VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", Belagavi – 590 018

A Mini Project Report on

"Toll Collecting Booth"

Submitted in partial fulfillment of the requirement for the Computer Graphics Laboratory with mini project of VI Semester

Bachelor of Engineering in

Computer Science and Engineering

Submitted By

JOYCE DOLA [2JH20CS027]

MD GOUSE M KILLEDAR [2JH20CS042]

B PRERNA NAIDU [2JH20CS012]

SANJANA ASTOTI [2JH20CS078]

Under the Guidance of

Prof. Sahana B

Department of Computer Science and Engineering

JAIN COLLEGE OF ENGINEERING AND TECHNOLOGY, Hubballi

2022 – 2023

JAIN COLLEGE OF ENGINEERING AND TECHNOLOGY

Department of Computer Science and Engineering

CERTIFICATE

Certified that the VI Semester Mini Project in Computer Graphics Laboratory with mini project titled **"Toll Collecting Booth"** carried out by **Ms. Joyce Dola, Mr. Md Gouse M Killedar, Ms. Prerna Naidu, Ms. Sanjana Astoti.** Bearing **USN 2JH20CS027, 2JH20CS042, 2JH20CS012, 2JH20CS078 (resp)** a bonafide student of Jain College of Engineering and Technology, in partial fulfillment for the award of the **BACHELOR OF ENGINEERING** in Computer Science and Engineering from **Visvesvaraya Technological University, Belagavi** during the year 2022-2023. It is certified that all the corrections/suggestions indicated for Internal Assessment have been incorporated in the report submitted in the department library. The Computer Graphics Mini Project report has been approved as it satisfies the academic requirements in respect of the miniproject work prescribed for the said Degree.

____________________	____________________
Prof. Sahana B	Prof. Maheshkumar Patil
Asst. Professor	HOD
Dept. of CSE	Dept. of CSE

Name of the Examiners with signature

1. ____________________ ____________________
2. ____________________ ____________________

CONTENTS

INTRODUCTION

1.1 Introduction to Computer Graphics

Computer Graphics is concerned with all aspects of producing pictures or images using a computer. We can create images that are indistinguishable from photographs of real objects. In other terms, Computer Graphics are the graphics created by the computers, and more generally, the representation and manipulation of image data by a computer. The development of computer graphics has been driven both by the needs of the user community and by advances in hardware and software.

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 images so produced, and manipulating visual content.

1.2 History of Computer Graphics

The phrase Computer Graphics was coined in 1960 by William Fetter, a graphic designer for Boeing. The field of Computer Graphics developed with the emergence of computer graphics hardware. Early projects like the Whirlwind and SAGE projects introduced the CRT as a viable display and interaction interface and introduced the light pen as an input device. Further advances in computing led to greater advancements in interactive computer graphics. In 1959, the TX-2 computer was developed at MIT Lincoln Laboratory. A light pen could be used to draw sketches on the computer using Ivan Sutherlands revolutionary Sketchpad software.

Also in 1961 another student at MIT, Steve Russell, created the first video game, Space war. E. E. Zajac, a scientist at Bell Telephone Laboratory (BTL), created a film called “Simulation of a two-giro gravity attitude control system” in 1963. In this computer generated film, Zajac showed how the attitude of a satellite could be altered as it orbits the Earth. Many of the most important early breakthroughs in computer graphics research occurred at the University of Utah in the 1970s.

The first major advance in 3D computer graphics was created at UU by these early pioneers, the hidden-surface algorithm. In order to draw a representation of a 3D object on the screen, the computer must determine which surfaces are "behind" the object from the viewer's perspective, and thus should be "hidden" when the computer creates (or renders) the image.

Graphics and application processing were increasingly migrated to the intelligence in the workstation, rather than continuing to rely on central mainframe and mini-computers. 3D graphics became more popular in the 1990s in gaming, multimedia and animation. Computer graphics used in films and video games gradually began to be realistic to the point of entering the uncanny valley. Examples include the later Final Fantasy games and animated films like The Polar Express.

1.3 Applications of Computer Graphics

The development of computer graphics has been driven both by the needs of the user community and by advances in hardware and software. The applications of computer graphics are many and varied. We can however divide them into four major areas.

- Display of information: More than 4000 years ago, the Babylonians developed floor plans of buildings on stones. Today, the same type of information is generated by architects using computers. Over the past 150 years, workers in the field of statistics have explored techniques for generating plots. Now, we have computer plotting packages. Supercomputers now allow researchers in many areas to solve previously intractable problems. Thus, Computer Graphics has innumerable applications.
- Design: Professions such as engineering and architecture are concerned with design. Today, the use of interactive graphical tools in CAD, in VLSI circuits, characters for animation have developed in a great way.
- Simulation and animation: One of the most important uses has been in pilots‟ training. Graphical flight simulators have proved to increase safety and reduce expenses. Simulators can be used for designing robots, plan it's path, etc. Video games and animated movies can now be made with low expenses.

- User interfaces: Our interaction with computers has become dominated by avisual paradigm. The users‟ access to internet is through graphical network browsers. Thus Computer Graphics plays a major role in all fields.

1.4 Introduction to OpenGL

OpenGL is a software interface to graphics hardware. This interface consists of about 150 distinct commands that are used 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.

These are certain characteristics of OpenGL:

- ✓ OpenGL is a better documented API.
- ✓ OpenGL is much easier to learn and program.
- ✓ OpenGL has the best demonstrated 3D performance for any API.

The OpenGL specification describes an abstract API for drawing 2D and 3D graphics. Although it's possible for the API to be implemented entirely in software, it's designed to be implemented mostly or entirely in hardware.

In addition to being language-independent, OpenGL is also platform-independent. 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.

OpenGL is an evolving API. New versions of the OpenGL specification are regularly released by the Khronos Group, each of which extends the API to support various new features. In addition to the features required by the core API, 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. Vendors can use extensions to expose custom APIs without needing support from other vendors or the Khronos Group as a whole, which greatly increases the flexibility of OpenGL. All extensions are collected in, and defined by, the OpenGL Registry.

