ACKNOWLEDGEMENT

The successful completion of any job is incomplete without mentioning the names of those who made it possible. We are indebted to the people who helped us to complete the project.

We express our thanks to the Principal **Dr. A. S. Deshpande**, for extending his support. We also extend our thanks to our Head of Department **Dr. S.F. Rodd** for providing us with the support we needed and providing excellent lab facilities.

We thank to our respected guide **Prof. Ajay Acharya** and **Prof. Arati.S.Shahapurkar** who supported us throughout the project. They have been kind enough to extend help to us in development of our project work, without whom this project would not been completed on time. Hence it is my duty to offer our gratefulness to them for taking deep interest in this great pursuit and played key role in successfully completing our project.

We would also like to express our deep appreciation to all our well-wishers and Friends for their support and much needed encouragement.

Shubham U

Shiavaprasad Nadagoudr

ABSTRACT

The simulation of traffic involves the modeling of a complex system that is open, shows emergent phenomena, and non-linear relationships. This project outlines the ideas involved in traffic simulation and reports on the results of a traffic simulator designed to model congestion amongst urban and suburban roads.

Overall this project successfully shows a relationship between the amount of congestion present in a traffic network and the mean speed of the vehicles in the network, and the effects of various different intersection controllers on traffic flow.

It is discovered that the best method for maintaining high mean speeds as traffic flows through intersections is to use a mixture of intersection controllers on connected roads to vary the flow of traffic on each road.

In this project we present the simulation of the traffic signal by making use of the built in functions in the header files including glut, stdio etc. for Fragmentation, Rasterization, Polygon filling, and Animation. For the interactive working of the program like stopping and directing the vehicles the mouse and keyboard.

CONTENTS:

CHAPTER 1

INTRODUCTION

1.1 Introduction to Computer Graphics

OpenGL is a software interface to graphics hardware. This interface consists of about 150 distinct commands that you use to specify the objects and operations needed to produce interactive three-dimensional applications.

OpenGL is designed as a streamlined, hardware-independent interface to be implemented on many different hardware platforms. To achieve these qualities, no commands for performing windowing tasks or obtaining user input are included in OpenGL; instead, you must work through whatever windowing system controls the particular hardware you're using. Similarly, OpenGL doesn't provide high-level commands for describing models of three-dimensional objects. Such commands might allow you to specify relatively complicated shapes such as automobiles, parts of body, airplanes or molecules. With OpenGL, you must build up your desired model from a small set of geometrical primitives – points, lines and polygons.

GLUT is the OpenGL Utility Toolkit, a window system independent toolkit for writing OpenGL programs. It implements a simple windowing application programming interface (API) for OpenGL. GLUT makes it considerably easier to learn about and explore OpenGL programming. GLUT provides a portable API so you can write a single OpenGL program that works across all PC and workstation OS platforms.

GLUT is designed for constructing small to medium sized OpenGL programs. While GLUT is well-suited to learning OpenGL and developing simple OpenGL applications, GLUT is not full-featured toolkit so large applications requiring sophisticated user interfaces are better off using native window system toolkits. GLUT is simple, easy and small.

The GLUT library has both C and C++ and even FORTRAN and ADA programming bindings. The GLUT source code distribution is portable to nearly all OpenGL implementations and platforms.

CHARECTERISTICS:

- OpenGL is a better documented API.
- OpenGL is also a cleaner API and much easier to learn and program.
- OpenGL has the best demonstrated 3D performance for any API.
- Microsoft's Direct3D group is already planning a major API change called Direct Primitive that will leave any existing investment in learning Direct3D immediate mode largely obsolete.

Advantages of Open GL:-

Industry Standard: An independent consortium, the OpenGL Architecture Review Board, Guides the OpenGL specification. With broad industry support, the OpenGL API is the only truly open, stable, vendor-neutral, multiplatform graphics standard. OpenGL implementation has been available for more than nine years on wide variety of platforms. Additions to the specification are well controlled, and proposed updates are announced in time for developers to adopt changes. Backward-compatibility requirements ensure that existing applications do not become obsolete.

Reliable and Portable: All OpenGL applications consistent visual display results on any OpenGL API-conformant hardware, regardless of operating system or windowing system.

Evolving: Because of thorough and forward-looking design, the OpenGL API allows new hardware innovations to be accessible through the API via the OpenGL extension mechanism. In this way, innovations appears in the API in a timely fashion, letting application developers and hardware vendors incorporate new features into their normal product release cycles.

Scalable: Applications based on the OpenGL API can run on systems ranging from consumer electronics to PCs, workstations and supercomputers. As a result, applications can scale to any class of machine that the developer chooses to target.

Easy to use: The OpenGL API is well structured with an initiative design and logical commands. Efficient OpenGL routines typically result in applications with fewer lines of code than those that make up programs generated using other graphics libraries and packages.

Well-documented: Numerous books have been published about the OpenGL API and a great deal of sample code is readily available, making information about the OpenGL API inexpensive and easy to obtain.

ABOUT OPENGL FUNCTION AND PRIMITIVES

OpenGL Functions:

• glutInit (int *argc, char **argv):-Initialize GLUT and processes any command line argument.

• glutInitDisplayMode (unsigned int mode):-Specifies whether to use an RGBA or color-index color mode. You can also specify whether you want a single or a double buffered window.

• glutInitWindowPosition (int x, int y):- specifies the screen location for the upper left corner of your window.

• glutInitWindowSize (int width, int size):- specifies the size in pixels, of your window.

• glutCreateWindow (char * string):-create a window OpenGL context it returns a unique identifier for the new window.

• glutDisplayFunc (void (*func) (void)):- it is the first and most important event callback function. Whenever GLUT determines the contents of the window needs to be redisplayed the callback function registered by the glutDisplayFunc () is executed. Therefore you should put all the routines u need to redraw the scene in the display callback function.

• glutPostRedisplay (void):- if your program changes the contents of the window, you have to call the function it gives glutMainLoop () a nudge to call the registered callback at its next opportunity

• glutMainLoop (void):- the very last thing you must do is call this function. Windows that have been created are now shown, and rendering to those windows is now effective event processing begins, and the registered display callback is triggered once this loop is entered it is never existed.

• glClear ():-sets the current clearing color for use in clearing color buffers in RGBA mode

• glColor3f ():-to set a color. It takes 3 parameters, all of which are floating point numbers between 0.0 and 1.0. The parameters are, in order, red, green, blue components of the color

• glVertex *():-specifies the vertex for use in describing a geometric object.

