POLYGON);
glVertex2f(16,13.5);
glVertex2f(17,13.5);
glVertex2f(17,13);
glVertex2f(16,13);
glEnd();
//green
glColor3f(0.0,r,0.0);
glBegin(GL_POLYGON);
glVertex2f(16,12.5);
glVertex2f(17,12.5);
glVertex2f(17,12);
glVertex2f(16,12);
glEnd();
glPopMatrix();
```

Code Snippet-3

```
void bus()
{
glPushMatrix();
glTranslated(a,50.0,0.0);
glScaled(40.0,40.0,0.0);
glColor3f(0.5,0.0,0.0);
//bus out line
glBegin(GL_POLYGON);
glVertex2f(25,8);
glVertex2f(25,9.5);
glVertex2f(26,11);
glVertex2f(32,11);
glVertex2f(32,8);
glEnd();
//window frame
glColor3f(0,0.1,1);
glBegin(GL_POLYGON);
glVertex2f(26.1,9.5);
glVertex2f(26.1,10.5);
glVertex2f(31.8,10.5);
glVertex2f(31.8,9.5);
glEnd();
```

```
//Doors
glColor3f(0,0.8,1);
glBegin(GL_POLYGON);
glVertex2f(26.2,9);
glVertex2f(26.2,10.4);
glVertex2f(27.7,10.4);
glVertex2f(27.7,9);
glEnd();


glColor3f(1,1,1);
glBegin(GL_POLYGON);
glVertex2f(27,8.4);
glVertex2f(27,10.4);
glVertex2f(27.7,10.4);
glVertex2f(27.7,8.4);
glEnd();
//small windows
glColor3f(0,1,1);
glBegin(GL_POLYGON);
glVertex2f(27.8,9.6);
glVertex2f(27.8,10.4);
glVertex2f(29,10.4);
glVertex2f(29,9.6);
glEnd();
glBegin(GL_POLYGON);
glVertex2f(29.1,9.6);
glVertex2f(29.1,10.4);
glVertex2f(30.2,10.4);
glVertex2f(30.2,9.6);
glEnd();
glBegin(GL_POLYGON);
glVertex2f(30.3,9.6);
glVertex2f(30.3,10.4);
glVertex2f(31.7,10.4);
glVertex2f(31.7,9.6);
glEnd();


//driver window
glColor3f(0,0.8,1);
glBegin(GL_POLYGON);
```

```
glVertex2f(25,9.5);
glVertex2f(26,11);
glVertex2f(26,9.5);
glEnd();
glPopMatrix();
//tyre
glPushMatrix();//front tyre
glTranslated(a+970,320,0.0);
glScaled(20.0,20.0,0.0);
glColor3f(0.0,0.0,0.0);
glBegin(GL_POLYGON);
glVertex2f(3.0,2.5);
glVertex2f(3.0,2.6);
glVertex2f(3.15,3.1);
glVertex2f(3.2,3.2);
glVertex2f(3.3,3.35);
glVertex2f(3.4,3.4);
glVertex2f(3.5,3.45);
glVertex2f(3.6,3.55);
glVertex2f(3.7,3.6);
glVertex2f(3.8,3.63);
glVertex2f(4.0,3.65);
glVertex2f(4.2,3.7);
glVertex2f(4.4,3.7);
glVertex2f(4.6,3.65);
glVertex2f(4.8,3.55);
glVertex2f(5.0,3.45);
glVertex2f(5.1,3.4);
glVertex2f(5.2,3.25);
glVertex2f(5.3,3.2);
glVertex2f(5.4,3.0);
glVertex2f(5.5,2.5);

glVertex2f(5.45,2.15);
glVertex2f(5.4,1.9);
glVertex2f(5.35,1.8);
glVertex2f(5.2,1.6);
glVertex2f(5.0,1.5);
glVertex2f(4.9,1.4);
glVertex2f(4.7,1.3);
glVertex2f(4.6,1.27);
glVertex2f(4.4,1.25);
```

```
glVertex2f(4.4,1.25);
glVertex2f(4.0,1.25);
glVertex2f(3.9,1.3);
glVertex2f(3.75,1.35);
glVertex2f(3.6,1.4);
glVertex2f(3.45,1.55);
glVertex2f(3.3,1.7);
glVertex2f(3.2,1.8);
glVertex2f(3.1,2.2);
glEnd();
glPopMatrix();

glPushMatrix();//back tyre
glTranslated(a+1140,320,0.0);
glScaled(20.0,20.0,0.0);
glColor3f(0.0,0.0,0.0);
glBegin(GL_POLYGON);
glVertex2f(3.0,2.5);
glVertex2f(3.0,2.6);
glVertex2f(3.15,3.1);
glVertex2f(3.2,3.2);
glVertex2f(3.3,3.35);
glVertex2f(3.4,3.4);
glVertex2f(3.5,3.45);
glVertex2f(3.6,3.55);
glVertex2f(3.7,3.6);
glVertex2f(3.8,3.63);
glVertex2f(4.0,3.65);
glVertex2f(4.2,3.7);
glVertex2f(4.4,3.7);
glVertex2f(4.6,3.65);
glVertex2f(4.8,3.55);
glVertex2f(5.0,3.45);
glVertex2f(5.1,3.4);
glVertex2f(5.2,3.25);
glVertex2f(5.3,3.2);
glVertex2f(5.4,3.0);
glVertex2f(5.5,2.5);