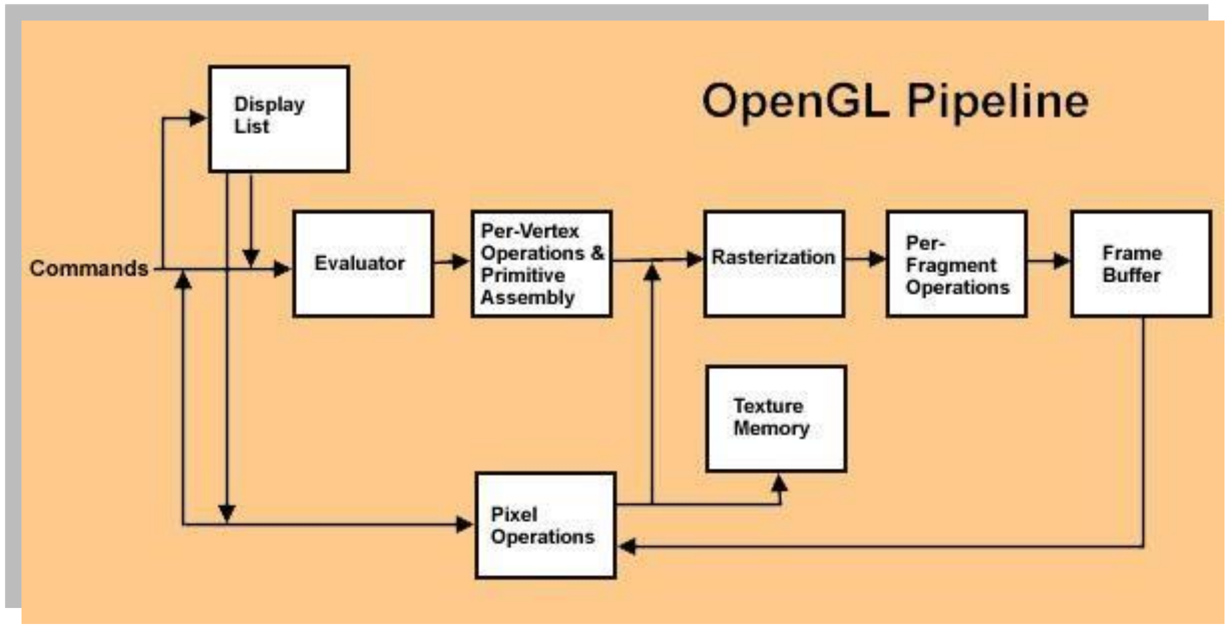


Figure 1.1 OpenGL Pipeline

1.5 Introduction to GLUT

GLUT is the OpenGL utility toolkit, a window system independent toolkit for writing OpenGL programs. It implements a simple windowing API for OpenGL. GLUT makes it 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 a full-featured toolkit so large applications requiringsophisticated user interfaces are better off using native window system toolkits. The GLUT library has both C, C++ (same as C), FORTRAN, and ADA programming bindings. The GLUT source code distribution is portable to nearly all OpenGL implementations and platforms.

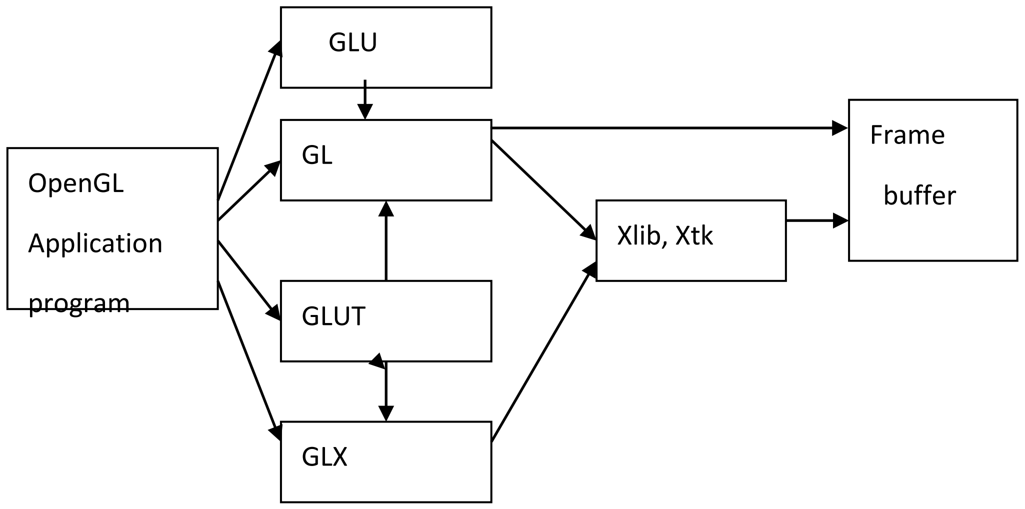


Figure 1.2 library organization of OpenGL

1.6 Applications of OpenGL

- ✓ **OpenGL** (**Open G**raphics **L**ibrary) is a cross-language, multi-platform API for rendering 2D and 3D computer graphics.
- ✓ The API is typically used to interact with a GPU, to achieve hardware accelerated rendering.
- ✓ It is widely used in CAD, virtual reality, scientific visualization, information visualization, flight simulation, and video games.

1.7 OpenGL primitives

OpenGL supports two classes of primitives:

- Geometric Primitives
- Image (Raster) Primitives

Geometric primitives are specified in the problem domain and include points, line segments, polygons, curves and surfaces.

Raster primitives, such as arrays of pixels pass through a separate parallel pipeline on their way to the frame buffer.