• void glBegin (GLenum mode):-marks the beginning of a vertex data list that describes the geometric primitive

• void glEnd (void):- marks the end of a vertex data list

• void glPointSize (GLfloat size):-sets the width in pixels for rendered points, size must be grater then 0.0 and by default is 1.0

• glutCreateMenu (void (*f) (int value)):-creates a top level menu that uses a callback f(), which is passed as the integer value for the menu entered. A unique identifier is returned for the menu created

• void glutAddMenuEntry (char *name, int value):- Adds an entry with name displayed to the current menu; value is returned to the menu callback.

• void glutAttachMenu (int button):- Attached the current menu to the specified mouse button (GLUT_RIGHT_BUTTON, GLUT_MIDDLE_BUTTON or GLUT_LEFT_BUTTON).

• glRasterPos2f:- specifies the raster position for pixel operations.

OpenGL Primitives:

• GL_POINTS:- Draws a point at each of the n vertices.

• GL_LINES:- Draws a series of unconnected line segments. Segments are drawn between v0 and v1, between v2 and v3, and so on. If n is odd, the last segment is drawn between vn-3 and vn-2, and vn-1 is ignored.

• GL_TRAINLGES:- Draws a series of triangles (three-sided polygons) using vertices v0, v1, v2, the v3, v4, v5, and so on. If n isn’t an exact multiple of 3, the final one or two vertices are ignored.

• GL_QUADS:- Draws a series of quadrilaterals(four-sided polygons) using vertices v0,v1,v2,v3, then v4,v45,v6,v7 and so on. If n isn’t a multiple of 4, the final one, two, three vertices are ignored.

OpenGL Libraries:

Graphics Libraries

In order to write C application using GLUT you'll need these files:

• glut.h:- This is the file you'll have to include in your source code. The common place to put this file is in GL folder which should be inside the include folder of your system.

• glut.lib (SGI version for windows) and glut32.lib (Microsoft's version):- This file must be linked to your application so make sure to put it your lib folder.

The other included header files in our projects are

• GL/gl.h:- Contains C language constants, variable type definitions, and ANSI function prototypes for OpenGL.

• stdlib.h:- It is header of the general purpose standard library of C programming language which includes functions involving memory allocation, process control, conversions and others.•stdio.h:- Which stands for "standard input /output header", is the header in the C standard library that contains macro definitions, constants, and declarations of functions and types used for various standard input and output operations.

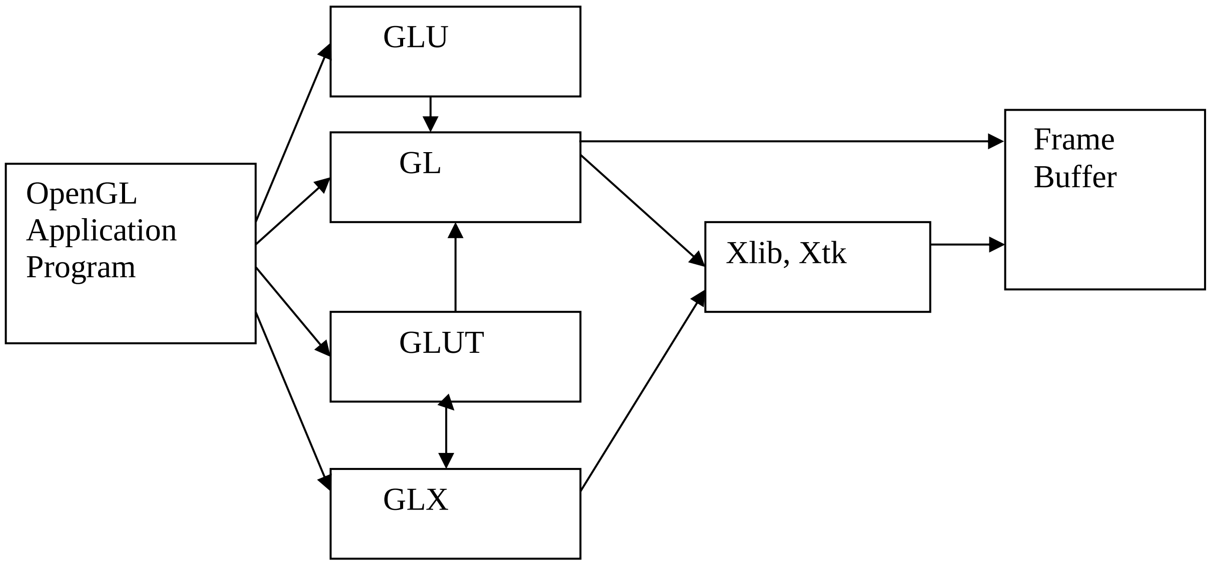


Fig 2.1: library organization of OpenGL

1.2 Motivation

The motivation for doing this project was an interest in solving a challenging problem using the Computer Graphics. This provide us opportunity to learn about a new area of computing.

The development of this project improved the knowledge about Computer Graphics and OpenGL toolkit. It also provided the good understanding of OpenGL prominent functions like translation, bounding boxes.

Keeping these factors of usability in mind the project to provides ease of use. This project will allow user to interact with scene created through the use of devices like keyboard and mouse.

1.3 Outline of the Project

Vehicle Traffic Simulations are designed to accurately model the flow of real world vehicle traffic in a graph-based network and model the development of congestion. They form a complex system that is open. Shows emergent phenomena, non-linear relationships and can contain feedback loops.

An accurate and realistic traffic simulation could aide in finding solutions for the easing of traffic congestion that in turn could aide and improve traffic flows in real world. Traffic lights, which may also be known as stoplights, traffic signals, signal lights are signaling devices positioned at road intersections, pedestrian crossings and other locations to control competing flows of traffic. Traffic lights alternate the right of way of road users by displaying lights of a standard color (red, yellow, green).

In the typical sequence of colored lights:

1. Illumination of the green light allows traffic to proceed in the direction denoted,
2. Illumination of the yellow light denoting, if safe to do so, prepare to stop short of the intersection, and
3. Illumination of the red signal prohibits any traffic from proceeding.

CHAPTER 2

PROBLEM DEFINITION AND REQUIREMENTS

2.1 Problem Definition

Program to demonstrate Simulation of Traffic Signal in 2D using OpenGL. Controlling the vehicles by lightening the Red, Yellow and Green light is done with interactive functions including mouse and keyboard functions.

2.2 Requirements Specification

User Requirements:

- Easy to understand and should be simple.
- The built-in functions should be utilized to maximum extent.
- OpenGL library facilities should be used.