glVertex2f(5.45,2.15);
glVertex2f(5.4,1.9);
glVertex2f(5.35,1.8);
```

```
glVertex2f(5.2,1.6);
glVertex2f(5.0,1.5);
glVertex2f(4.9,1.4);
glVertex2f(4.7,1.3);
glVertex2f(4.6,1.27);
glVertex2f(4.4,1.25);
glVertex2f(4.0,1.25);
glVertex2f(3.9,1.3);
glVertex2f(3.75,1.35);
glVertex2f(3.6,1.4);
glVertex2f(3.45,1.55);
glVertex2f(3.3,1.7);
glVertex2f(3.2,1.8);
glVertex2f(3.1,2.2);
glEnd();
glPopMatrix();
}
```

Code Snippet-4

```
void car()
{
glPushMatrix(); //making color for outer line
glTranslated(b,190.0,0.0);
glScaled(20.0,20.0,0.0);
glColor3f(1.0,0.0,0.0);
glBegin(GL_POLYGON);
glVertex2f(2.5,2.5);
glVertex2f(3.0,3.5);
glVertex2f(3.5,3.75);
glVertex2f(4.0,4.0);
glVertex2f(4.5,4.0);
glVertex2f(5.0,3.75);
glVertex2f(5.5,3.5);
glVertex2f(5.75,3.0);
glVertex2f(6.0,2.5);
glVertex2f(16.5,2.5);
glVertex2f(16.75,3.0);
glVertex2f(17.0,3.5);
glVertex2f(17.5,3.75);
glVertex2f(18.0,4.0);
glVertex2f(18.5,4.0);
glVertex2f(19.0,3.75);
glVertex2f(19.5,3.5);
```

```
glVertex2f(17.5,3.75);
glVertex2f(18.0,4.0);
glVertex2f(18.5,4.0);
glVertex2f(19.0,3.75);
glVertex2f(19.5,3.5);
glVertex2f(19.75,3.0);
glVertex2f(20.0,2.5);
glVertex2f(21.0,2.5);
glVertex2f(21.0,4.0);
glVertex2f(21.5,4.0);
glVertex2f(21.0,4.5);
glVertex2f(20.0,5.0);
glVertex2f(15.0,5.0);
glVertex2f(14.0,5.5);
glVertex2f(13.0,6.0);
glVertex2f(12.0,6.5);
glVertex2f(11.0,7.0);
glVertex2f(6.0,7.0);
glVertex2f(5.0,6.5);
glVertex2f(4.5,6.25);
glVertex2f(4.25,6.0);
glVertex2f(4.0,5.75);
glVertex2f(3.5,5.5);
glVertex2f(3.0,5.5);
glVertex2f(1.9,5.45);
glVertex2f(1.8,5.4);
glVertex2f(1.7,5.35);
glVertex2f(1.6,5.3);
glVertex2f(1.5,5.25);
glVertex2f(1.4,5.15);
glVertex2f(1.3,5.0);
glVertex2f(1.2,4.85);
glVertex2f(1.1,4.7);
glVertex2f(1.0,4.3);
glVertex2f(1.0,3.2);
glVertex2f(1.1,3.05);
glVertex2f(1.2,2.9);
glVertex2f(1.3,2.9);
glVertex2f(1.4,2.75);
glVertex2f(1.5,2.65);
glVertex2f(1.6,2.6);
glVertex2f(1.7,2.55);
```

```
glVertex2f(1.8,2.5);
glVertex2f(1.9,2.45);
glVertex2f(2.0,2.5);
glEnd();

glColor3f(1.0,1.0,1.0); //color for outer window
glBegin(GL_POLYGON);
glVertex2f(5.0,5.0);
glVertex2f(14.0,5.0);
glVertex2f(11.5,6.5);
glVertex2f(10.5,6.75);
glVertex2f(7.0,6.75);
glEnd();

glColor3f(0.0,0.0,0.0); //making outer line for car
glBegin(GL_LINE_LOOP);
glVertex2f(2.5,2.5);
glVertex2f(3.0,3.5);
glVertex2f(3.5,3.75);
glVertex2f(4.0,4.0);
glVertex2f(4.5,4.0);
glVertex2f(5.0,3.75);
glVertex2f(5.5,3.5);
glVertex2f(5.75,3.0);
glVertex2f(6.0,2.5);
glVertex2f(16.5,2.5);
glVertex2f(16.75,3.0);
glVertex2f(17.0,3.5);
glVertex2f(17.5,3.75);
glVertex2f(18.0,4.0);
glVertex2f(18.5,4.0);
glVertex2f(19.0,3.75);
glVertex2f(19.5,3.5);
glVertex2f(19.75,3.0);
glVertex2f(20.0,2.5);
glVertex2f(21.0,2.5);
glVertex2f(21.0,4.0);
glVertex2f(21.5,4.0);
glVertex2f(21.0,4.5);
glVertex2f(20.0,5.0);
glVertex2f(15.0,5.0);
glVertex2f(14.0,5.5);
```