There are ten basic OpenGL primitives:

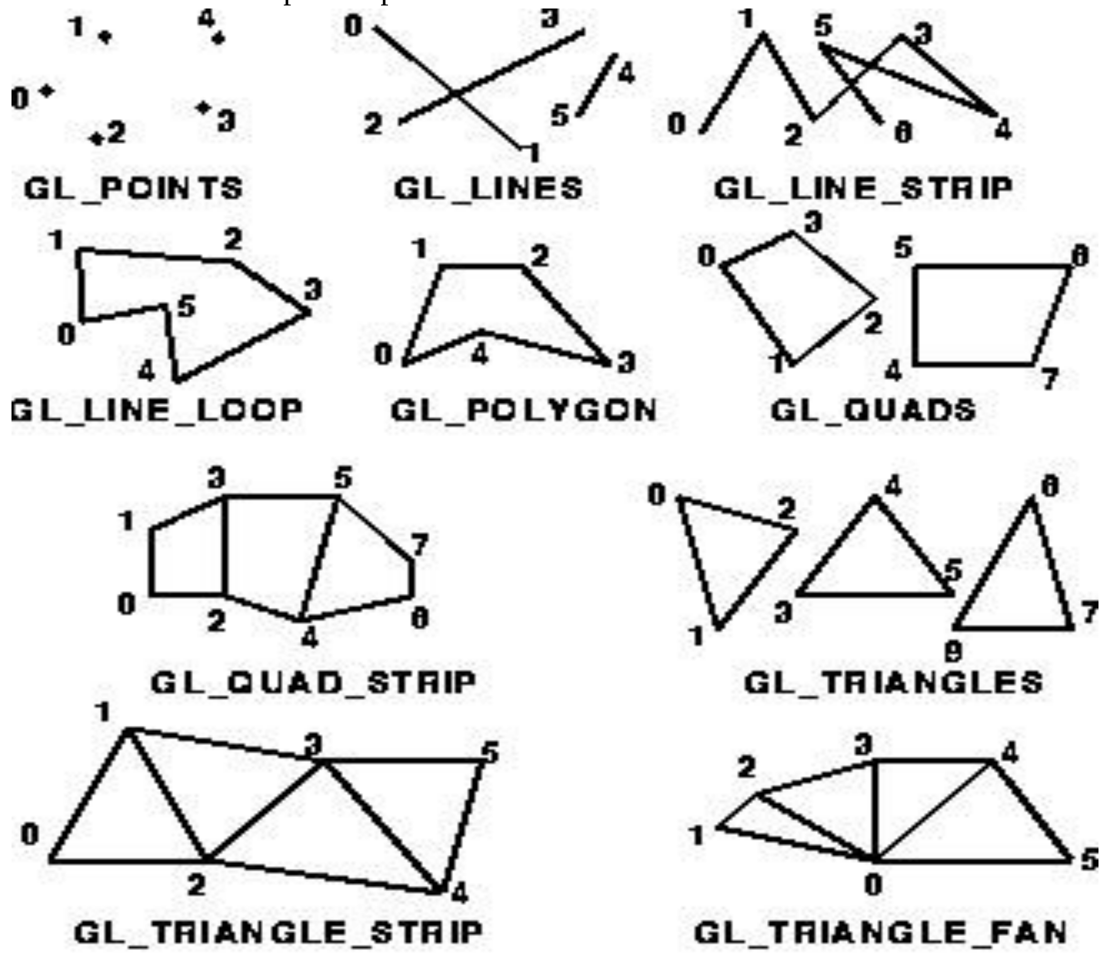


Figure 1.3 OpenGL Primitives

2. LITRETURE SURVEY

The basic functions like glColor3f(…); glRotatef(..), glTranslatef(..) etc that are most commonly used in the code are taken from the prescribed VTU Text book "INTERACTIVE COMPUTER GRAPHICS" 5th edition by Edward Angel.[1]. The lab programs in the syllabus also serve as a basic template for creating a project. The usage of colors and specifications are taken from the various programs that were taught in the lab.[1]. The VTU prescribed text book serves as a huge database of functions and they are used in the project.

3.SYSTEM REQUIREMENTS

The graphics editor has been programmed in C. It makes use of Turbo C Graphics library package for creating the user interface. This is a subroutine library for terminal independent screen painting and input event handling.

3.1 Hardware Requirements:

The standard output device is assumed to be a Color Monitor. It is quite essential for any graphics package to have this, as provision of color options to the user is a must. The mouse, the main input device, has to be functional i.e., used to give input in the game. A keyboard is used for controlling and inputting data in the form of characters, numbers i.e., to change the user views. Minimum Requirements expected are cursor movement, creating objects like lines, squares, rectangles, polygons, etc. Transformations on objects/selected area should be possible. Filling of area with the specified color should be possible.

- Processor: Intel dual core i5.
- RAM: 1GB RAM or above.
- Input devices: Keyboard.
- Output devices: Monitor.

3.2Software Requirements:

The editor has been implemented on the OpenGL platform and mainly requires the appropriate version of Microsoft Visual Studio.

- Operating System: Microsoft Windows 10.
- Microsoft Visual Studio 2010
- glut.h header file
- glut.dll library file

4.ABOUT THE PROJECT

4.1 PROPOSED SYSTEM:

To achieve three dimensional effects, OpenGL software is proposed. It is software which provides a graphical interface. It is a interface between application program and graphics hardware. The advantages are:

1. OpenGL is designed as a streamlined.
2. It is 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.
6. It provides call back function.

Main theme of the project is:

The main idea behind this project is to display the concept of toll collecting booth with computer graphics. This graphics package is based on the OpenGL library functions. The programming language used here is C using OpenGL libraries. The idea behind this project is to display the concept of toll collection booths through computer graphics. This project will include a toll collecting booth where as soon as the car comes the barrier is imposed. This barrier will then be lifted by the user. There will be two way path in at both way we have at toll collecting booth for both direction. To open the left side barrier we use to press “L|l” key, as soon as the key is pressed the left side car will move. And in the same way, we have “R|r” key to open the barrier and allow the car to move further. Here we also we can increase or decrease the speed of the cars by using proper keys.

Module Description

MAIN FUNCTION: Main is a function from which execution of the program ill start. It initializes the window of size 1000*1000 pixels. And also call various functions defined in the program, also includes a menu which intern calls the related functions (such as my menu, submenu, etc…).

