VISVESVARAYA TECHNOLOGICAL UNIVERSITY
Belagavi-590018, Karnataka

A

REPORT

ON

"TRAFFIC SIGNAL"

Submitted in partial fulfillment for the award of the degree in **Bachelor of Engineering in Computer Science & Engineering**

Submitted by

SHREYA KUMARI **[1BH20CS040]**
SOUNDARYA SINNUR **[1BH20CS043]**

Under the Guidance of

Mrs. Monica C
Assistant Professor
Department of Computer Science and Engineering

VTU Affiliated AICTE Approved BTI BANGALORE TECHNOLOGICAL INSTITUTE

Bangalore Technological Institute

BANGALORE TECHNOLOGICAL INSTITUTE

(ISO 9001:2015 Certified Institute)

Sarjapur Road, Kodathi, Bangalore-560035

Department of Computer Science & Engineering

2022-2023

BANGALORE TECHNOLOGICAL INSTITUTE
Bengaluru-35
(An ISO 9001:2015 Certified Institute)
Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Mini project report on **"TRAFFIC SIGNAL"** carried out by **SHREYA KUMARI [1BH20CS040] AND SOUNDARYA SINNUR [1BH20CS043]** the Bonafide students of **Bangalore Technological Institute**, **Bangalore** in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of **Visvesvaraya Technological University, Belgavi** during the year **2022-2023**. Thus, it is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report. The seminar report has been approved, as it satisfies the academic requirement in the respect of the seminar report prescribed for the said degree.

Mrs. Monica C
Assistant Professor
Department of CSE

Dr. Sohan Kumar Gupta
H.O.D, Department of CSE

External Viva

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

BANGALORE TECHNOLOGICAL INSTITUTE
Bengaluru-35
(An ISO 9001:2015 Certified Institute)
Department of Computer Science and Engineering

DECLARATION

We, student of sixth semester **B.E, COMPUTER SCIENCE AND ENGINEERING, BANGALORE TECHNOLOGICAL INSTITUTE, BANGALORE,** hereby declare that the Mini project on "**TRAFIC SIGNAL** " has independently carried out by me at **Bangalore Technological Institute, Bengaluru** and submitted in partial fulfillment of the requirements for the award of the degree in **Bachelor of Engineering in Computer Science & Engineering** of the **Visvesvaraya Technological University, Belagavi** during the academic year **2022-2023**. We also declare that, to the best of my knowledge and believe the work reported here does not form or part of any other dissertation on the basis of which a degree or award was conferred on an early occasion of this by any other students.

PLACE: BENGALURU

DATE:

SHREYA KUMARI **[1BH20CS040]** ……

SOUNDARYA SINNUR **[1BH20CS043]** ……

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 and elders. I would like to take this opportunity to thank them all.

We heartily extend my words of gratitude to our Mini project Coordinator , **Mrs. Monica C** , for her valuable advice, encouragement and suggestion given to me in the course of my Mini project work. We convey gratitude to her for having constantly monitored the development of the seminar report and setting up precise deadlines.

We would like to express my immense gratitude to the Head of Department **Dr. Sohan Kumar Gupta** , for his unfailing encouragement and suggestions given to me in the course of the work.

We would like to take this opportunity to express my gratitude to the Principal, **Dr. H S Nanda** , for giving me this opportunity to enrich knowledge. We are grateful to the President **Dr. A Prabhakara Reddy** and Secretary , **Sri. C L Gowda** for having provided us with a great infrastructure and well-furnished labs.

Finally, a note of thanks to the Department of Computer Science and Engineering,both teaching and non-teaching staff for their cooperation extended.

Last but not the least, We acknowledge the support and feedback of my parents guardians and friends, for their indispensable help always.

SHREYA KUMARI [1BH20CS040]

SOUNDARYA SINNUR [1BH20CS043]

ABSTRACT

A Traffic Signal is a containerization platform that allows developers to package applications and their dependencies into a single unit called a container, making it easy to deploy and run applications consistently across different environments. Traffic Signal provides a flexible and efficient way to build, ship, and run applications, simplifying the development, testing, and deployment process. With Traffic Signal, developers can create lightweight and portable containers that can run anywhere, from local development environments to production servers in the cloud. Traffic Signal has become a popular technology in the DevOps community, enabling teams to build and deploy applications faster and with greater efficiency. This abstract provides a brief overview of Traffic Signal technology and its benefits in modern software development.

TABLE OF CONTENTS

TITLE	PAGE NO.
Acknowledgement	I
Abstract	II
Content	III-IV
List of figures	V

CHAPTERS

1: INTRODUCTION

1.2 Statement of Problem	2
1.3 Objective of the Problem	2

2:LITERATURE SURVEY 3-5

3: SYATEM REQUIREMENT SPECIFICATION

3.1 Hardware Requirements	6
3.2 Software Requirements	6

4: DESIGN

4.1 Existing System	7
4.2 Propsed System	7
4.3 Low level Design	8

5: IMPLEMENTATION

5.1 Functions	9

6: RESULTS AND SCREENSHOTS 12-14

7: TESTING 15

8: CONCLUSION AND FUTURE SCOPE

8.1 : Conclusion 16
8.2 : Future Enhancement 16

APPENDIX 18-35

BIBLIOGRAPHY 36

LIST OF FUGURES

Fig. No.	Figure Name	Page No.
6.1	Home Screen	12
6.2	Starting Screen	12
6.3	Left Red light and right Green Light	13
6.4	Left Green light and right GreenLight	13
6.5	Left Yellow light and right Green Light	14
6.6	Left Yellow light and right Yellowlight	14