Hardware Specifications:

- Processor- Intel or AMD (Advanced Micro Devices)
- RAM- 512MB (minimum)
- Hard Disk-1MB (minimum)
- Mouse, Keyboard, Monitor

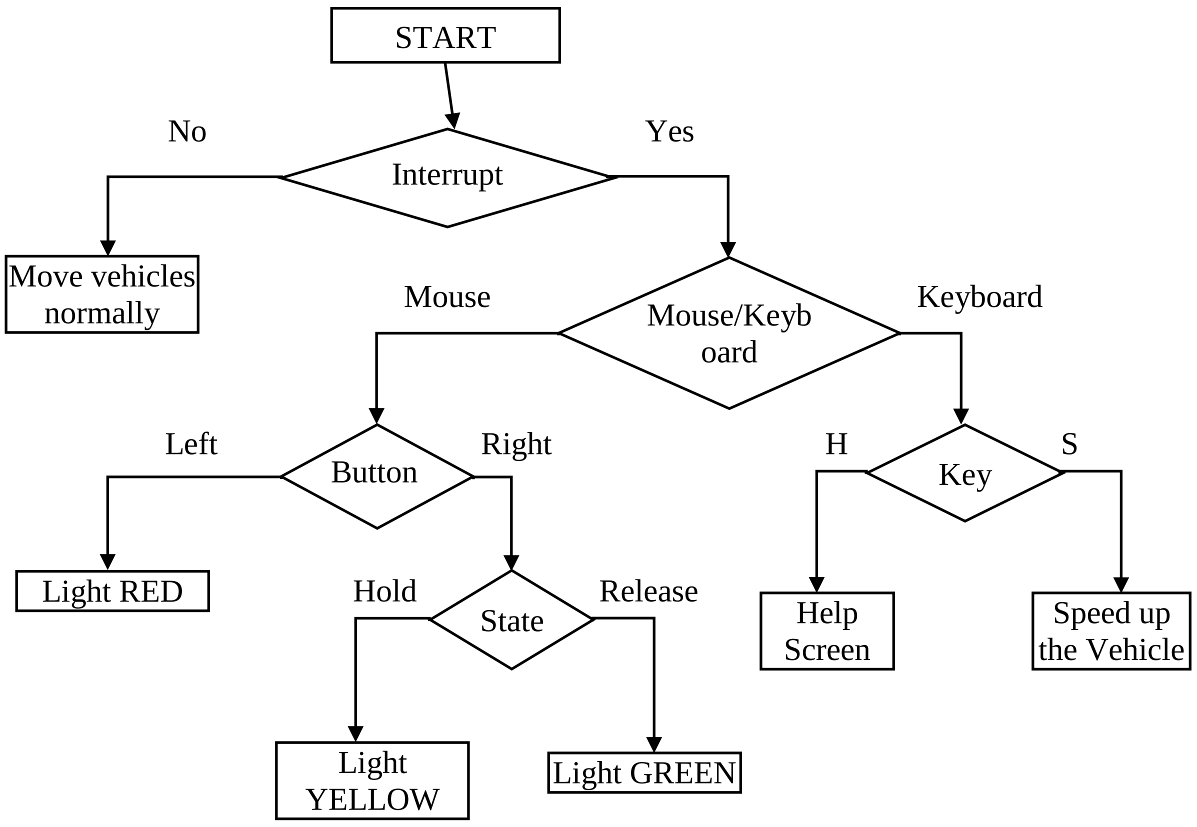
Software Specifications:

- Operating System: Ubuntu, Windows 98 and Windows XP.
- Languages Used: C
- Compiler: DEV-cpp (Windows Environment) and Kdeveloper (LINUX environment)

CHAPTER 3

DESIGN AND IMPLEMENTATION

3.1 User Interface Design



3.2 Implementation of Project

Main function

This main function is capable of handling the arguments given in the argument list at the command prompt as we have used variable ‘argc’ for total number of arguments and ‘argv’ for the array of argument list.

Main function initializes the Display Mode, Window Size and position. Then it invokes the display function within glutDisplayFunc () as the call back function.

```
void main (int argc, char* argv[])
{
        glutInit (&argc, argv);
        glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB);
        glutInitWindowSize(1346,728);
        glutInitWindowPosition(0,0);
        glutCreateWindow("Traffic signal");
        glutDisplayFunc(mydisplay);
        /*call back functions*/
        glutKeyboardFunc (myKeyboard);
        glutMouseFunc (myMouse);
        myinit();
        glutMainLoop();
}
```

Keyboard function

This Interactive function is invoked at the main program within glutKeyboardFunc() and it is repeatedly called during the execution of the program and hence handles the keyboard interrupts from the users. When a key is been pressed the ASCII code of the key and the position of the screen pointer at the time of interruption will be sent to the function.

```
void myKeyboard( unsigned char key, int x, int y )
{
        switch(key)
        {
                case 13:
                        if(flag==1)
                        {
                                flag=2;
                                mydisplay();
                        }
                        if(flag==0) //Ascii of 'enter' key is 13
```

```
                    {
                            flag=1;
                            mydisplay();
                    } break;

            case 's': mydisplay();break;
            case 'h': flag=1;mydisplay(); break;

            default: break;
    }
}
```

Mouse function

This Interactive function is invoked at the main program within glutMoseFunc() and it is repeatedly called during the execution of the program and hence handles the Mouse interrupts from the users. When a button is been pressed the identity of the button and the state of the button and the position of the screen pointer at the time of interruption will be sent to the mouse function.

```
void myMouse(int button,int state,int x,int y)
{
        if(button==GLUT_LEFT_BUTTON && state==GLUT_DOWN)
        {
                traffic_regulator=0;
                p=1;q=0;r=0;
        }
        if(button==GLUT_RIGHT_BUTTON && state==GLUT_DOWN)
        {
                traffic_regulator=0;
                p=0;q=1;r=0;
        }
        if(button==GLUT_RIGHT_BUTTON && state==GLUT_UP)
        {
                traffic_regulator=1;
                p=0;q=0;r=1;
        }
        glutPostRedisplay();
}
```

Displaying text on the screen

The setFont() function is used to set the type of the font we are using. The drawstring() function takes the position of the string to be displayed on the screen in X Y Z coordinates and the string to display.

```
void *currentfont;
void setFont(void *font)
{
        currentfont=font;
}

void drawstring(float x,float y,float z,char *string)
{
        char *c;
        glRasterPos3f(x,y,z);
        for(c=string;*c!='\0';c++)
        {
                 glColor3f(0.0,0.0,0.0);
                glutBitmapCharacter(currentfont,*c);
        }
}

void frontscreen(void)
{
        setFont(GLUT_BITMAP_TIMES_ROMAN_24);
        glClearColor(0.15,0.1,0.01,0);/*background for cover page*/
        glClear(GL_COLOR_BUFFER_BIT);
        glColor3f(1,0,0);
        drawstring(450.0,700.0,0.0,"BEARYS INSTITUTE OF TECHNOLOGY ");
        glFLush();
}
```