```
glVertex2f(13.0,6.0);
glVertex2f(12.0,6.5);
glVertex2f(11.0,7.0);
glVertex2f(6.0,7.0);
glVertex2f(5.0,6.5);
glVertex2f(4.5,6.25);
glVertex2f(4.25,6.0);
glVertex2f(4.0,5.75);
glVertex2f(3.5,5.5);
glVertex2f(3.0,5.5);
glVertex2f(1.9,5.45);
glVertex2f(1.8,5.4);
glVertex2f(1.7,5.35);
glVertex2f(1.6,5.3);
glVertex2f(1.5,5.25);
glVertex2f(1.4,5.15);
glVertex2f(1.3,5.0);
glVertex2f(1.2,4.85);
glVertex2f(1.1,4.7);
glVertex2f(1.0,4.3);
glVertex2f(1.0,3.2);
glVertex2f(1.1,3.05);
glVertex2f(1.2,2.9);
glVertex2f(1.3,2.9);
glVertex2f(1.4,2.75);
glVertex2f(1.5,2.65);
glVertex2f(1.6,2.6);
glVertex2f(1.7,2.55);
glVertex2f(1.8,2.5);
glVertex2f(1.9,2.45);
glVertex2f(2.0,2.5);
glEnd();

glColor3f(0.0,0.0,0.0);
glBegin(GL_LINE_LOOP); //outer line for design a car
glVertex2f(8.0,3.0);
glVertex2f(16.0,3.0);
glVertex2f(16.5,3.5);
glVertex2f(17.0,4.0);
glVertex2f(16.5,4.25);
glVertex2f(16.0,4.5);
glVertex2f(15.0,4.5);
```

```
glVertex2f(15.0,5.0);
glVertex2f(14.0,5.0);
glVertex2f(11.5,6.5);
glVertex2f(10.5,6.75);
glVertex2f(7.0,6.75);
glVertex2f(5.0,5.0);
glVertex2f(7.0,5.0);
glVertex2f(6.5,4.5);
glEnd();


glBegin(GL_LINES); //connecting outer line
glVertex2d(7.0,5.0);
glVertex2d(15.0,5.0);
glEnd();

glColor3f(0.0,0.0,0.0); //connecting outer line
glBegin(GL_LINES);
glVertex2d(15.0,4.0);
glVertex2d(17.0,4.0);
glEnd();

glColor3f(0.0,0.0,0.0); //connecting outer line
glBegin(GL_LINES);
glVertex2d(15.0,3.5);
glVertex2d(16.5,3.5);
glEnd();

glColor3f(0.0,0.0,0.0); //connecting outer line
glBegin(GL_LINES);
glVertex2d(15.0,5.0);
glVertex2d(14.0,3.0);
glEnd();

glColor3f(0.0,0.0,0.0); //connecting outer line
glBegin(GL_LINES);
glVertex2d(12.0,5.0);
glVertex2d(12.0,6.2);
glEnd();

glColor3f(0.0,0.0,0.0); //connecting outer line
glBegin(GL_LINES);
```

```
glVertex2d(7.0,5.0);
glVertex2d(7.0,6.7);
glEnd();

glBegin(GL_POLYGON); //drawing a back tyre
glVertex2f(3.0,2.5);
glVertex2f(3.0,2.6);
glVertex2f(3.15,3.1);
glVertex2f(3.2,3.2);
glVertex2f(3.3,3.35);
glVertex2f(3.4,3.4);
glVertex2f(3.5,3.45);
glVertex2f(3.6,3.55);
glVertex2f(3.7,3.6);
glVertex2f(3.8,3.63);
glVertex2f(4.0,3.65);
glVertex2f(4.2,3.7);
glVertex2f(4.4,3.7);
glVertex2f(4.6,3.65);
glVertex2f(4.8,3.55);
glVertex2f(5.0,3.45);
glVertex2f(5.1,3.4);
glVertex2f(5.2,3.25);
glVertex2f(5.3,3.2);
glVertex2f(5.4,3.0);
glVertex2f(5.5,2.5);

glVertex2f(5.45,2.15);
glVertex2f(5.4,1.9);
glVertex2f(5.35,1.8);
glVertex2f(5.2,1.6);
glVertex2f(5.0,1.5);
glVertex2f(4.9,1.4);
glVertex2f(4.7,1.3);
glVertex2f(4.6,1.27);
glVertex2f(4.4,1.25);
glVertex2f(4.0,1.25);
glVertex2f(3.9,1.3);
glVertex2f(3.75,1.35);
glVertex2f(3.6,1.4);
glVertex2f(3.45,1.55);
glVertex2f(3.3,1.7);
```