TEST CASE

Table No.	TABLE NAME	PAGE No.
7.1	Test cases for Keyboard interface	13

Chapter 1

INTRODUCTION

Computer graphics is conscerned with all aspects of producing picture or an image using computers and, more generally, the representation and manipulation of pictorial data by a computer. The development of computer graphics has made computers easier to interact with and better for understanding and interpreting many types of data. Developments in computer graphic had a profound impact on many types of media and have revolutionized the animation and video game industry. Today computers and computer-generated images touch many aspects of our daily life. Computer imagery is found on television, in newspapers, in weather reports, and during surgical procedures. A well-constructed graph can present complex statistics in a form that is easier to understand and interpret. Such graphs are used to illustrate papers, reports, theses, and other presentation material. A range of tools and facilities are available to enable users to visualize their data, and computer graphics are used in many disciplines. We implement computer graphics using the OpenGL API. OpenGL (Open Graphics Library) is a standard specification defining a cross-language, cross-platform API for writing applications that produce 2D and 3D computer graphics. OpenGL is a low-level graphics library specification. It makes available to the programmer a small set of geometric primitives - points, lines, polygons, images, and bitmaps. OpenGL provides a set of commands that allow the specification of geometric objects in two or three dimensions, using the provided primitives, together with commands that control how these objects are rendered (drawn). Since OpenGL drawing commands are limited to those that generate simple geometric primitives (points, lines, and polygons), the OpenGL Utility Toolkit (GLUT) has been created to aid in the development of more complicated three-dimensional objects such as a sphere, a torus, and even a teapot. GLUT may not be satisfactory for full- featured OpenGL applications, but it is a useful starting point for learning OpenGL.GLUT is designed to fill the need for a window system independent programming.

Using this editor you can draw and paint using the mouse. It can also perform a host of other functions like drawing lines, circles, polygons and so on. Interactive picture construction techniques such as basic positioning methods, rubber-band methods, dragging and drawing are used. Block operations like cut, copy and paste are supported to edit large areas of the workspace simultaneously. It is user friendly and intuitive to use.

OpenGL has historically been influential on the development of 3D accelerator, promoting a base level of functionality that is now common in consumer level hardware:

- A transform and lighting pipeline.
- Z buffering.
- Texture Mapping.
- Alpha
- Blending.

1.1 STATEMENT OF PROBLEM

A traffic signal is used as an instructing device that indicates the road user to act according to the displayed sign. Following the traffic signal ensures road safety and to make things simple to understand, these signals have been using a universal colour code. Traffic lights or traffic signals are signalling devices positioned at road intersections, pedestrian crossings, and other locations to control flows of traffic

1.2 OBJECTIVE OF THE PROBLEM

The objective is to build a Traffic signal rules are put in place to ensure the safety of all road users and to regulate the flow of traffic. The most common traffic signals in India are the red, yellow, and green lights, which are used to indicate when to stop, slow down, and proceed, respectively.

The basic requirements of the traffic signal are analyzed to be:

1) User Interface- User should be able to select a signal and start thesimulation on their own. They can start, stop and slow down the vehicles and traffic of their choice and after this they can exit the simulation.

2) Selection- User can select the element from red,yellow and green for the simulation.

3) Start/ Stop Simulation- User after selecting an element from thementioned list he/she can start the simulation. As soon as they select the green light the vehicles will start moving. And if they choose yellow vehicles needs to slow down. And if they select the red light the vehicles needs to stop.

Chapter 2

2. LITERATURE SURVEY

Computer graphics started with the display of data on hardcopy plotters and cathode ray tube (CRT) screens soon after the introduction of computers.

Computer graphics today largely interactive, the user controls the contents, structure, and appearance of objects and of displayed images by using input devices, such as keyboard, mouse, or touch-sensitive panel on the screen. Graphics based user interfaces allow millions of new users to control simple, low-cost application programs, such as spreadsheets, word processors, and drawing programs.

OpenGL (Open Graphics Library) is a standard specification defining a crosslanguage, cross-platform API for writing applications that produce 2D and 3D computer graphics. The interface consists of over 250 different function calls which can be used to draw complex three-dimensional scenes from simple primitives. OpenGL was developed by Silicon Graphics Inc. (SGI) in 1992 and is widely used in CAD, virtual reality, scientific visualization, information visualization, and flight simulation. It is also used in video games, where it competes with Direct3D on Microsoft Windows platforms (see Direct3D vs. OpenGL). OpenGL is managed by the non-profit technology consortium, the Khronos Group.

In the 1980s, developing software that could function with a wide range of graphics hardware was a real challenge. By the early 1990s, Silicon Graphics (SGI) was a leader in 3D graphics for workstations. SGI's competitors (including Sun Microsystems, Hewlett-Packard and IBM) were also able. In addition, SGI had a large number of software customers; by changing to the OpenGL API they planned to keep their customers locked onto SGI (and IBM) hardware for a few years while market support for OpenGL matured to bring to market 3D hardware, supported by extensions made to the PHIGS standard. In 1992, SGI led the creation of the OpenGL architectural review board (OpenGL ARB), the group of companies that would maintain and expand .OPENGL specification took for years to come. .