MYINIT FUNCTION: This function is used to convert one coordinate system to another coordinate system. We used the GL_MatrixMode(GL_PROJECTION) and glLoadIdentity() in this function.

MENU FUNCTION: It creates the main and the sub-menus, whereas the main menu is to "Quit, increase the size and decrease size" of the objects. And sub-menus are used for color. The menu is assigned to mouse click when the mouse (LEFT or RIGHT) button is pressed the menu will appear and action is performed.

MOUSE FUNCTION: This is an event-driven function which is used to select the objects. Here in this project, we have used this mouse function to select the paint objects, specified in a specific region.

GLSL Built-in Variables

There are two very important built-in variables in GLSL. Without their use in shaders, no rendering will occur. They are:

- gl_Position;
- gl_FragColor;

gl_Position

The gl_Position is used by the vertex shader to hand over the
transformed vertex position to the OpenGL pipeline.

gl_FragColor

The gl_FragColor is used by the fragment shader to hand over the color of the fragment to the OpenGL pipeline.

The gl_Position and gl_FragColor variables are mandatory in the vertex and fragment shaders, respectively.

5.IMPLEMENTATION

5.1 FUNCTIONS USED

Void glColor3f(float red, float green, float blue):

This function is used to mention the color in which the pixel should appear. The number 3 specifies the number of arguments that the function would take. “f “ gives the type that is float. The arguments are in the order RGB (Red, Green, Blue). The color of the pixel can be specified as the combination of these 3 primary colors.

Void glClearColor(int red, int green, int blue, int alpha):

This function is used to clear the color of the screen. The 4 values that are passed as arguments for this function are (RED, GREEN, BLUE, ALPHA) where the red green and blue components are taken to set the background color and alpha is a value that specifies depth of the window. It is used for 3D images.

Void glutKeyboardFunc():

void glutKeyboardFunc(void (*func)(unsigned char key, int x, int y)); where func is the new keyboard callback function. glutKeyboardFunc sets the keyboard callback for the current window. When a user types into the window, each key press generating an ASCII character will generate a keyboard callback. The key callback parameter is the generated ASCII character. The x and y callback parameters indicate the mouse location in window relative coordinates when the key was pressed. When a new window is created, no keyboard callback is initially registered, and ASCII key strokes in the window are ignored. Passing NULL to glutKeyboardFunc disables the generation of keyboard callbacks.

Void glFlush():

Different GL implementations buffer commands in several different locations, including network buffers and the graphics accelerator itself. glFlush empties all of these buffers, causing all issued commands to be executed as quickly as they are accepted by the actual rendering engine.

Void glMatrixMode(GLenum mode):

where mode Specifies which matrix stack is the target for subsequent matrix operations. Three values are accepted: **GL_MODELVIEW, GL_PROJECTION, and GL_TEXTURE.** The initial value is **GL_MODELVIEW**. **glMatrixMode** sets the current matrix mode. Mode can assume one of three values:

GL_MODELVIEW:	Applies subsequent matrix operations to the model view matrix stack.
GL_PROJECTION:	Applies subsequent matrix operations to the projection matrix stack.

void glutInit(int *argc, char **argv):

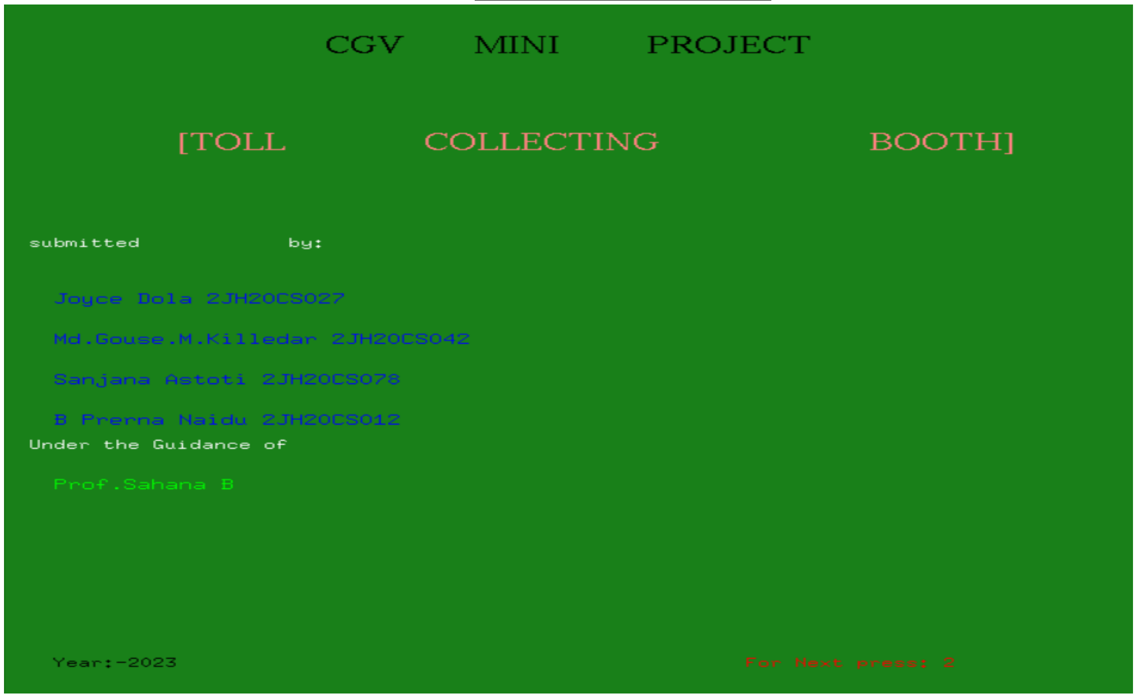
glutInit will initialize the GLUT library and negotiate a session with the window system. During this process, glutInit may cause the termination of the GLUT program with an error message to the user if GLUT cannot be properly initialized. Examples of this situation include the failure to connect to the window system, the lack of window system support for OpenGL, and invalid command line options. glutInit also processes command line options, but the specific options parse are window system dependent. **void glutReshapeFunc(void (*func)(int width, int height)):**

glutReshapeFunc sets the reshape callback for the current window. The reshape callback is triggered when a window is reshaped. A reshape callback is also triggered immediately before a window's first display callback after a window is created or whenever an overlay for the window is established. The width and height parameters of the callback specify the new window size in pixels. Before the callback, the current window is set to the window that has been reshaped.

void glutMainLoop(void):

glutMainLoop enters the GLUT event processing loop. This routine should be called at most once in a GLUT program. Once called, this routine will never **glutPostRedisplay():** glutPostRedisplay, glutPostWindowRedisplay — marks the current or specified window as needing to be redisplayed.