```
glVertex2f(3.2,1.8);
glVertex2f(3.1,2.2);
glEnd();


glBegin(GL_POLYGON); //drawing front tyre
glVertex2f(17.0,2.5);
glVertex2f(17.0,2.6);
glVertex2f(17.15,3.1);
glVertex2f(17.2,3.2);
glVertex2f(17.3,3.35);
glVertex2f(17.4,3.4);
glVertex2f(17.5,3.45);
glVertex2f(17.6,3.55);
glVertex2f(17.7,3.6);
glVertex2f(17.8,3.63);
glVertex2f(18.0,3.65);
glVertex2f(18.2,3.7);
glVertex2f(18.4,3.7);
glVertex2f(18.6,3.65);
glVertex2f(18.8,3.55);
glVertex2f(19.0,3.45);
glVertex2f(19.1,3.4);
glVertex2f(19.2,3.25);
glVertex2f(19.3,3.2);
glVertex2f(19.4,3.0);

glVertex2f(19.5,2.5);
glVertex2f(19.45,2.15);
glVertex2f(19.4,1.9);
glVertex2f(19.35,1.8);
glVertex2f(19.2,1.6);
glVertex2f(19.0,1.5);
glVertex2f(18.9,1.4);
glVertex2f(18.7,1.3);
glVertex2f(18.6,1.27);
glVertex2f(18.4,1.25);
glVertex2f(18.0,1.25);
glVertex2f(17.9,1.3);
glVertex2f(17.75,1.35);
glVertex2f(17.6,1.4);
glVertex2f(17.45,1.55);
```

```
glVertex2f(17.3,1.7);
glVertex2f(17.2,1.8);
glVertex2f(17.1,2.2);
glEnd();
glPopMatrix();
}
void car2()
{
glPushMatrix(); //making color for outer line
glTranslated(b-2000,190.0,0.0);
glScaled(20.0,20.0,0.0);
glColor3f(1.0,1.0,0.4);
glBegin(GL_POLYGON);
glVertex2f(2.5,2.5);
glVertex2f(3.0,3.5);
glVertex2f(3.5,3.75);
glVertex2f(4.0,4.0);
glVertex2f(4.5,4.0);
glVertex2f(5.0,3.75);
glVertex2f(5.5,3.5);
glVertex2f(5.75,3.0);
glVertex2f(6.0,2.5);
glVertex2f(16.5,2.5);
glVertex2f(16.75,3.0);
glVertex2f(17.0,3.5);
glVertex2f(17.5,3.75);
glVertex2f(18.0,4.0);
glVertex2f(18.5,4.0);
glVertex2f(19.0,3.75);
glVertex2f(19.5,3.5);
glVertex2f(19.75,3.0);
glVertex2f(20.0,2.5);
glVertex2f(21.0,2.5);
glVertex2f(21.0,4.0);
glVertex2f(21.5,4.0);
glVertex2f(21.0,4.5);
glVertex2f(20.0,5.0);
glVertex2f(15.0,5.0);
glVertex2f(14.0,5.5);
glVertex2f(13.0,6.0);
glVertex2f(12.0,6.5);
glVertex2f(11.0,7.0);
```

```
glVertex2f(6.0,7.0);
glVertex2f(5.0,6.5);
glVertex2f(4.5,6.25);
glVertex2f(4.25,6.0);
glVertex2f(4.0,5.75);
glVertex2f(3.5,5.5);
glVertex2f(3.0,5.5);
glVertex2f(1.9,5.45);
glVertex2f(1.8,5.4);
glVertex2f(1.7,5.35);
glVertex2f(1.6,5.3);
glVertex2f(1.5,5.25);
glVertex2f(1.4,5.15);
glVertex2f(1.3,5.0);
glVertex2f(1.2,4.85);
glVertex2f(1.1,4.7);
glVertex2f(1.0,4.3);
glVertex2f(1.0,3.2);
glVertex2f(1.1,3.05);
glVertex2f(1.2,2.9);
glVertex2f(1.3,2.9);
glVertex2f(1.4,2.75);
glVertex2f(1.5,2.65);
glVertex2f(1.6,2.6);
glVertex2f(1.7,2.55);
glVertex2f(1.8,2.5);
glVertex2f(1.9,2.45);
glVertex2f(2.0,2.5);
glEnd();

glColor3f(1.0,1.0,1.0); //color for outer window
glBegin(GL_POLYGON);
glVertex2f(5.0,5.0);
glVertex2f(14.0,5.0);
glVertex2f(11.5,6.5);
glVertex2f(10.5,6.75);
glVertex2f(7.0,6.75);
glEnd();

glColor3f(0.0,0.0,0.0); //making outer line for car
glBegin(GL_LINE_LOOP);
glVertex2f(2.5,2.5);
```