On 17 December 1997, Microsoft and SGI initiated the Fahrenheit project, which was a joint effort with the goal of unifying the OpenGL and Direct3D interfaces (and adding a scene-graph API too). In 1998 HewlettPackard joined the project.[4] It initially showed some promise of bringing order to the world of interactive 3D computer graphics APIs, but on account of financial constraints at SGI, strategic reasons at Microsoft, and general lack of industry support, it was abandoned in 1999[8]. Many opengl functions are used for rendering and transformation purposes. Transformations functions like glRotate (), glTranslate (), glScaled () can be used. OpenGL provides a powerful but primitive set of rendering command, and all higherlevel drawing must be done in terms of these commands. There are several libraries that allow you to simplify your programming tasks, including the following: OpenGL Utility Library (GLU) contains several routines that use lower-level OpenGL commands to perform such tasks as setting up matrices for specific viewing orientations and projections and rendering surfaces. OpenGL Utility Toolkit (GLUT) is a window-system-independent toolkit, written by Mark Kill guard, to hide the complexities of differing window APIs. To achieve the objective of the project, information related to the light sources is required with OpenGL we can manipulate the lighting and objects in a scene to create many different kinds of effects. It explains how to control the lighting in a scene, discusses the OpenGL conceptual model of lighting, and describes in detail how to set the numerous illumination parameters to achieve certain effects. This concept is being obtained from . To demonstrate the transformation and lightening, effects, different polygons have to be used. Polygons are typically drawn by filling in all the pixels enclosed within the boundary, but we can also draw them as outlined polygons or simply as points at the vertices. This concept is obtained from. The properties of a light source like its material, diffuse, emissive, has to mention in the project. So to design the light source and the objects, programming guide of an OpenGL is used.

ADVANTAGES

1. User-friendly.
2. Simple and interactive.
3. Multiple actions can be performed at once.
4. OpenGL is more functional than any other API
5. Reliability and Portability.

Chapter 3

SYSTEM REQUIREMENTS SPECIFICATION

Code blocks is the smart client applications by letting developers quickly create connected applications that deliver the highest qualityrich user experiences. This new version 20.03 lets any size organization create more secure, more manageable, and more reliable applications that take advantage of features provided by Code blocks.

3.1 Software Requirements

- An MS-DOS based operating system like Windows 98, Windows 2000 or WindowsXP, vista, windows 7 is the platform required to develop the 2D and 3D graphics applications.
- A Visual C/C++ compiler is required for compiling the source code to make the executable file which can then be directly executed.
- A built in graphics library like glut and glut32, and header file like GL\glut.h and also dynamic link libraries like glut and glut32 are required for creating the 3D layout.

3.2 Hardware Requirements

The hardware requirements are very minimal and the software can run on most of themachines.

- Processor - Intel 486/Pentium processor or above.
- Processor Speed - 500 MHz or above
- RAM - 64MB or above Storage Space - 2 MB or above, hard disk - 10MB.
- Monitor resolution - A color monitor with a minimum resolution of 1000*700

CHAPTER 4

DESIGN

4.1 EXISTING SYSTEM

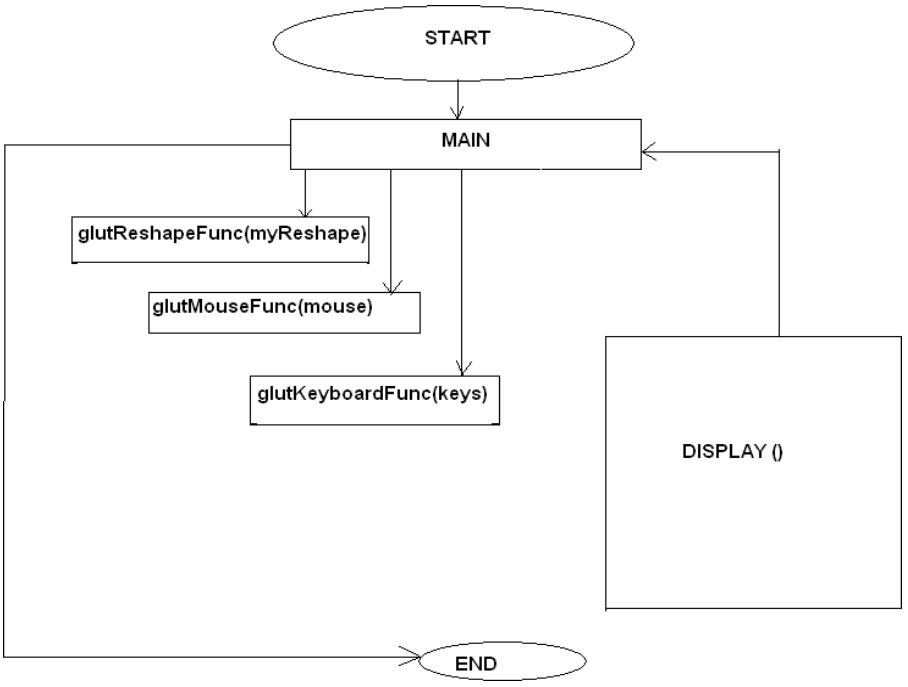
Existing system for a graphics is the TC++. This system will support only the 2D graphics. 2D graphics package being designed should be easy to use and understand. It should provide various op- tions such as free hand drawing, line drawing, polygon drawing, filled polygons, flood fill, translation, rotation, scaling, clipping etc. Even though these properties were supported, it was difficult to render 2D graphics cannot be very difficult to get a 3 Dimensional object. Even the effects like lighting, shad- ing cannot be provided. So we go for Microsoft Visual Studio software.

4.2 PROPOSED SYSTEM

To achieve three dimensional effects, open GL software is proposed. It is software which pro-vides a graphical interface. It is a interface between application program and graphics hardware. The advantages are:

1. Open GL is designed as a streamlined.
2. It‟s a hardware independent interface i.e it can be implemented on many different hardware platforms.
3. With openGL we can draw a small set of geometric primitives such as points, lines and polygons etc.
4. It provides double buffering which is vital in providing transformations.
5. It is event driven software.

4.3 Low level design



CHAPTER 5

IMPLEMENTATION

5.1 Functions

The **glColor3f (float, float, float) :-**This function will set the current drawing color .

gluOrtho2D (GLdouble left, GLdouble right, GLdouble bottom, GLdouble top):-whichdefines a two dimensional viewing rectangle in the plane z=0.

glClear( ):- Takes a single argument that is the bitwise OR of several values indicating whichbuffer is to be cleared.