Display functions

We have used three display functions in this program the first one that is the myDisplay function is called in the main program using glutDisplayFunc() call back function. The mydisplay function controls the displaying of the front screen, help screen or the display function. The second display function that is display() calls the objects like road(),car() etc. According to the order that we have written hence it indirectly handles the depth information.

```
void mydisplay(void)
{
        glClear(GL_COLOR_BUFFER_BIT);
        if(flag==0)
                frontscreen ();
        if(flag==1)
                helpscreen();
        if(flag==2)
                display();
        glutSwapBuffers();
        glutPostRedisplay();
}

void display(void)
{
        if(traffic_regulator)
                glutTimerFunc(50,update,0);
        glClear(GL_COLOR_BUFFER_BIT);
        glClearColor(red,green,blue,0);/*back ground for sky*/
        road();
        signal();
        car();
        car2();
        glFlush();
}
```

Update functions

Update function is used for the transformation of the objects this function is invoked in display function inside glutTimerFunc(50,update,0). This function calls the update every 50 mille seconds hence the variable 'a', 'b' are all modified then when we translate the object with this points we will see the animation effect.

```
void update(int value)
{
        a=a-6;
        b=b+6;
        control();
        glutPostRedisplay();
}
```

3.3 Various Features

- Easy to understand and use.
- No complicated Mouse or Keyboard events are used.

CHAPTER 4

SOURCE CODE

```
#include<stdio.h>
#include<GL/glut.h>
#include<string.h>
#include<windows.h>
void road();
void signal();
void car();
void car2();
void mydisplay();
void display();
void frontsreen();
void drawstring(float x,float y,float z,char *string);
void setFont();
void myMouse();
void update();
void helpscreen();
GLint a=300,b=-300,flag=0,traffic_regulator=1;
GLfloat red=0.196078,blue=0.8,green=0.6;
GLfloat p=0,q=0,r=1;
void *currentfont;

void setFont(void *font)
{
      currentfont=font;
}

void drawstring(float x,float y,float z,char *string)
{
      char *c;
      glRasterPos3f(x,y,z);
```

```
        for(c=string;*c!='\0';c++)
        {       glColor3f(0.0,0.0,0.0);
                glutBitmapCharacter(currentfont,*c);
        }
}

void frontscreen(void)
{
        setFont(GLUT_BITMAP_TIMES_ROMAN_24);
        glClearColor(0.0,0.0,0.0,0);/*background for cover page*/
        glClear(GL_COLOR_BUFFER_BIT);
        glColor3f(1,1,1);
        drawstring(450.0,700.0,0.0,"GOGTE INSTITUTE OF TECHNOLOGY");
        glColor3f(1,1,1);
        drawstring(330,650,0.0,"DEPARTMENT  OF  COMPUTER  SCIENCE  AND
ENGINEERING");
        glColor3f(1,1,1);
        drawstring(530,600,0.0,"A MINI PROJECT ON");
        glColor3f(1,1,1);
        drawstring(450,500,0.0,"SIMULATION TRAFFIC SIGNAL");
        glColor3f(1,1,1);
        drawstring(200,400,0.0,"BY:");
        glColor3f(1,1,1);
        drawstring(100,300,0.0,"SHUBHAM UROLAGIN        (2GI14CS148)");
        glColor3f(1,1,1);
        drawstring(100,240,0.0,"SHIVAPRASAD NADAGOUDR   (2GI14CS143)");
        glColor3f(1,1,1);
        drawstring(980,400,0.0,"GUIDES:");
        glColor3f(1,1,1);
        drawstring(930,300,0.0,"Prof. AJAY ACHARYA");
        glColor3f(1,1,1);
        drawstring(930,240,0.0,"Prof. ARATI.S.SHAHAPURKAR");
        glColor3f(1,1,1);
        drawstring(543,100,0.0,"PRESS ENTER TO START");
        glFlush();
```

```
}

void helpscreen()
{
      setFont(GLUT_BITMAP_TIMES_ROMAN_24);
      glClearColor(0,0,0,0);/*background for cover page*/
      glClear(GL_COLOR_BUFFER_BIT);
      glColor3f(1,1,1);
      drawstring(550.0,700.0,0.0,"TIPS");
      glColor3f(1,1,1);
      drawstring(650.0,700.0,0.0,"AND");
      glColor3f(1,1,1);
      drawstring(750.0,700.0,0.0,"TRICKS");
      glColor3f(1,1,1);
      drawstring(350.0,640.0,0.0,"Stop   the   traffic   (Red   Light)
MOUSE LEFT CLICK");
      glColor3f(1,1,1);
      drawstring(350.0,540.0,0.0,"Yellow                        Signal
MOUSE RIGHT BUTTON (HOLD ON)");
      glColor3f(1,1,1);
      drawstring(350.0,440.0,0.0,"Green                         Signal
MOUSE RIGHT BUTTON (RELEASE)");
      glColor3f(1,1,1);
      drawstring(350.0,340.0,0.0,"Speed      up      the      vehicles
PRESS 'S'");
      glColor3f(1,1,1);
      drawstring(350.0,90.0,0.0,"Help
PRESS 'H'");
      glColor3f(1,1,1);
      drawstring(350.0,40.0,0.0,"Escape
PRESS 'ENTER'");
      glFlush();
}

void myKeyboard( unsigned char key, int x, int y )
{
	switch(key)
	{
		case 13:
			if(flag==1)
			{
				flag=2;
				mydisplay();
			}
			if(flag==0) //Ascii of 'enter' key is 13
			{
				flag=1;
				mydisplay();
			}
		break;

		case 's':
			mydisplay();
		break;

		case 'h':
			flag=1;
			mydisplay();
		break;

		default:break;
	}
}

void myMouse(int button,int state,int x,int y)
{
	if(button==GLUT_LEFT_BUTTON && state==GLUT_DOWN)
	{
		traffic_regulator=0;
		p=1;
		q=0;
		r=0;
	}

	if(button==GLUT_RIGHT_BUTTON && state==GLUT_DOWN)
	{
		traffic_regulator=0;
		p=0;
		q=1;
		r=0;
	}

	if(button==GLUT_RIGHT_BUTTON && state==GLUT_UP)
	{
		traffic_regulator=1;
		p=0;
		q=0;
		r=1;
	}

	glutPostRedisplay();
}

void mydisplay(void)
{
	glClear(GL_COLOR_BUFFER_BIT);
	if(flag==0)
		frontscreen ();
	if(flag==1)
		helpscreen();
	if(flag==2)
		display();
	glutSwapBuffers();
}

void update(int value)
{
	a=a-6;
```

```
      b=b+6;

      glutPostRedisplay();
}

void display(void)
{
      if(traffic_regulator)
            glutTimerFunc(50,update,0);
      glClear(GL_COLOR_BUFFER_BIT);
      glClearColor(red,green,blue,0);/*back ground for sky*/
      road();
      signal();
      car();
      car2();
      glFlush();
}

void road()
{
      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();

//strips start
      glColor3f(1.0,1.0,1.0);
            glBegin(GL_POLYGON);
            glVertex2f(0,7.3);
            glVertex2f(6,7.3);
            glVertex2f(6,7.5);
```

```
			glVertex2f(0,7.5);
		glEnd();

		glColor3f(1.0,1.0,1.0);
			glBegin(GL_POLYGON);
			glVertex2f(7,7.3);
			glVertex2f(13,7.3);
			glVertex2f(13,7.5);
			glVertex2f(7,7.5);
		glEnd();

		glColor3f(1.0,1.0,1.0);
			glBegin(GL_POLYGON);
			glVertex2f(14,7.3);
			glVertex2f(20,7.3);
			glVertex2f(20,7.5);
			glVertex2f(14,7.5);
		glEnd();

		glColor3f(1.0,1.0,1.0);
			glBegin(GL_POLYGON);
			glVertex2f(21,7.3);
			glVertex2f(27,7.3);
			glVertex2f(27,7.5);
			glVertex2f(21,7.5);
		glEnd();
	//strips end

	//zebra crossing
		glColor3f(1.0,1.0,1.0);
			glBegin(GL_POLYGON);
			glVertex2f(28,7.1);
			glVertex2f(38,7.1);
			glVertex2f(38,7.9);
			glVertex2f(28,7.9);
		glEnd();
```

```
glColor3f(1.0,1.0,1.0);
      glBegin(GL_POLYGON);
      glVertex2f(28,6.1);
      glVertex2f(38,6.1);
      glVertex2f(38,6.9);
      glVertex2f(28,6.9);
glEnd();

glColor3f(1.0,1.0,1.0);
      glBegin(GL_POLYGON);
      glVertex2f(28,5.1);
      glVertex2f(38,5.1);
      glVertex2f(38,5.9);
      glVertex2f(28,5.9);
glEnd();

glColor3f(1.0,1.0,1.0);
      glBegin(GL_POLYGON);
      glVertex2f(28,8.1);
      glVertex2f(38,8.1);
      glVertex2f(38,8.9);
      glVertex2f(28,8.9);
glEnd();

glColor3f(1.0,1.0,1.0);
      glBegin(GL_POLYGON);
      glVertex2f(28,9.1);
      glVertex2f(38,9.1);
      glVertex2f(38,9.9);
      glVertex2f(28,9.9);
glEnd();

//green edge
glColor3f(0.1,0.2,0.1);
glBegin(GL_POLYGON);
```

```
			glVertex2f(0,5);
			glVertex2f(40,5);
			glVertex2f(40,4);
			glVertex2f(0,4);
		glEnd();

		glPopMatrix();
	}

	void signal()
	{
		glPushMatrix();
		glTranslated(20,120.0,0.0);
		glScaled(40.0,40.0,0.0);
		//stand
		glColor3f(0.1,0.2,0.1);
		glBegin(GL_POLYGON);
			glVertex2f(30,7);
			glVertex2f(30,8);
			glVertex2f(33,8);
			glVertex2f(33,7);
		glEnd();

		//pole
		glBegin(GL_POLYGON);
			glVertex2f(31,7);
			glVertex2f(32,8);
			glVertex2f(32,15);
			glVertex2f(31,15);
		glEnd();
		//board
		glBegin(GL_POLYGON);
			glVertex2f(30.5,15);
			glVertex2f(32.5,15);
			glVertex2f(32.5,10);
			glVertex2f(30.5,10);
		glEnd();
		//red
		glColor3f(p,0.0,0.0);
		glBegin(GL_POLYGON);
			glVertex2f(31,14.5);
			glVertex2f(32,14.5);
			glVertex2f(32,14);
			glVertex2f(31,14);
		glEnd();
		//yellow
		glColor3f(q,q,0.0);
		glBegin(GL_POLYGON);
			glVertex2f(31,13.5);
			glVertex2f(32,13.5);
			glVertex2f(32,13);
			glVertex2f(31,13);
		glEnd();
		//green
		glColor3f(0.0,r,0.0);
		glBegin(GL_POLYGON);
			glVertex2f(31,12.5);
			glVertex2f(32,12.5);
			glVertex2f(32,12);
			glVertex2f(31,12);
		glEnd();

		glPopMatrix();
	}

	void car()
	{
		glPushMatrix(); //making color for outer line
		glTranslated(b,290.0,0.0);
		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(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-500,190.0,0.0);
		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(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 myinit()
{
      glMatrixMode(GL_PROJECTION);
      glLoadIdentity();
      gluOrtho2D(0.0,1346.0,0.0,728.0);
}

int main(int argc, char** argv)
{
      glutInit(&argc, argv);
      glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
      glutInitWindowSize(1346,728);
      glutInitWindowPosition(0,0);
      glutCreateWindow("Traffic Signal");
      /*call back functions*/
      glutDisplayFunc(mydisplay);
      glutKeyboardFunc(myKeyboard);
      glutMouseFunc(myMouse);
      myinit();
      glutMainLoop();
}
```