```
glVertex2f(3.0,3.5);
glVertex2f(3.5,3.75);
glVertex2f(4.0,4.0);
glVertex2f(4.5,4.0);
glVertex2f(5.0,3.75);
glVertex2f(5.5,3.5);
glVertex2f(5.75,3.0);
glVertex2f(6.0,2.5);
glVertex2f(16.5,2.5);
glVertex2f(16.75,3.0);
glVertex2f(17.0,3.5);
glVertex2f(17.5,3.75);
glVertex2f(18.0,4.0);
glVertex2f(18.5,4.0);
glVertex2f(19.0,3.75);
glVertex2f(19.5,3.5);
glVertex2f(19.75,3.0);
glVertex2f(20.0,2.5);
glVertex2f(21.0,2.5);
glVertex2f(21.0,4.0);
glVertex2f(21.5,4.0);
glVertex2f(21.0,4.5);
glVertex2f(20.0,5.0);
glVertex2f(15.0,5.0);
glVertex2f(14.0,5.5);
glVertex2f(13.0,6.0);
glVertex2f(12.0,6.5);
glVertex2f(11.0,7.0);
glVertex2f(6.0,7.0);
glVertex2f(5.0,6.5);
glVertex2f(4.5,6.25);
glVertex2f(4.25,6.0);
glVertex2f(4.0,5.75);
glVertex2f(3.5,5.5);
glVertex2f(3.0,5.5);
glVertex2f(1.9,5.45);
glVertex2f(1.8,5.4);
glVertex2f(1.7,5.35);
glVertex2f(1.6,5.3);
glVertex2f(1.5,5.25);
glVertex2f(1.4,5.15);
glVertex2f(1.3,5.0);
```

```
glVertex2f(1.2,4.85);
glVertex2f(1.1,4.7);
glVertex2f(1.0,4.3);
glVertex2f(1.0,3.2);
glVertex2f(1.1,3.05);
glVertex2f(1.2,2.9);
glVertex2f(1.3,2.9);
glVertex2f(1.4,2.75);
glVertex2f(1.5,2.65);
glVertex2f(1.6,2.6);
glVertex2f(1.7,2.55);
glVertex2f(1.8,2.5);
glVertex2f(1.9,2.45);
glVertex2f(2.0,2.5);
glEnd();

glColor3f(0.0,0.0,0.0);
glBegin(GL_LINE_LOOP); //outer line for design a car
glVertex2f(8.0,3.0);
glVertex2f(16.0,3.0);
glVertex2f(16.5,3.5);
glVertex2f(17.0,4.0);
glVertex2f(16.5,4.25);
glVertex2f(16.0,4.5);
glVertex2f(15.0,4.5);
glVertex2f(15.0,5.0);
glVertex2f(14.0,5.0);
glVertex2f(11.5,6.5);
glVertex2f(10.5,6.75);
glVertex2f(7.0,6.75);
glVertex2f(5.0,5.0);
glVertex2f(7.0,5.0);
glVertex2f(6.5,4.5);
glEnd();


glBegin(GL_LINES); //connecting outer line
glVertex2d(7.0,5.0);
glVertex2d(15.0,5.0);
glEnd();

glColor3f(0.0,0.0,0.0); //connecting outer line
```

```
glBegin(GL_LINES);
glVertex2d(15.0,4.0);
glVertex2d(17.0,4.0);
glEnd();

glColor3f(0.0,0.0,0.0); //connecting outer line
glBegin(GL_LINES);
glVertex2d(15.0,3.5);
glVertex2d(16.5,3.5);
glEnd();

glColor3f(0.0,0.0,0.0); //connecting outer line
glBegin(GL_LINES);
glVertex2d(15.0,5.0);
glVertex2d(14.0,3.0);
glEnd();

glColor3f(0.0,0.0,0.0); //connecting outer line
glBegin(GL_LINES);
glVertex2d(12.0,5.0);
glVertex2d(12.0,6.2);
glEnd();

glColor3f(0.0,0.0,0.0); //connecting outer line
glBegin(GL_LINES);
glVertex2d(7.0,5.0);
glVertex2d(7.0,6.7);
glEnd();

glBegin(GL_POLYGON); //drawing a back tyre
glVertex2f(3.0,2.5);
glVertex2f(3.0,2.6);
glVertex2f(3.15,3.1);
glVertex2f(3.2,3.2);
glVertex2f(3.3,3.35);
glVertex2f(3.4,3.4);
glVertex2f(3.5,3.45);
glVertex2f(3.6,3.55);
glVertex2f(3.7,3.6);
glVertex2f(3.8,3.63);
glVertex2f(4.0,3.65);
glVertex2f(4.2,3.7);
```