.

glClearColor ():- Specifies the red, green, blue, and alpha values used by **glClear** to clear thecolor buffers.

GlLoadIdentity( ):- the current matrix with the identity matrix.

glMatrixMode(mode):- Sets the current matrix mode, mode can be GL_MODELVIEW, GL_PROJECTION or GL_TEXTURE.

Void glutInit (int *argc, charargv):-**Initializes GLUT, the arguments from main are passed inand can be used by the application.

.

Void glutInitDisplayMode (unsigned int mode):-Requests a display with the properties in mode. The value of mode is determined by the logical OR of options including the color model and buffering.

Void glutInitWindowSize (int width, int height):-Specifies the initial position of the top-leftcorner of the window in pixels.

Int glutCreateWindow (char *title):-A window on the display. The string title can be used to label the window. The return value provides references to the window that can be used when thereare multiple windows.

Void glutMouseFunc(void *f(int button, int state, int x, int y):- Register the mouse callbackfunction f. The callback function returns the button, the state of button after the event and the position of the mouse relative to the top-left corner of the window.

Void glutKeyboardFunc(void(*func) (void)):-This function is called every time when you pressenter key to resume the game or when you press „b‟ or „B‟ key to go back to the initial screen or when you press esc key to exit from the application.

Void glutDisplayFunc (void (*func) (void)):-Register the display function func that is executedwhen the window needs to be redrawn.

Void glutSpecialFunc(void(*func)(void)):-This function is called when you press the special keys in the keyboard like arrow keys, function keys etc. In our program, the func is invoked whenthe up arrow or down arrow key is pressed for selecting the options in the main menu and when the left or right arrow key is pressed for moving the object(car) accordingly.

glut PostReDisplay ( ):-which requests that the display callback be executed after the currentcallback returns.

Void MouseFunc (void (*func) void)):-This function is invoked when mouse keys are pressed.This function is used as an alternative to the previous function i.e., it is used to move the ob ject(car) to right or left in our program by clicking left and right button respectively.

Void glutMainLoop ()

Cause the program to enter an event-processing loop. It should be the last statement in mainfunction.

Chapter 6

RESULTS & SNAPSHOTS

Figure 6.1: HOME SCREEN

User can move to the arena by pressing the Enter key.

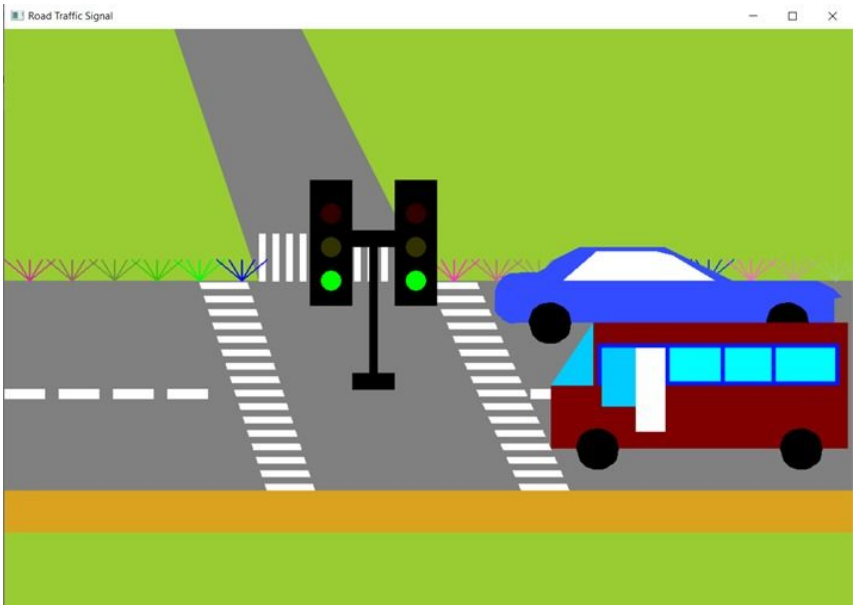


Figure 6.2 : STARTING SCREEN

Initial screen of the arena where user can use traffic signal stimulation.

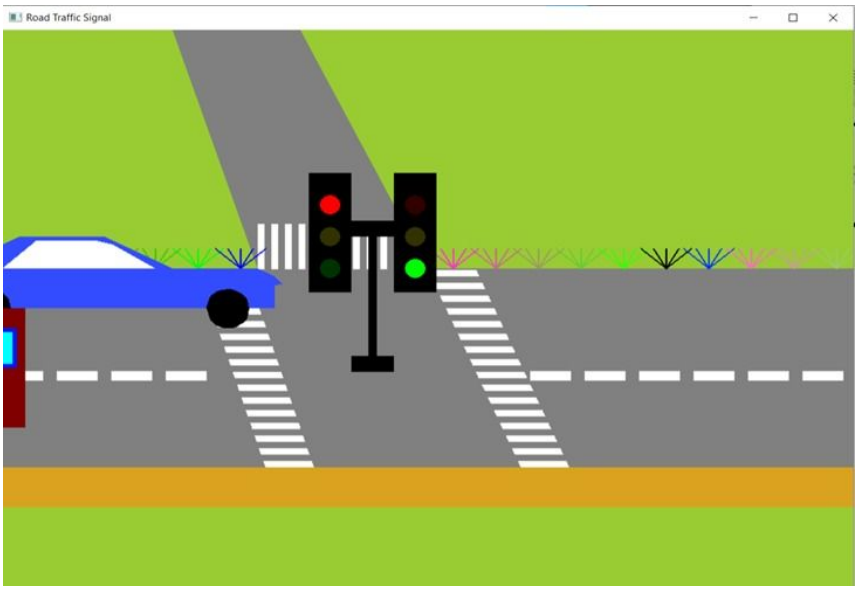


Figure 6.3: Left red light and right green light

Here user can use red and green signals while having traffic.