CHAPTER 5

EXPERIMENTAL RESULTS AND ANALYSIS

The project designed has been tested for its working, and is found to be working properly to meet all its requirements.

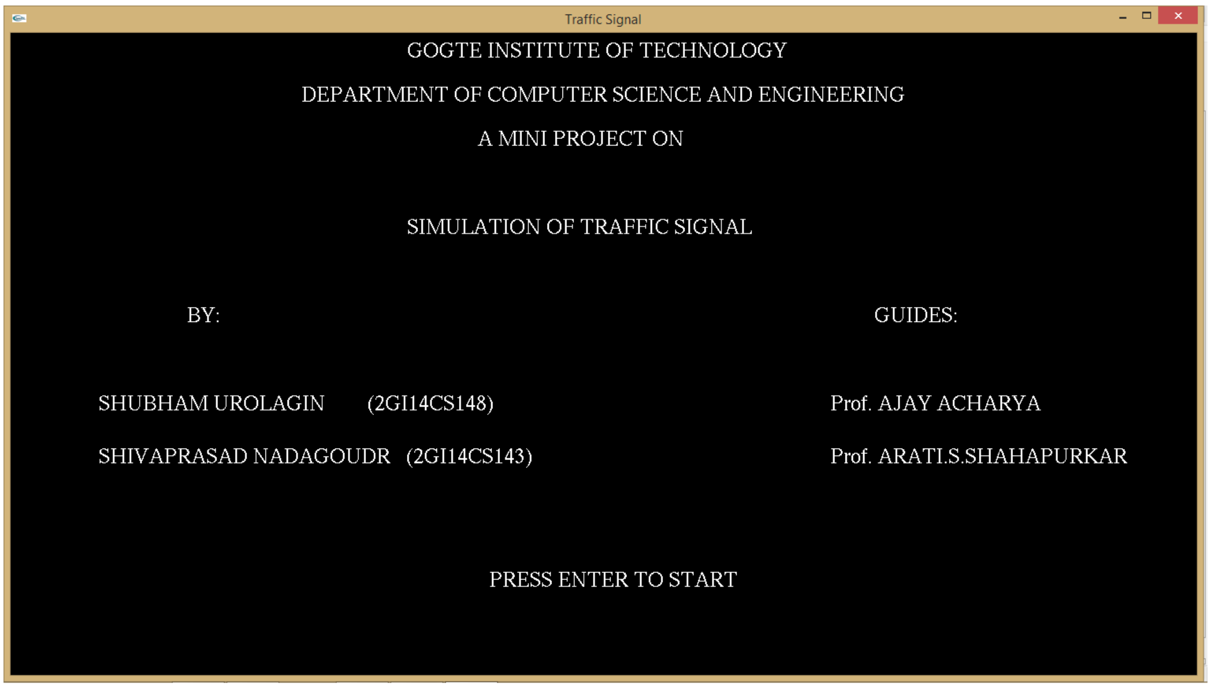
The project has been found to be giving correct outputs to the inputs that were given, like pressing of left mouse button will glows the Red light, on holding right mouse button Yellow light glows and releasing of same button will results in Green light.