```
glVertex2f(18.4,3.7);
glVertex2f(18.6,3.65);
glVertex2f(18.8,3.55);
glVertex2f(19.0,3.45);
glVertex2f(19.1,3.4);
glVertex2f(19.2,3.25);
glVertex2f(19.3,3.2);
glVertex2f(19.4,3.0);

glVertex2f(19.5,2.5);
glVertex2f(19.45,2.15);
glVertex2f(19.4,1.9);
glVertex2f(19.35,1.8);
glVertex2f(19.2,1.6);
glVertex2f(19.0,1.5);
glVertex2f(18.9,1.4);
glVertex2f(18.7,1.3);
glVertex2f(18.6,1.27);
glVertex2f(18.4,1.25);
glVertex2f(18.0,1.25);
glVertex2f(17.9,1.3);
glVertex2f(17.75,1.35);
glVertex2f(17.6,1.4);
glVertex2f(17.45,1.55);
glVertex2f(17.3,1.7);
glVertex2f(17.2,1.8);
glVertex2f(17.1,2.2);
glEnd();
glPopMatrix();
}
```

4.2.1 User defined Functions

Void bus(): This function is used to create a first scene of the project which has project title and other information.

Void road(): This road function is used to draw the road.

Void signal(): This signal function is used to draw the signal.

Void car(): This car function is used to draw the car.

Void car2(): This car2 function is used to draw another car.

Void mydisplay(): This mydisplay function is used to display the introduction scene.

Void display(): This function is used for displaying the output.

Void frontscreen(): This frontscreen function is used to draw the front scene.

Void mymouse(): This function allows a smooth control of traffic signal.

Void helpscreen(): This function is used to notify the working of screen.

4.2.2 Built-in Functions

- glColor3f(GLfloat red, GLfloat green, GLfloat blue): This function sets present RGB colors. Different color is given to object using the colors parameters such as red, green, blue. The maximum and minimum values of the floating point types are 1.0 and 0.0 respectively.
- gluOrtho2D(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble near, GLdouble far): Defines the orthographic volume with all the parameters measured from the center of projection of the plane.
- glLoadIdentity(): Sets the current transformation matrix to an identity matrix.
- voidglClear(GL_COLOR_BUFFER_BIT): Clears all buffer whose bits are set in mask. The mask is formed by the logical OR of values defined in gl.h GL_COLOR_BUFFER_BIT refers to color buffers.
