# VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangma, Belagavi – 590 018, Karnataka, India.

A MINI PROJECT ON

# "TOLL COLLECTION BOOTH"

A Mini Project submitted in partial fulfilment of the requirement for the degree of

BACHELOR OF ENGINEERING In COMPUTER SCIENCE & ENGINEERING

Submitted by


AFREEN N 1RG18CS002

ANUSHA K BHAT 1RG18CS003


Under the Guidance of


Mrs. Geetha Pawar

Asst. Prof. Dept. of CSE
RGIT, Bangalore – 36



Department of Computer Science & Engineering

RAJIV GANDHI INSTITUTE OF TECHNOLOGY

Cholanagar, R.T. Nagar Post, Bangalore – 560 036

2020-2021

# RAJIV GANDHI INSTITUTE OF TECHNOLOGY

(Affliliated to Visveshwaraya Technological University)

Cholanagar, R.T. Nagar Post, Bangalore – 560 032

## Department of Computer Science Engineering

This is to certify the Mini Project entitled "STEAM ENGINE" is a bonafide work carried out by ABDUL HAKEYM (1RG18CS001) and CHIRAG V K (1RG18CS013) in partial fulfillment for the award of Bachelor of Engineering in computer Science Engineering, During the year 2020-2021. It is certified that all corrections/suggestions given for the internal assessment have been incorporated in the report. This Mini Project has been approved as it satisfies the academic requirements in respect of the seminar work.

| | |
|---|---|
| Signature of Guide | Signature of HOD |
| Mrs. Geetha Pawar<br>Asst. Professor,<br>Dept. of CSE<br>RGIT, Bangalore | Mrs. Arudra A<br>Associate Prof. & HOD<br>Dept. of CSE<br>RGIT, Bangalore |

EXTERNAL VIVA

| Name of the Examiners | Signature with Date |
|---|---|
| 1. | |
| 2. | |

VISVESVARAYA TECHNOLOGY UNIVERSITY
JNANA SANGAMA, BELAGAVI – 590 018

RAJIV GANDHI INSTITUTE OF TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

# DECLARATION

We hereby declare that the Mini Project work entitled "TOLL COLLECTION BOOTH" submitted to the Visvesvaraya Technological University, Belagavi during the academic year 2020-2021, is a record of an original work done by us under the guidance of Mrs. Geetha Pawar, Assistant Professor, Department of Compute Science and Engineering, Rajiv Gandhi Institute of Technology, Bangalore and this project work is submitted in the partial fulfilment of requirements for the award of the degree of Bachelor of Engineering in Computer Science & Engineering. The results embodied in this thesis have not been submitted to any other University or Institute for award if any degree or diploma.

AFREEN N 1RG18CS002

ANUSHA K BHAT 1RG18CS003

# ACKNOWLEDGEMENT

We take this opportunity to express our sincere gratitude and respect to the Rajiv Gandhi Institute of Technology, Bangalore for providing us an opportunity to carry out our project work.

We express our sincere regards and thanks to Dr. NAGARAJ A M, Principal, RGIT, Bangalore and Mrs. Arudra A, Associate Prof. & HOD of Department of Computer Science and Engineering, RGIT, Bangalore, for their encouragement and Support throughout the project.

With profound sense of gratitude, we acknowledge the guidance and support extended by Mrs. Geetha Pawar, Asst. Prof and Miss. Rajini Kodagali, Asst. Prof, Department of Computer Science & Engineering, RGIT, Bangalore. Her incessant encouragement and valuable technical support have been of immense help in realizing this project. Herr guidance gave us the environment to enhance our knowledge, skills and to reach the pinnacle with sheer determination, dedication and hard work.

We also extend our thanks to the entire faculty of the Department of CSE, RGIT, Bangalore, who have encouraged us throughout the course of Bachelor Degree

AFREEN N 1RG18CS002

ANUSHA K BHAT 1RG18CS003

# ABSTRACT

This project is about the creation of moving primitive 2D & 3D Objects. We are implementing it using different OpenGL libraries combining them together in a required manner to achieve the objective. Our objective to understand the impact of light source, which can be observed on the real 3D environment which we create. By this we will understand the reflection, shadow and surface smoothness on the created primitive object.

## Table of Contents

# CHAPTER 1

# INTRODUCTION

## 1.1 Computer Graphics

Computer graphics is concerned with all aspects of producing pictures or images using a computer. The field began humbly 50 years ago, with the display of few lines on a cathode-ray tube (CRT). Now, we generate images with the computer that indistinguishable from the photographs of real objects. We routinely train pilots with simulated airplanes, generating graphical displays of a virtual environment in real time. Features length movies made entirely by computers have been successful, both critically and financially.

## 1.2 Applications of Computer Graphics

The development of Computer Graphics had been driven both by needs of the user community and by advances in hardware and software. The application of computer graphics are many and varied, we can however divide them into of four major areas:

1. Display of information
2. Design
3. Simulation
4. User interface

Although many application span two or more of these areas, the development of the field was based on separate work in each.