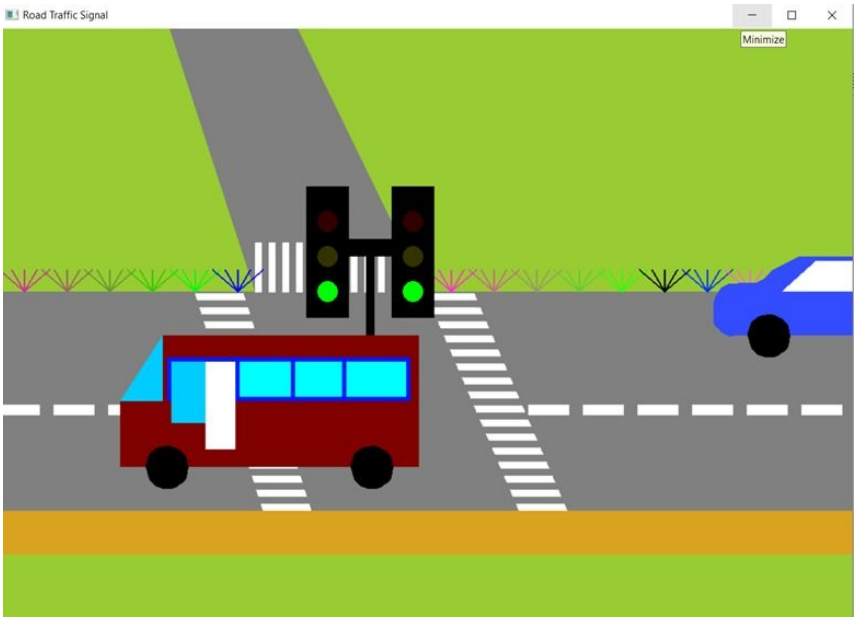


Figure 6.4: Left green light and right green light

Here user can expenience both side green signals while having traffic.

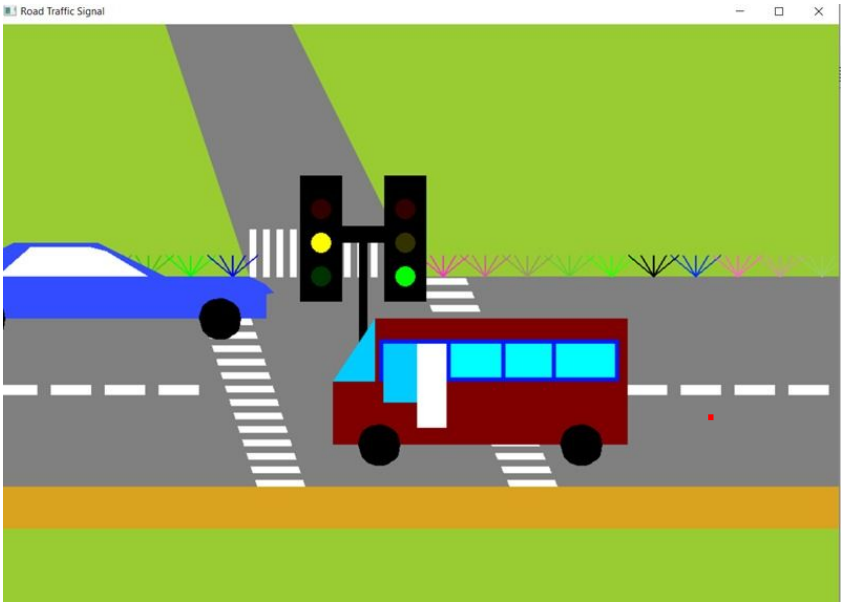


Figure 6.5: Left yellow light and right green light

Here user can use yellow and green signals while having traffic.

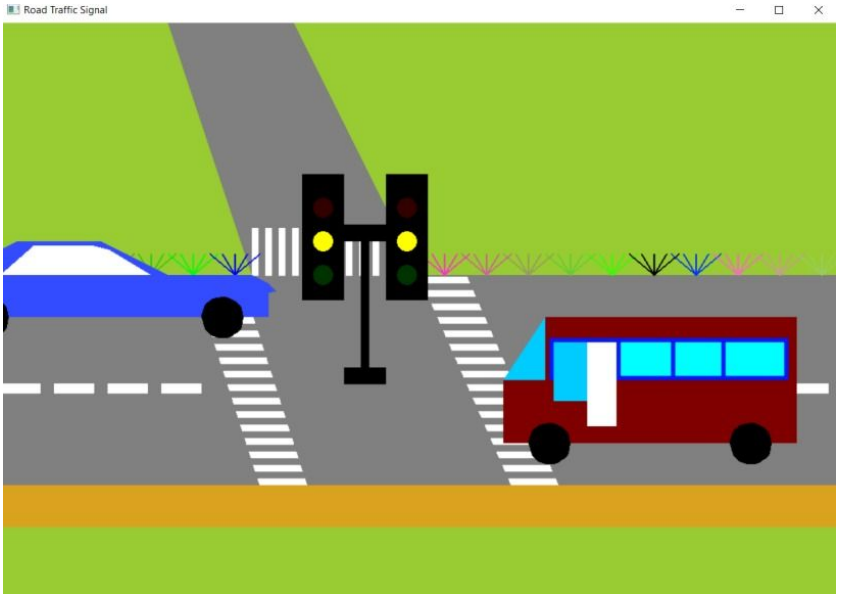


Figure 6.6: Left yellow light and right yellow light

Here user can experience both sides Yellow signals while having traffic.