6.SNAP SHOTS



CGV MINI PROJECT

[TOLL COLLECTING BOOTH]

submitted by:

Joyce Dola 2JH20CS027

Md.Gouse.M.Killedar 2JH20CS042

Sanjana Astoti 2JH20CS078

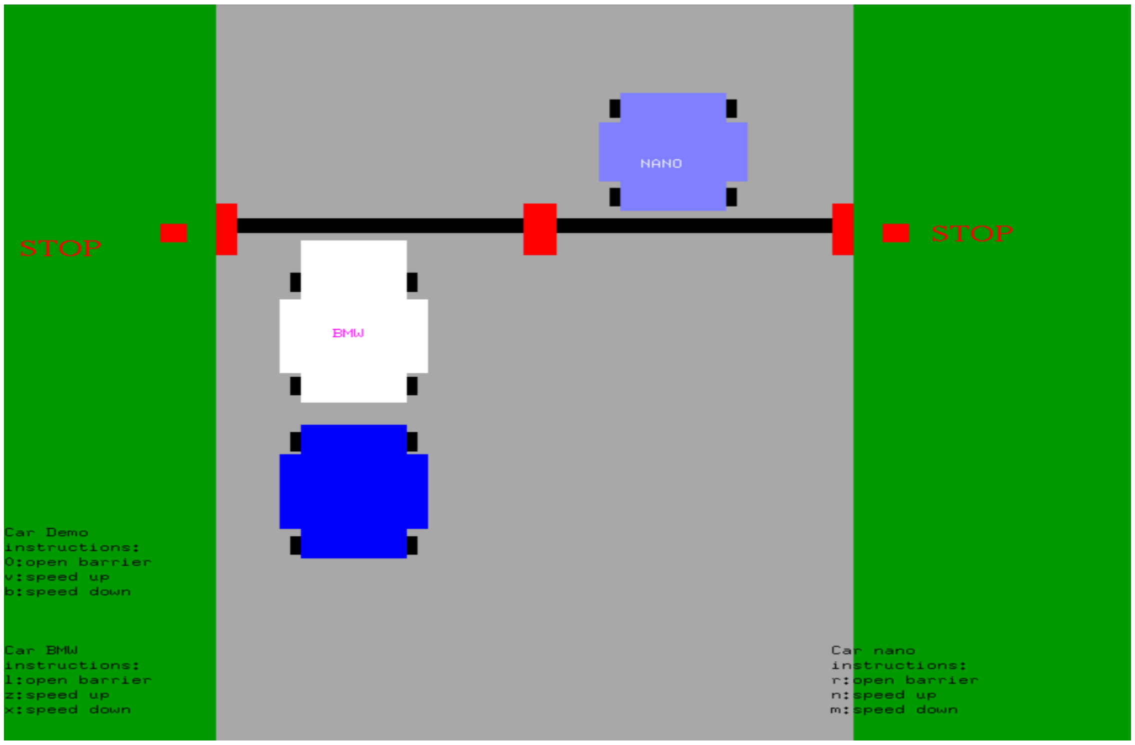
B Prerna Naidu 2JH20CS012

Under the Guidance of

Prof.Sahana B

Year:-2023

For Next press: 2

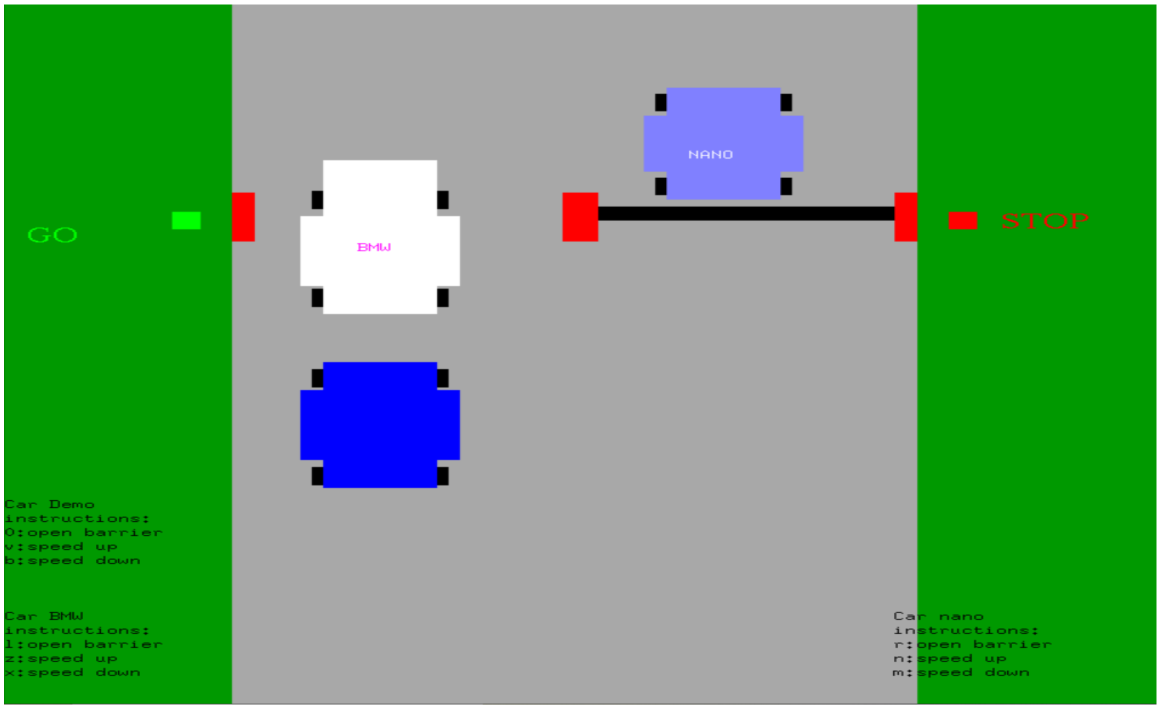


```
Car Demo
instructions:
O:open barrier
v:speed up
b:speed down

Car BMW
instructions:
l:open barrier
z:speed up
x:speed down

Car nano
instructions:
r:open barrier
n:speed up
m:speed down
```

BMW car and Nano car are paying toll.