The designed project has been tested for errors and has been found to be meeting all the requirements of design with which it was started. Thus the project is declared to be working properly.

5.1 Snapshots

Working of the project is explained through snapshots as follows.

1) Front screen showing project details



GOGTE INSTITUTE OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

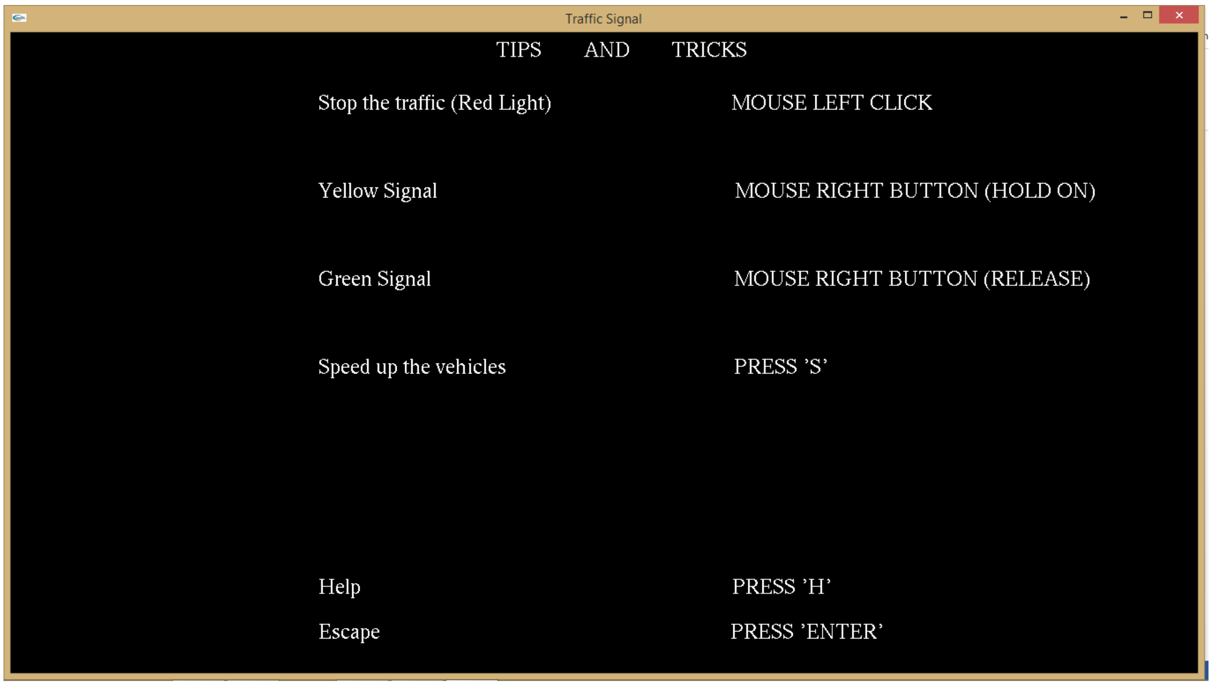
A MINI PROJECT ON

SIMULATION OF TRAFFIC SIGNAL

BY:		GUIDES:
SHUBHAM UROLAGIN	(2GI14CS148)	Prof. AJAY ACHARYA
SHIVAPRASAD NADAGOUDR	(2GI14CS143)	Prof. ARATI.S.SHAHAPURKAR