- glBegin() and void glEnd(): The glBegin and glEnd functions delimit the vertices of a primitive or a group of like primitives. Syntax of glBegin is as follows: glBegin(GLenum mode);
- glFlush(void): The glFlush function forces execution of OpenGL function in finite time. This function has no parameters. This function does not return a value.
- glMatrixMode(GL_PROJECTION): Projection matrixes are stored in OpenGL projection mode. So to set the projection transformation matrix. That mode is invoked through the statement, glMatrixMode(GL_PROJECTION).
- glutDisplayFunc(): Sets the display call back for the current window.
- glutPostRedisplay(): Mark the normal plane of current window as needing to be redisplayed. The next iteration through glutMainLoop, the window's display call back will be called to redisplay the window's normal plane. Multiple calls to glutPostRedisplay before the next display call back opportunity generates only a single redisplay call back. GlutPostRedisplay may be called within a window's display or over lay display call back to remark that windows for redisplay.
- glutInit(int argc, char **argv[]): The glutInit will initialize the GLUT library, the arguments form main are passed in and can used by the application. This will

negotiate a session with the window system. During this process, glutInit may cause the termination of the GLUT program with an error message to user if GLUT cannot be properly initialized.

- glutInitDisplayMode(unsigned int mode): The glutInitDisplayMode sets the initial display mode. It requests a display with the properties in mode. The value of mode is determined by the logical OR of option including the color model(GLUT_RGB), buffering(GLUT_DOUBLE) and (GLUT_DEPTH).
- glutInitWindowSize(int width, int height): This function specifies the initial height and width in pixels.
- glutInitWindowPosition(int x, int y): This function specifies the initial position of the top-left corner of the windows in pixel.