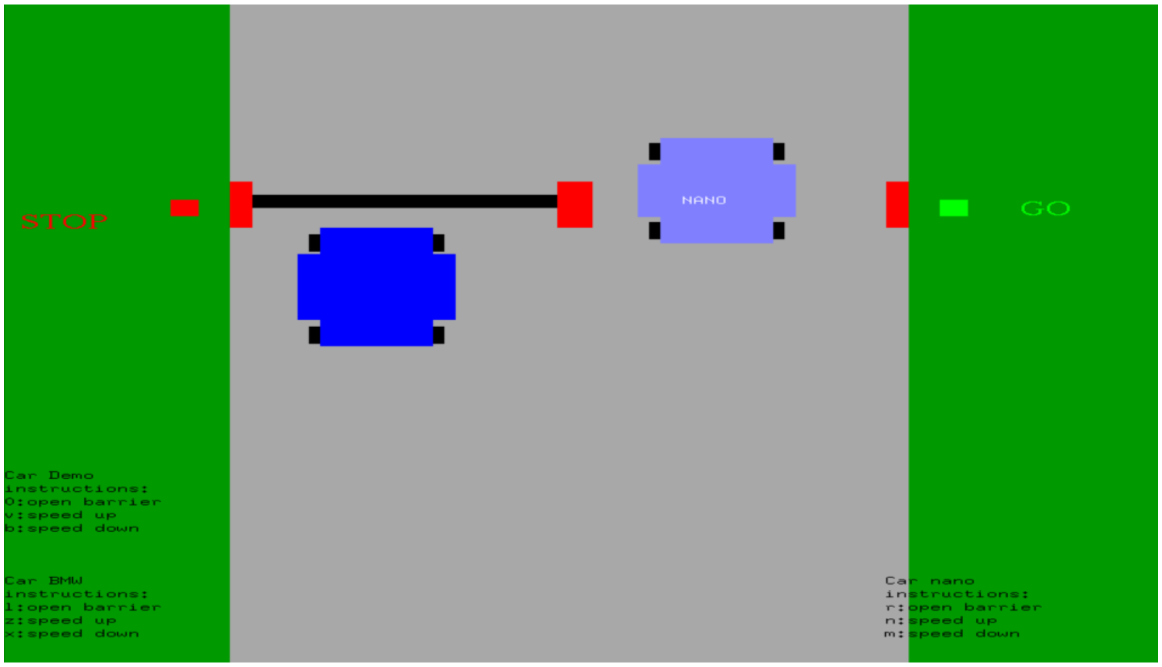


```
Car Demo
instructions:
0:open barrier
v:speed up
b:speed down

Car BMW
instructions:
1:open barrier
z:speed up
x:speed down

Car nano
instructions:
r:open barrier
n:speed up
m:speed down
```

After paying toll car BMW went and the next car is waiting to pay the toll



```
Car Demo
instructions:
0:open barrier
v:speed up
b:speed down

Car BMW
instructions:
1:open barrier
z:speed up
x:speed down

Car nano
instructions:
r:open barrier
n:speed up
m:speed down
```

As soon as we pay the toll the barriers will open and car will move, from both sides.

7.CONCLUSION AND FUTURE SCOPE

Electronic toll collection system allows the vehicle drives to pass the toll tax booths without stopping at the toll booth. The toll amount is deduced from the RFID card. This RFID cards is rechargeable and account is stopped on the records. Automatic Toll Tax systems have really helped a lot in reducing the heavy congestion caused in the metropolitan cities of today. It is one of the easiest methods used to organize the heavy flow of traffic. When the car moves through the toll gate on any road, it is indicated on the RFID reader that it has crossed the clearing. The need for manual toll based systems is completely reduced in this methods and the tolling system works through RFID. The system thus installed is quite expedient reducing the time and cost of travellers since the tag can be deciphered from a distance. As we all know that transportation is the backbone of any country's economy. Improvement in transportation systems result into the good lifestyle in which we achieve extraordinary freedom for movement, immense trade in manufactured goods and services, as well as higher rate of employment levels and social ability. In fact, the economic condition of a nation has been closely related to efficient ways of transportation. Increasing number of vehicles on the road, result into number of problems such as congestion, accident rate, air pollution and many others One of the important aspects of modern electronic technology is embedded systems based on micro controllers. The main aim of science has always been to make our lives easier. In this project, we discuss an innovative concept of tool booth. Some of the issues faced by customer are: Wastage of valuable time, Traffic jam and money loss, fuel loss. The necessity for vehicles to stop or slow down for toll fee payment results in traffic congestion and reduces fuel efficiency. Hence, a system that enables road users to pay the toll fees without stopping or slowing down was proposed and developed. Hardware and software designs were carried out to develop a RFID based highway toll collection system. This system has developed using a microcontroller. Different modules such as GSM module, Liquid Crystal Display (LCD) module.

8.FUTURE ENHANCEMENTS

This project has been designed such that it works on the windows platform. The project can be designed using different languages and better graphical interfaces. The following features can be incorporated.