PRESS ENTER TO START

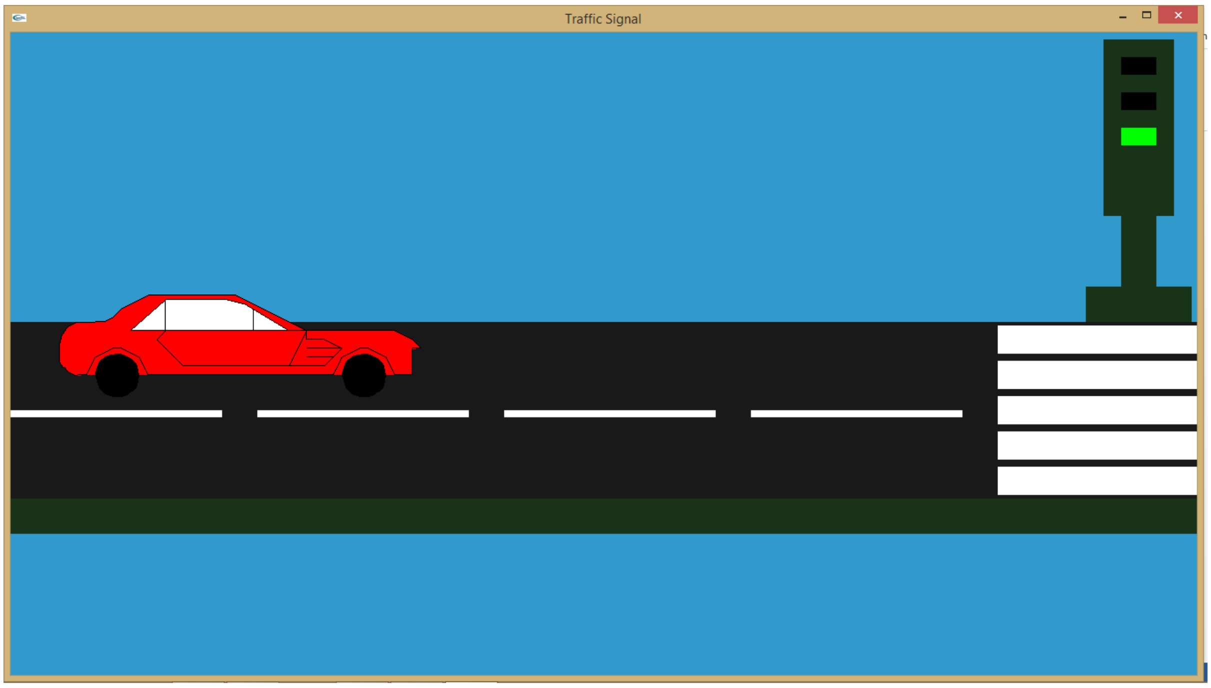
2) Screen showing user instructions



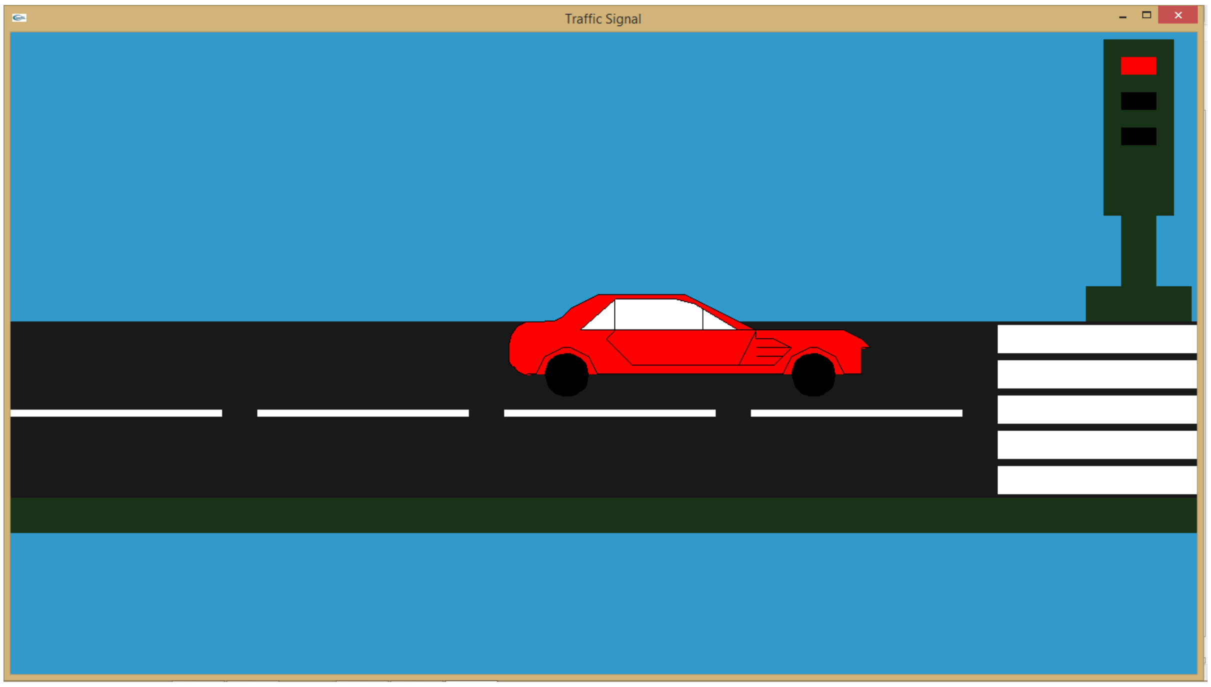
TIPS AND TRICKS

Stop the traffic (Red Light)	MOUSE LEFT CLICK
Yellow Signal	MOUSE RIGHT BUTTON (HOLD ON)
Green Signal	MOUSE RIGHT BUTTON (RELEASE)
Speed up the vehicles	PRESS 'S'
Help	PRESS 'H'
Escape	PRESS 'ENTER'

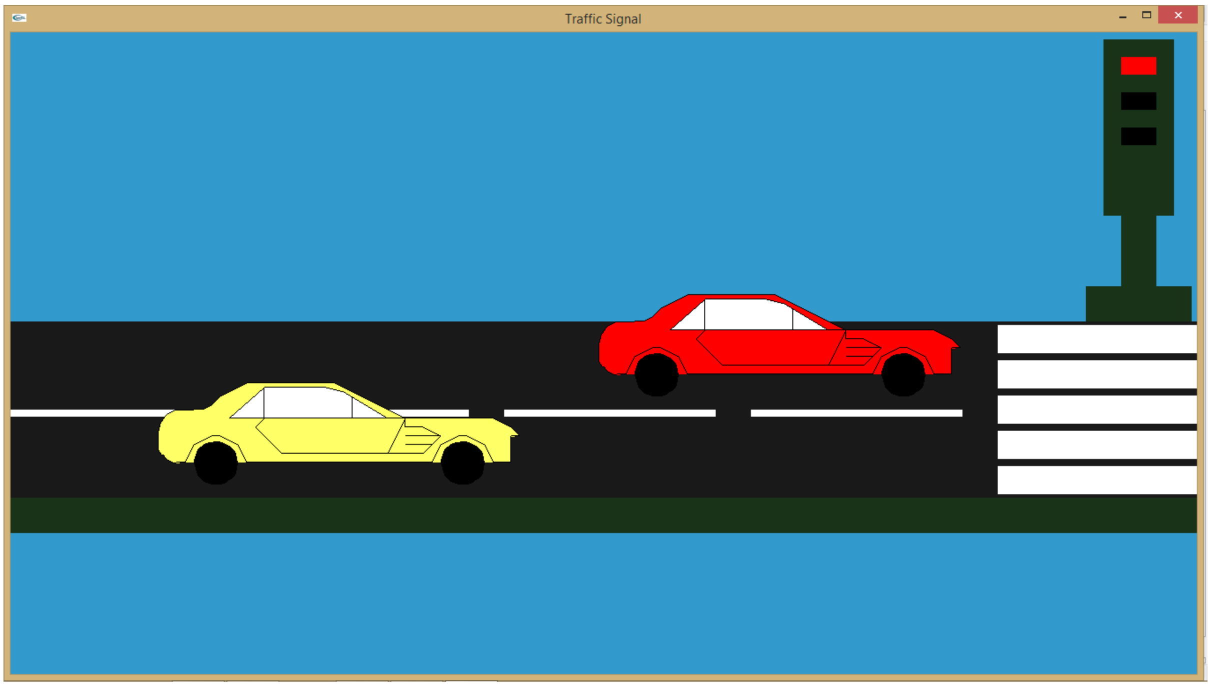
3) Car is moving on the road towards signal which is green.