Chapter 7

TESTING

Sl. No	Functionality	Comments	Remarks
1.	Choosing the options	It shows the list of option to user from which they can select an option.	
2.	1. Simulate	Starts the simulation. (Space bar)	Pass
3.	2. Red light	Press („Q“) for left and Press („A“) for right	Pass
4.	3. Yellow light	Press („W“) for left and Press („S“) for right	Pass
5.	4. Green light	Press („E“) for left and Press („D“) for right	Pass

Table 7.1 Test cases for Keyboard interface

Chapter 8

CONCLUSION AND FUTURE SCOPE

8.1 CONCLUSION

This traffic light is very good project. Users can very easily understand the signal of the traffic light. The user interface supports keyboard interface. We have tried our best to make thissimulator very realistic, so that user can easily understand the concepts of red light,yellow light and green light.Here „red‟ indicates that the vehicles must stop,

„yellow‟ means that the vehicles must slow down and finally „green‟ means to go ahead. The colours red, yellow and green have been chosen as traffic lights due to their visibility and wavelength compared to the other colours. Traffic signal rules are simple,uncomplicated and the same everywhere in the world.

8.2 FUTURE ENHANCEMENTS

The following are some of the features that are planned to be supported inthe futureversions of the atom simulator.

- **Integration with Smart City Infrastructure:**

Integrate the traffic signalproject with other smart city infrastructure components, such as intelligent transportation systems, traffic management centers, and connected vehicles.

- **Energy Efficiency and Sustainability:** Optimize the traffic signal system forenergy efficiency by using low-power components and implementing intelligent power management techniques.
- **Visualization and Augmented Reality:** Develop interactive visualizations oraugmented reality interfaces to

- provide real-time information to drivers, pedestrians, and cyclists.
- **Integration with Autonomous Vehicles:** Account for the increasing prevalence of autonomous vehicles by incorporating communication protocols andinfrastructure support that facilitate the safe and efficient movement of self-drivingcars through traffic signals.