9.Code

```
void welcome()
{
glClear(GL_COLOR_BUFFER_BIT);
glColor3f(0.0,0.0,0.0);
drawstring(150,400,0.0,"WEL-COME");
glColor3f(1.0,1.0,0.0);
drawstring1(100,300,0.0,"Mini Project created by:");
glColor3f(0.0,0.0,1.0);
drawstring(180,260,0.0,"GOUSE,JOYCE,PRERNA,
SANJANA");
glColor3f(1.0,1.0,0.0); drawstring2(300,20,0.0,"For
Next Press:1");
}
 void frontscreen()
{
glClear(GL_COLOR_BUFFER_BIT);
glColor3f(0.0,0.0,0.0); drawstring(130,470,0.0,"CG");
drawstring(190,470,0.0,"Mini");
drawstring(260,470,0.0,"Project");
glColor3f(1.0,0.5,0.5);
drawstring(70,410,0.0,"[TOLL");
drawstring(170,410,0.0,"COLLECTING");
drawstring(350,410,0.0,"BOOTH]");
glColor3f(1.0,1.0,1.0);
drawstring1(10,350,0.0,"Submitted");
drawstring1(115,350,0.0,"by:");
glColor3f(0.0,0.0,0.0);
```

```
drawstring1(20,320,0.0,"Students Name
USN"); glColor3f(0.0,0.0,1.0);
drawstring2(20,265,0.0,"Sanjana Astoti
2JH20CS078");
 drawstring2(20,240,0.0,"B Prerna Naidu
2JH20CS012");
drawstring2(20,290,0.0,"Md.Gouse.M.Killedar
2JH20CS042");
drawstring2(20,315,0.0,"Joyce Dola 2JH20CS027");
glColor3f(1.0,1.0,1.0);
drawstring1(10,190,0.0,"Under the Guidance of");
glColor3f(0.0,1.0,0.0);
drawstring2(20,160,0.0,"Prof.SAHANA B");
glColor3f(0.0,0.0,0.0);
drawstring1(20,90,0.0,"Year:2021 ");
glColor3f(1.0,0.0,0.0); drawstring1(300,90,0.0,"For
Next press: 2");
 glFlush();
}
 void color() { glColor3f(0.0,0.0,1.0); if(color1==1) glColor3f(0.7,0.2,0.5);
}
 void display()
{
glClear(GL_COLOR_BUFFER_BIT);
if(flaga==0)
{ welcome();
} else {
if(flagd==0)
frontscreen()
```

```
; else {
glClearColor
(0.0,0.6,0.0,
1.0);
glColor3f(0.
658824,0.65
8824,0.6588
24);
glBegin(GL
_POLYGON
);
glVertex2f(1
00,500);
glVertex2f(4
00,500);
glVertex2f(4
00,0);
glVertex2f(1
00,0);
glEnd();
glColor3f(1.
0,0.0,0.0);
glBegin(GL_POLYGON);
glVertex2f(100,365);
glVertex2f(110,365);
glVertex2f(110,330);
glVertex2f(100,330); glEnd();
glColor3f(1.0,0.0,0.0);
```

```
glBegin(GL_POLYGON);
glVertex2f(390,365);
glVertex2f(400,365);
glVertex2f(400,330);
glVertex2f(390,330); glEnd();
glColor3f(1.0,0.0,0.0);
glBegin(GL_POLYGON);
glVertex2f(245,365);
glVertex2f(260,365);
glVertex2f(260,330);
glVertex2f(245,330); glEnd();
glColor3f(0.5,0.5,1.0);
glBegin(GL_QUADS);
glVertex2f(bx+290,by1);
glVertex2f(bx+340,by2);
glVertex2f(bx+340,by3);
glVertex2f(bx+290,by4);
glEnd();
glBegin(GL_QUADS);
glVertex2f(bx+280,by5);
glVertex2f(bx+350,by6);
glVertex2f(bx+350,by7);
glVertex2f(bx+280,by8); glEnd();
glBegin(GL_QUADS);
glVertex2f(bx+290,by9);
glVertex2f(bx+340,by10);
glVertex2f(bx+340,by11);
glVertex2f(bx+290,by12); glEnd();
glBegin(GL_QUADS);
```

```
glColor3f(0.0,0.0,0.0); glVertex2f(bx+285,by13);//rt
cr tp lft wl glVertex2f(bx+290,by14);
glVertex2f(bx+290,by15);
glVertex2f(bx+285,by16); glEnd();
glBegin(GL_QUADS);
glColor3f(0.0,0.0,0.0);
glVertex2f(bx+340,by17);//rt cr tp rt wl
glVertex2f(bx+345,by18);
glVertex2f(bx+345,by19);
glVertex2f(bx+340,by20); glEnd();
glBegin(GL_QUADS);
glColor3f(0.0,0.0,0.0);
glVertex2f(bx+285,by21);//rt cr btm lft wel
glVertex2f(bx+290,by22);
glVertex2f(bx+290,by23);
glVertex2f(bx+285,by24); glEnd();
glBegin(GL_QUADS);
glColor3f(0.0,0.0,0.0);
glVertex2f(bx+340,by25);//rt cr btm rt wel
glVertex2f(bx+345,by26);
glVertex2f(bx+345,by27);
glVertex2f(bx+340,by28); glEnd();
//car name

glColor3f(1.0,1.0,1.0); drawstring1(bx+300,zy2,0,"Nano");
//left side car glColor3f(1.0,1.0,1.0);
glBegin(GL_QUADS);
glVertex2f(ax+140,ay1);
glVertex2f(ax+190,ay2);
```

```
glVertex2f(ax+190,ay3);
glVertex2f(ax+140,ay4); glEnd();
glBegin(GL_QUADS);
glVertex2f(ax+130,ay5);
glVertex2f(ax+200,ay6);
glVertex2f(ax+200,ay7);
glVertex2f(ax+130,ay8); glEnd();
glBegin(GL_QUADS);
glVertex2f(ax+140,ay9);
glVertex2f(ax+190,ay10);
glVertex2f(ax+190,ay11);
glVertex2f(ax+140,ay12); glEnd();
//left car name
glColor3f(1.0,0.0,1.0);
drawstring1(ax+155,zy1,0,"BMW");
//wheel