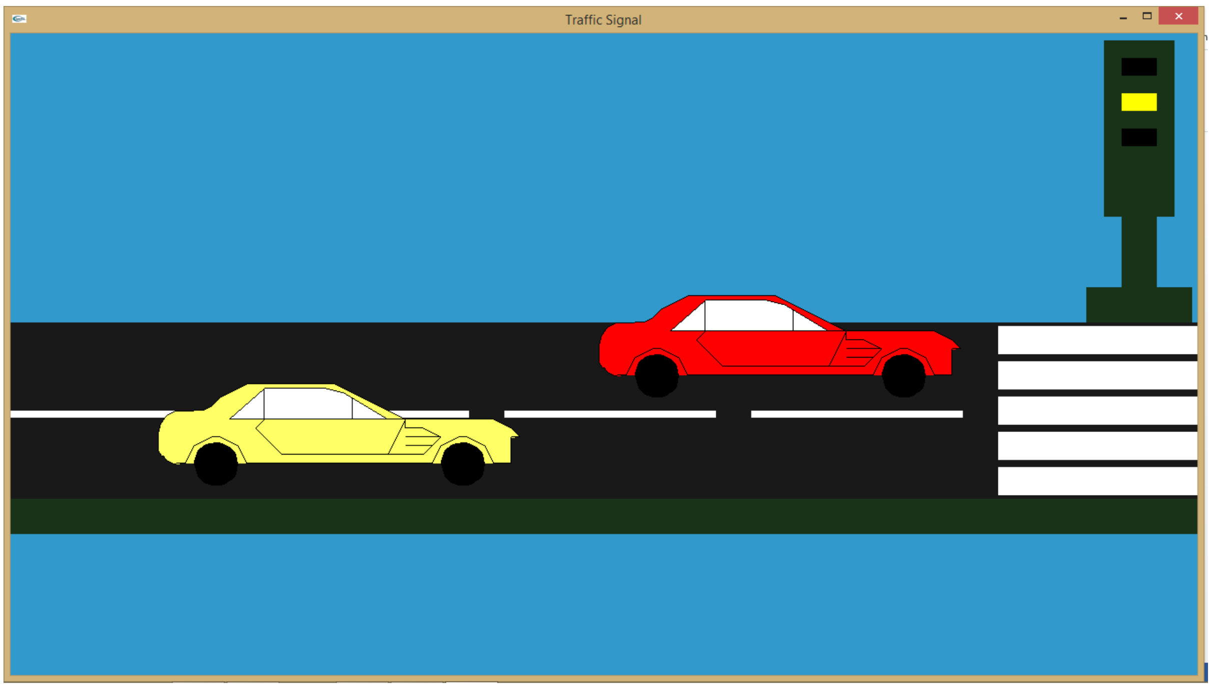
4) Since the signal becomes RED the car stops



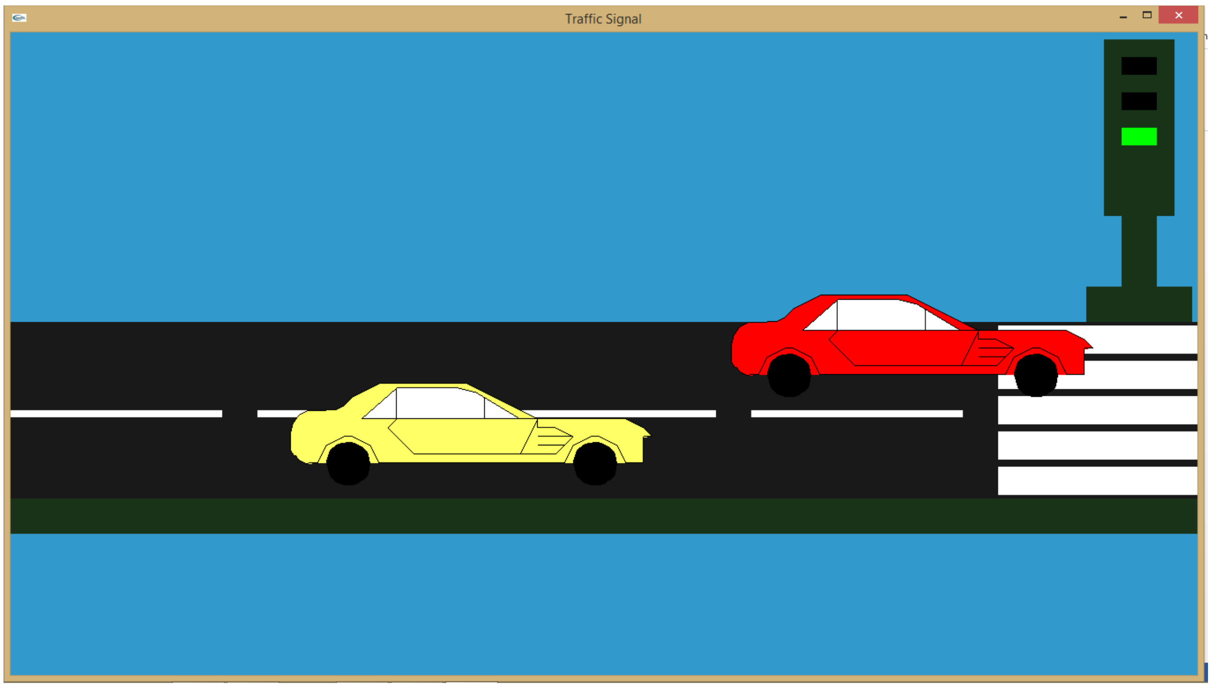
5) Traffic is stopped because of the RED signal



6) Signal is YELLOW and vehicles are ready to move.



7) Signal becomes GREEN and vehicles start moving.



CHAPTER 6

CONCLUSION

Graphics is a part of our day to day activities, thus computer graphics can be applied in various fields such as security systems, products animation studios to make animated films, cartoons; gaming industries by displaying use of high-end graphics in designing games; in engineering, architecture, medical fields by creating real-time models for better understanding, clarity and bringing out fresh, new ideas to enhance them.

Many improvements can be thought of to this project, such as we can add different vehicles, include more graphical images, automatic change of signal with respect to time and use of keyboard interface that provides more flexibility in processing the model as per user requirements. Care was taken to avoid bugs. We are looking forward to develop more such projects with an appetite in learning more in computer graphics.

REFERENCES

[1] Edward Angel's Interactive Computer Graphics, A top-down Approach, 5th Edition, Addison-Wesley, 2008.

[2] F.S. Hill's Computer Graphics Using OpenGL, Pearson Education, 2001.

[3] OpenGL Super Bible! By Richard S. Wright, Jr. and Michel Sweet.

[4] OpenGL Programming Guide (Addison-Wesley Publishing Company). The Official Guide to learning OpenGL, Version 1.1 -2nd edition.

[5] Donald Hearn and Pauline Baker, Computer Graphics – OpenGL Version, 2nd Edition, Pearson Education, 2003

[6] http://www.opengl.com