APPENDIX

```
#include "GL/freeglut.h"
#include "GL/gl.h"
#include <iostream>
#define ESCAPE 27
#define RED 0
#define YELLOW 1
#define GREEN 2
#define drawOneLine(x1,y1,x2,y2)  glBegin(GL_LINES);  \
   glVertex2f ((x1),(y1)); glVertex2f ((x2),(y2)); glEnd();
using namespace std;
GLint crx = -500, blx = -300;
int view=0;
bool rl1[] = {false, false, true};
bool rl2[] = {false, false, true};
void text(int x, int y, string s, int font) {
        int i=0;
        glColor3f(0.0,0.0,0.8);
        glRasterPos2f(x,y);
        for(i=0;i<s.length();i++) {
                if(font==1)
                        glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,s[i]);
                else if(font==2)
                        glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18,s[i]);
                else if(font==3) {
                        glColor3f(1.0,0.0,0.0);
                        glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18,s[i]);
                }
        }
}
void First_win() {
        glClearColor(1.0,1.0,0.6,1.0);
        text(290,700,"ROAD TRAFFIC SIGNAL",1);

text(390,660,"Using OpenGL",1);
        text(430,620,"Made By:",2);
        text(300,580,"Shreya Kumari and Soundarya Sinnur",1);
        text(400,540,"INSTRUCTIONS::",2);
        text(100,500,"For Left Traffic Light",2);
        text(100,460,"Q -- Red Light",2);
```

```
        text(100,420,"W -- Yellow Light",2);
        text(100,380,"E -- Green Light",2)
        text(560,500,"For Right Traffic Light",2);
        text(600,460,"A -- Red Light",2);
        text(600,420,"S -- Yellow Light",2);
        text(600,380,"D -- Green Light",2);
        text(200,100,"!!!!PRESS SPACEBAR TO ENTER INTO THE ARENA!!!!",3);
        glutPostRedisplay();
        //glutSwapBuffers();
}
void init() {
        glClearColor(0, 0, 0, 0);
        glPointSize(5.0);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        glOrtho(0.0,1000,0.0,800,50.0,-50.0);
        glutPostRedisplay();
}
void car_chalao() {
        if(rl1[GREEN] || crx > 0)
                crx += 1;
        if(!rl1[GREEN] && (crx > -100 && crx < 0)) {
                crx -= 1;
        }
        if(rl1[RED] && crx < -100)
                crx += 1;
        if(rl1[YELLOW] && (crx > 0 || crx < -100))
                crx += 1;
        if(crx > 1200) {

crx = -300;
        }
}
void bus_chalao() {
        // blx -= 2;
        if(rl2[GREEN] || blx > 700) {
                blx -=1;
        }
        if(!rl2[GREEN] && (blx < -650 && blx > -720)) {
                blx += 1;
        }
        if((rl2[RED] || rl2[YELLOW]) && (blx > -650 || blx < -720)) {
                blx -= 1; }
```

```
if(blx < -1650) {
        blx = -300; }

void bus() {
glPushMatrix();
glTranslated(blx,-100,0);
glScaled(50.0,50.0,0.0);
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(blx+1220,250,0.0);
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();
glPushMatrix();//back tyre
glTranslated(blx+1460,250,0.0);
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();
}
void car() {
        glPushMatrix(); //making color for outer line
        glLineWidth(1.0);
        glTranslated(crx,400.0,0.0);
        glScaled(20.0,20.0,0.0);
        glColor3f(0.2,0.3,1.0);
        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(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);
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 traffic_light() {
        //traffic signal controller
        glPushMatrix();
        glTranslatef(-190,20,0);
        glColor3f(0.0,0.0,0.0);
        glBegin(GL_POLYGON);
        // glColor3f(0.7,0.3,0.0);
                glVertex2d(600,350);  //1
                glVertex2d(650,350);  //2
                glVertex2d(650,370);  //3
                glVertex2d(600,370);  //12
        glEnd();
        glPopMatrix();
        glPushMatrix();
        glTranslatef(-190,20,0);
        glBegin(GL_POLYGON);
        // glColor3f(0.6,0.2,0.0);
                glVertex2d(630,370);  //4
                glVertex2d(630,520);  //5
                glVertex2d(620,520);  //10
                glVertex2d(620,370);  //11
        glEnd();
        glPopMatrix();
        // Left Red Light
        glPushMatrix();
        glTranslatef(-190,20,0);
        glBegin(GL_POLYGON);
        // glColor3f(0.7,0.3,0.0);
                glVertex2d(600,450);  //6
```

```
        glVertex2d(600,600);  //7
        glVertex2d(550,600);  //8
        glVertex2d(550,450);  //9
glEnd();
glPopMatrix();
//Right Red Light
glPushMatrix();
glTranslatef(-190,20,0);
glBegin(GL_POLYGON);
// glColor3f(0.7,0.3,0.0);
        glVertex2d(700,450);  //11
        glVertex2d(700,600);  //12
        glVertex2d(650,600);  //13
        glVertex2d(650,450);  //14
glEnd();
glPopMatrix();
//Connecting Rod
glPushMatrix();
glTranslatef(-190,20,0);
glBegin(GL_POLYGON);
// glColor3f(0.7,0.3,0.0);
        glVertex2d(650,520);  //15
        glVertex2d(650,540);  //16
        glVertex2d(600,540);  //17
        glVertex2d(600,520);  //18
glEnd();
glPopMatrix();
//Left Lights
//Red 1
glPushMatrix();
glTranslatef(385, 580, 0);
if(rl1[RED])
        glColor3f(1.0, 0.0, 0.0);
Else
        glColor3f(0.2, 0.0, 0.0);
glutSolidSphere(12, 80, 80);
glPopMatrix();
//Yellow 1
glPushMatrix();
glTranslatef(385, 540, 0);
if(rl1[YELLOW])
        glColor3f(1.0, 1.0, 0.0);
Else
```

```
        glColor3f(0.2, 0.2, 0.0);
        glutSolidSphere(12, 80, 80);
        glPopMatrix();
        //Green 1
        glPushMatrix();
        glTranslatef(385, 500, 0);
        if(rl1[GREEN])
               glColor3f(0.0, 1.0, 0.0);
        Else
               glColor3f(0.0, 0.2, 0.0);
        glutSolidSphere(12, 80, 80);
        glPopMatrix();
        //Right Lights
        //Red 2
        glPushMatrix();
        glTranslatef(485, 580, 0);
        if(rl2[RED])
               glColor3f(1.0, 0.0, 0.0);
        Else
               glColor3f(0.2, 0.0, 0.0);
        glutSolidSphere(12, 80, 80);
        glPopMatrix();
        //Yellow 2
        glPushMatrix();
        glTranslatef(485, 540, 0);
        if(rl2[YELLOW])
               glColor3f(1.0, 1.0, 0.0);
        Else
               glColor3f(0.2, 0.2, 0.0);
        glutSolidSphere(12, 80, 80);
        glPopMatrix();
        //Green 2
        glPushMatrix();
        glTranslatef(485, 500, 0);
        if(rl2[GREEN])
               glColor3f(0.0, 1.0, 0.0);
        Else
               glColor3f(0.0, 0.2, 0.0);
        glutSolidSphere(12, 80, 80);
        glPopMatrix();
}
void gra(int x,int y,int p,int q,float r,float g,float b)
{
```

```
        glPushMatrix();
        glColor3f(r,g,b);
        glLineWidth(2.0);
        glBegin(GL_LINES);
                glVertex2d(x,y);
                glVertex2d(p,q);
        glEnd();
        glPopMatrix();
}
void grass()
{
        float r=1.0,g=0.0,b=0.8;
        for(int k=0;k<300;k+=50)   ////left grass
        {
                if(r>0)
                        r-=.2;
                Else
                        r=1.0;