glBegin(GL_QUADS);
glColor3f(0.0,0.0,0.0);
glVertex2f(ax+135,ay13);//lft cr btm
glVertex2f(ax+140,ay14);//lt wel
glVertex2f(ax+140,ay15);
glVertex2f(ax+135,ay16); glEnd();
glBegin(GL_QUADS); glColor3f(0.0,0.0,0.0); glVertex2f(ax+190,ay17);//lft cr btm rt wel
glVertex2f(ax+195,ay18); glVertex2f(ax+195,ay19); glVertex2f(ax+190,ay20); glEnd();

glBegin(GL_QUADS);
glColor3f(0.0,0.0,0.0);
glVertex2f(ax+135,ay21);//left cr top
```

```
glVertex2f(ax+140,ay22);//lft wel
glVertex2f(ax+140,ay23);
glVertex2f(ax+135,ay24); glEnd();
glBegin(GL_QUADS);
glColor3f(0.0,0.0,0.0);
glVertex2f(ax+190,ay25);//lft car top rt wel
glVertex2f(ax+195,ay26);
glVertex2f(ax+195,ay27);
glVertex2f(ax+190,ay28); glEnd(); // car3
//color(); color();
glBegin(GL_QUADS); glVertex2f(ax+140,cy1);
glVertex2f(ax+190,cy2);
glVertex2f(ax+190,cy3);
glVertex2f(ax+140,cy4); glEnd();
glBegin(GL_QUADS);
glVertex2f(ax+130,cy5);
glVertex2f(ax+200,cy6);
glVertex2f(ax+200,cy7);
glVertex2f(ax+130,cy8); glEnd();
glBegin(GL_QUADS);
glVertex2f(ax+140,cy9);
glVertex2f(ax+190,cy10);
glVertex2f(ax+190,cy11);
glVertex2f(ax+140,cy12); glEnd();
//wheel
glBegin(GL_QUADS);
glColor3f(0.0,0.0,0.0);
glVertex2f(ax+135,cy13);//lft cr btm
```

```
glVertex2f(ax+140,cy14);//lt wel
glVertex2f(ax+140,cy15);
glVertex2f(ax+135,cy16); glEnd();
glBegin(GL_QUADS);
glColor3f(0.0,0.0,0.0); glVertex2f(ax+190,cy17);//lft
cr btm rt wel glVertex2f(ax+195,cy18);
glVertex2f(ax+195,cy19); glVertex2f(ax+190,cy20);
glEnd();
glBegin(GL_QUADS);
glColor3f(0.0,0.0,0.0);
glVertex2f(ax+135,cy21);//left cr top
glVertex2f(ax+140,cy22);//lft wel
glVertex2f(ax+140,cy23);
glVertex2f(ax+135,cy24); glEnd();
glBegin(GL_QUADS);
glColor3f(0.0,0.0,0.0);
glVertex2f(ax+190,cy25);//lft car top rt wel
glVertex2f(ax+195,cy26);
glVertex2f(ax+195,cy27);
glVertex2f(ax+190,cy28); glEnd();

// display instruction for left car
glColor3f(0.0,0.0,0.0);
drawstring1(ax+0,140,0,"Demo");
drawstring1(ax+0,130,0,"instructions:");
drawstring1(ax+0,120,0,"0:open barrier");
drawstring1(ax+0,110,0,"v:speed up");
drawstring1(ax+0,100,0,"b:speed down"); //display
```

```
instructions for car bmw glColor3f(0.0,0.0,0.0);
drawstring1(ax+0,60,0,"BMW");
drawstring1(ax+0,50,0,"instructions:");
drawstring1(ax+0,40,0,"l:open barrier");
drawstring1(ax+0,30,0,"z:speed up");
drawstring1(ax+0,20,0,"x:speed down");

// display instruction for left car
glColor3f(0.0,0.0,0.0);
drawstring1(ax+390,60,0,"Nano");
drawstring1(ax+390,50,0,"instructions:");
drawstring1(ax+390,40,0,"r:open barrier");
drawstring1(ax+390,30,0,"n:speed up");
drawstring1(ax+390,20,0,"m:speed down");

if(flagl==0||flag2==0) {//BARRIER
1
glColor3f(0.0,0.0,0.0);
glBegin(GL_POLYGON);
glVertex2f(260,355);
glVertex2f(390,355);
glVertex2f(390,345);
glVertex2f(260,345); glEnd();
glPointSize(20.0);
glColor3f(1.0,0.0,0.0);
glBegin(GL_POINTS);
glVertex2f(420,345); glEnd();
drawstring(437,340,0,"STOP");
```

```
} else
{
glCol
or3f(
0.0,1.
5,0.0
);
glPoi
ntSiz
e(20.
0);
glBe
gin(
GL_
POI
NTS)
;
glVer
tex2f
(420,
345);
glEn
d();
draw
string
(450,
340,0
,"GO
");
}
//BARRIER 2
if(flagr==0||flag3==0) {
glColor3f(0.0,0.0,0.0);
glBegin(GL_POLYGON);
glVertex2f(110,355);
glVertex2f(245,355);
glVertex2f(245,345);
glVertex2f(110,345);
glEnd(); glPointSize(20.0);
glColor3f(1.0,0.0,0.0);
glBegin(GL_POINTS);
glVertex2f(80,345); glEnd();
drawstring(8,330,0,"STOP");
} else { glPointSize(20.0);
glColor3f(0.0,1.5,0.0);
glBegin(GL_POINTS);
glVertex2f(80,345); glEnd();
drawstring(10,330,0,"GO");
}
} }
glFlush();
}
```

10.BIBLOGRAPHY

1. F.S. Hill Jr: “COMPUTER GRAPHICS USING OPENGL, 2nd
2. EDITION”.
3. James D. Foley, John .F. Hughes : “COMPUTER GRAPHICS ,1997.”
4. Donald Hearn and Pauline baker : “COMPUTER GRAPHICS” C-Version,2nd edition, Pearson Education , 2003.
5. Internet source : www.angelfire.com Wikipedia.org.
   Google.
   Opengl.org.