- glutCreateWindow(char *title): The glutCreateWindow create a window on the display. The string title can be used to label the window. The return value provides a reference to the window that can be used when there are multiple windows.
- glutMainLoop(): Cause the program to enter an event-processing loop. It should be the last statement in the main.
- myinit(void): Sets the background color and viewing.

CHAPTER 5

RESULTS AND SNAPSHOTS

5.1 Snapshot-1

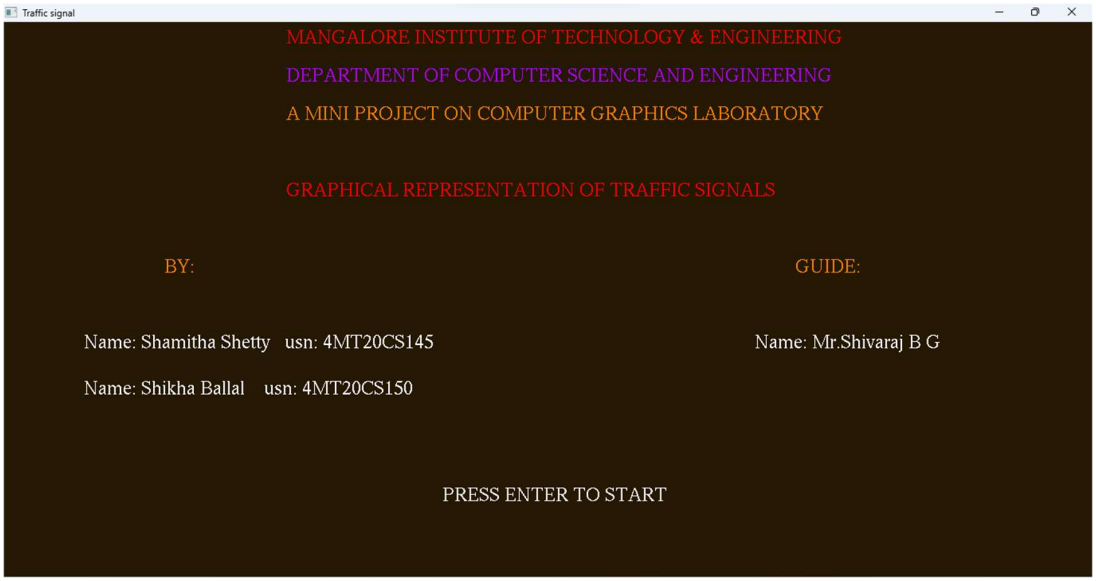


Figure 5.1: It shows the initial view of our project.

5.2 Snapshot-2

Figure 5.2: It refers to the control tips.

5.3 Snapshot-3

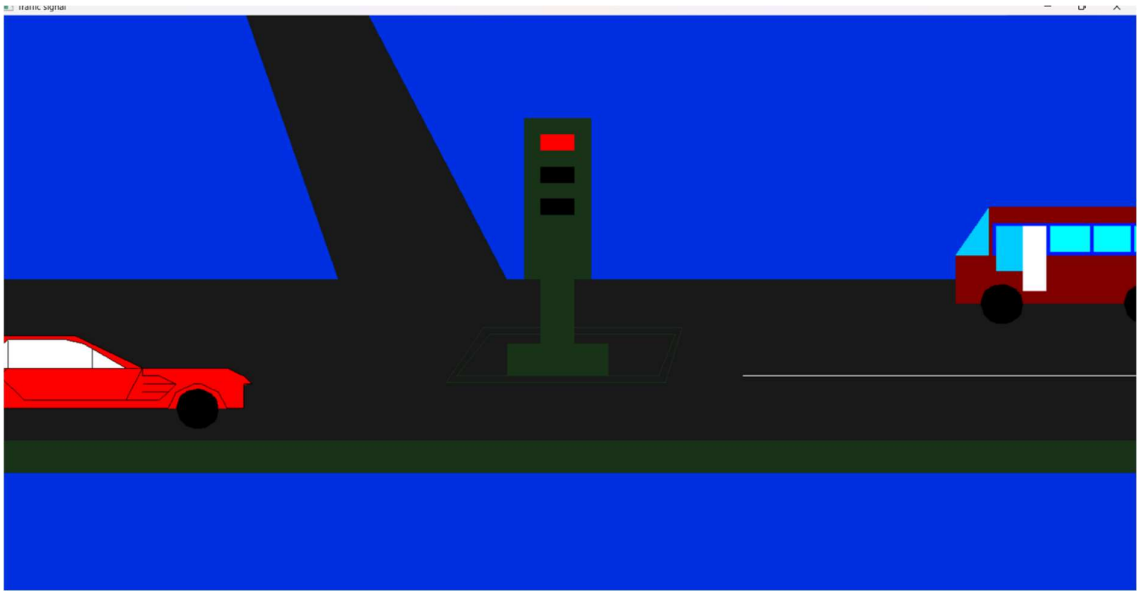


Figure 5.3: It refers to the second scene where traffic signal is red and which means to stop the vehicle.

5.4 Snapshot-4

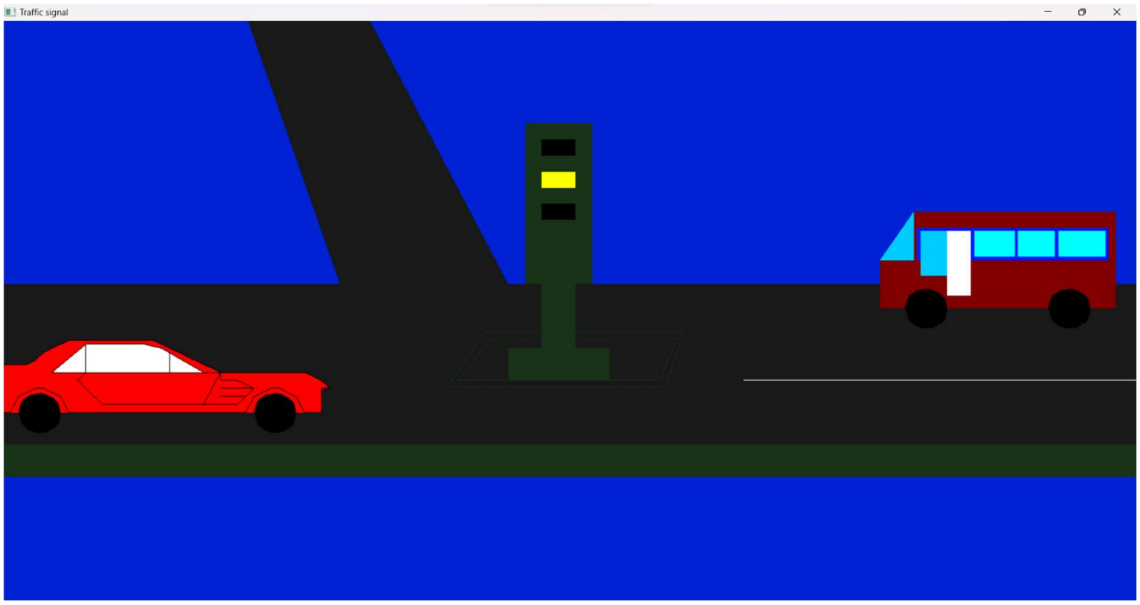


Figure 5.4: It refers to the second scene where traffic signal is yellow.

5.5 Snapshot-5

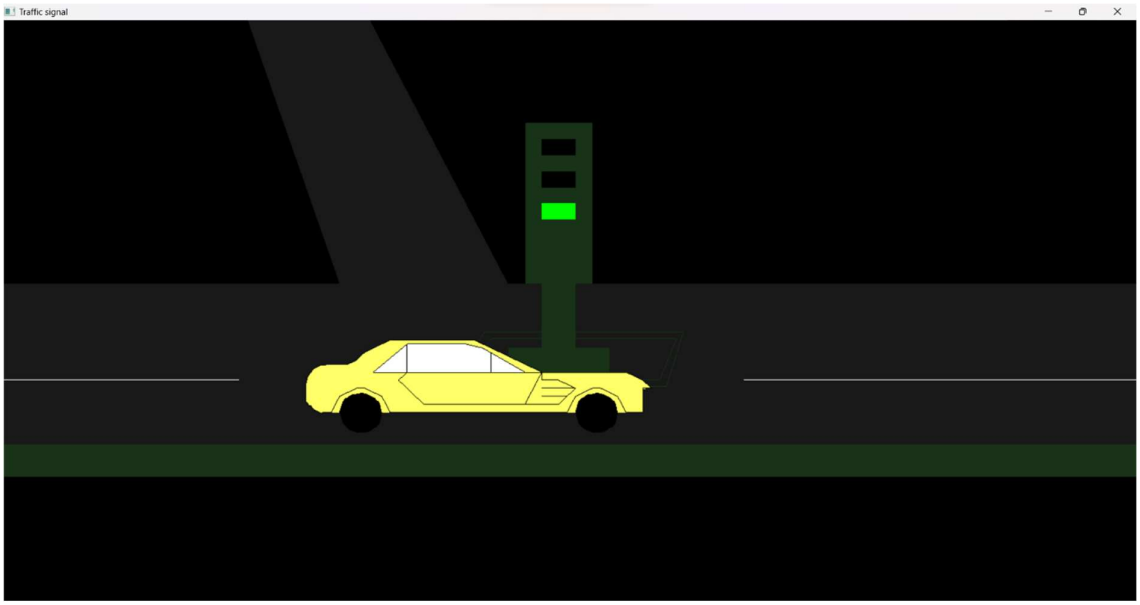


Figure 5.5: It refers to the third scene where traffic signal shows green light which means that the vehicle can go.

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

6.1 Conclusion

This project demonstrates the working of arrival and departure of the vehicles using computer graphics and different functions of OpenGL. This project mainly tries to give an idea of how the traffic signal works on the road so that we can use his graphics in some animated stories or use to demonstrate to the small children. The overall implementation of the code is done in Dev-C++ and the programming language used is OpenGL which is mainly used to design the computer graphics project.

6.2 Future Enhancement

This project shows certain simulation process which is based on moving train. A better visualization can be given using OpenGL graphics library. The proposed project uses OpenGL's graphic functions in 2D. as an improvement one can make use of some 3D effect. Compact and sophisticated code can be used.

REFERENCES

1. Edward Angel, Interactive Computer Graphics A Top-Down Approach with OpenGL, 5th Edition, Addison-Wesley, 2008.
2. https://webeduclick.com/c-program-to-draw-traffic-signal-using-graphics/