                if(g<1)
                        g+=.2;
                Else
                        g=0.0;
                if(b>0)
                        b-=.2;
                Else

                          b=1.0;
                for(int i=0, j=525,c=0;c<5;i+=15,j-=5,c++)
                        gra(i+k,525,30+k,500,r,g,b);
        }
        for(int k=0;k<500;k+=50)   //right grass
        {
                if(r>0)
                        r-=.2;
                Else
                        r=1.0;
                if(g<1)
                        g+=.2;
                Else
                        g=0.0;
                if(b>0)
                        b-=.2;
                Else
```

```
                        b=1.0;
                for(int i=500,c=0;c<5;i+=15,c++)
                        gra(i+k,525,530+k,500,r,g,b);
        }
}
void white_strips() {
        //Left
        glPushMatrix();
        glColor3f(1.0, 1.0, 1.0);
        glEnable(GL_LINE_STIPPLE);
        glLineWidth(12.0);
        glLineStipple (4, 0x0FFF);
    drawOneLine (0.0, 365.0, 250.0, 365.0);
        glDisable(GL_LINE_STIPPLE);
        glPopMatrix();

//Right
        glPushMatrix();
        glColor3f(1.0, 1.0, 1.0);
        glEnable(GL_LINE_STIPPLE);
        glLineWidth(12.0);
        glLineStipple (4, 0x0FFF);
    drawOneLine (620.0, 365.0, 1500.0, 365.0);
        glDisable(GL_LINE_STIPPLE);
        glPopMatrix();
}
void zebra() {
        //Left
        glPushMatrix();
        glColor3f(1.0, 1.0, 1.0);
        glEnable(GL_LINE_STIPPLE);
        glLineWidth(12.0);
        glLineStipple (1, 0x00FF);
        for(float i=360.0, j=280.0, c=0; c<10; i-=5, j-=5, c++) {
                drawOneLine (i, 250.0, j, 500.0);
        }
        glDisable(GL_LINE_STIPPLE);
    glPopMatrix();
        //Right
        glPushMatrix();
        glColor3f(1.0, 1.0, 1.0);
        glEnable(GL_LINE_STIPPLE);
        glLineWidth(12.0);
```

```
        glLineStipple (1, 0x00FF);
        // drawOneLine (660, 250.0, 550, 500.0);
        for(float i=660.0, j=550.0, c=0; c<10; i-=5, j-=5, c++) {
               drawOneLine (i, 250.0, j, 500.0);
        }
        glDisable(GL_LINE_STIPPLE);
        glPopMatrix();

            //Top
        glPushMatrix()
        glColor3f(1.0, 1.0, 1.0);
        glEnable(GL_LINE_STIPPLE);
        glLineWidth(12.0);
        glLineStipple (1, 0x00FF);
        // drawOneLine (300, 505.0, 500, 505.0);
        for(float i=550.0, j=550.0, c=0; c<10; i-=5, j-=5, c++) {
               drawOneLine (300, i, 500, j);
        }
        glDisable(GL_LINE_STIPPLE);
        glPopMatrix();
}
void road_strips() {
        //first bottom green strip
        glPushMatrix();
        glBegin(GL_POLYGON);
        glColor3f(0.85,0.64,0.12);
               glVertex2d(0,200);
               glVertex2d(0,250);
               glVertex2d(1000,250);
               glVertex2d(1000,200);
        glEnd();
        glPopMatrix();
        //second grey strip
        glPushMatrix();
        glBegin(GL_POLYGON);
        glColor3f(0.5,0.5,0.5);
               glVertex2d(0,250);
               glVertex2d(0,500);
               glVertex2d(1000,500);
               glVertex2d(1000,250);
        glEnd();
        glPopMatrix();
        //tilted grey strip
```

```
        glPushMatrix();
        glBegin(GL_POLYGON);
        glColor3f(0.5,0.5,0.5);
                glVertex2d(300,500);
                glVertex2d(200,800);
                glVertex2d(350,800);
                glVertex2d(500,500);
        glEnd();
        glPopMatrix();
        zebra();
        //Grass
        grass();
        //Dotted strips
        white_strips();
        car();
        traffic_light();
        bus();
        car_chalao();
        bus_chalao();
}
void main_display() {
        glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
        glMatrixMode(GL_MODELVIEW);
        glLoadIdentity();
        if(view==0) {
                init();
                First_win();
        }
        else {
                //---------CONSTANT PARTS
                //ROAD STRIP
                road_strips();
        }
        glutSwapBuffers();
}
void traffic_start() {
}
void keyboard(unsigned char key, int x, int y) {
        switch(key) {
                case ESCAPE:exit(1);
                  case ' ':
                        view=1;
                        glClearColor(0.6, 0.8, 0.2, 0);
```

```
                //glColor3f(0.85,0.64,0.12);
                traffic_start();
                break;
        case 'q':
                cout<<"Red light 1 to RED"<<endl;
                rl1[RED] = true;
                rl1[YELLOW] = false;
                rl1[GREEN] = false;
                break;
        case 'w':
                cout<<"Red light 1 to YELLOW"<<endl;
                rl1[RED] = false;
                rl1[YELLOW] = true;
                rl1[GREEN] = false;
                break;
        case 'e':
                cout<<"Red light 1 to GREEN"<<endl;
                rl1[RED] = false;
                rl1[YELLOW] = false;
                rl1[GREEN] = true;

                  break;
        case 'a':
                cout<<"Red light 2 to RED"<<endl;
                rl2[RED] = true;
                rl2[YELLOW] = false;
                rl2[GREEN] = false;
                break;
        case 's':
                cout<<"Red light 2 to YELLOW"<<endl;
                rl2[RED] = false;
                rl2[YELLOW] = true;
                rl2[GREEN] = false;
                break;
        case 'd':
                cout<<"Red light 2 to GREEN"<<endl;
                rl2[RED] = false;
                rl2[YELLOW] = false;
                rl2[GREEN] = true;
                break;
        default:
                cout<<"You pressed: "<<key;
}}
```

```
void reshape(int w,int h) {
        glViewport(0,0,w,h);
        cout<<"Width="<<w<<" height= "<<h<<endl;
}
         int main(int argc,char** argv) {
         glutInit(&argc,argv);
         glutInitDisplayMode(GLUT_RGBA|GLUT_DEPTH|GLUT_DOUBLE);
         glutInitWindowSize(1000,800);
         glutInitWindowPosition(0,0);
         glutCreateWindow("Road Traffic Signal");
         glutDisplayFunc(main_display);
         glutReshapeFunc(reshape);
         glutKeyboardFunc(keyboard);
         glutIdleFunc(main_display);
         glutMainLoop();
         return 0;
}
```

BIBLIOGRAPHY

BOOKS

- Edward Angel's Interactive Computer Graphics Pearson Education 5th Edition.
- Interactive computer Graphics --A top down approach using open GL— by EdwardAngel.
- Jackie.L.Neider,MarkWarhol,Tom.R.Davis,"OpenGLRed Book",Second. Revised Edition, 2005.
- Donald D Hearn and M.PaulineBaker,"Computer Graphics with OpenGL", 3rd Edition.

WEBSITES

- www.OpenGL Redbook.com
- www.OpenGL simple examples.
- www.OpenGL programming guide.
- http://www.wikipedia.com
- http://basic4gl.wikispaces.com
- http://www.opengl.org
- http://pyopengl.sourceforge.net/documentation/manual/glut.3GLUT.htm