### 1.2.1 Display of information

Medical Imaging possesses interesting and important data analysis problem. Modern imaging technologies such as Computed Tomography (CT), Magnetic

Resonance Imaging (MRI), Ultrasound and Position Emission Tomography (PET), generate 3D data that must be subjected to algorithmic manipulation provide useful information. The field of scientific visualization provides graphical tools that help the researchers interpret the fast quantity of data that generate.

### 1.2.2 Design

Professions such as engineering and architecture are concerned with design. Starting with a set of specifications, engineers and architects seek a cast effective and aesthetic solution that satisfies the specifications.

Design is an interactive process. Design problems are either over determined such that they possess no solution that satisfies all criteria, much less an optimal solution, or undetermined, such that they have multiple solutions that satisfies the design criteria.

### 1.2.3 Simulation and Animation

Graphics system evolved to be capable of generating sophisticated images in real time, engineers and researchers began to use them as simulators. One of the most important use had been training of pilots. The use of special PLSI chips has led to a generation of arcade games as sophisticated as flight simulators.

The simulators can be used for designing the robot, planning its path, and simulating its behavior in complex environment. The success of flight simulators led to the use of Computer Graphics for animation in TV, motion pictures and advertising industries. Entire animated movies can now be made by computer at a cost less than that of movies with traditional ways.

### 1.2.4 User Interface

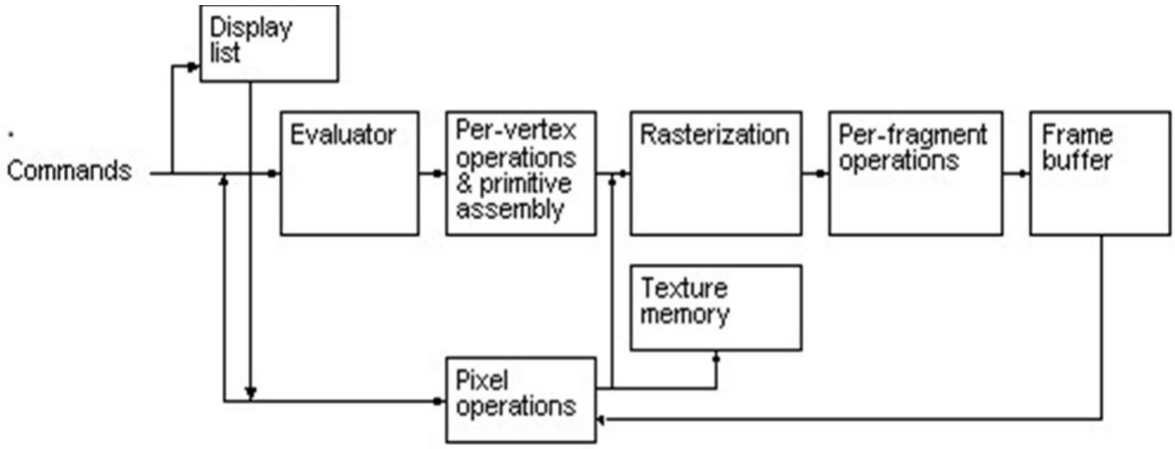
Our interaction with computers has become dominated by visual paradigm that includes icons, menus and pointing devices such as mouse. From user's perspective, winding system such as X and window system, Microsoft windows.

## 1.3 Introduction to OpenGL

Most of the applications will be designed to access OpenGL directly through functions in three libraries. Function in the main GL (or OpenGL in windows) library have names that begin with the letters gl and are stored in a library usually referred to as GL. The second is the OpenGL Utility Library (GLU). This library uses only GL functions but contains code for creating common objects and simplifying programmers prefer not to write the code repeatedly. The GLU library is available in all OpenGL implementations; functions in the GLU library begins with letters glu.

To interface with the window system and to get input from external devices into programs, need at least one more system-specific library that provides the "glue" between the window system and OpenGL. For the X window system, this library is functionality that should be expected in any modern windowing system. For this window system, GLUT will use GLX and the X libraries. The application program, however, can use only GLUT functions and thus be recompiled with the GLUT library for other window systems. We use GLU and Pygame Library for this project.

## 1.4 Block Diagram of OpenGL




## 1.5 Introduction to TOLL COLLECTION BOOTH

The main idea behind our project is to display the concept of toll collecting booth with computer graphics.

We are implementing it using different OpenGL libraries combining them together in a required manner to achieve the objective. The programming language used here is C using OpenGL libraries.

This project will include a toll collecting booth where as soon as the vehicle comes, the barrier will be imposed.

There will be two way path where in at both way we have a toll collecting booth for both directions.

To open the barriers certain key has to pressed, and the barrier will open.

And in the same way we will have another key set to allow the car to move further.

Here we can also increase and decrease the speed of the vehicle by using certain keys. It is a form of road pricing typically implemented to help recoup the costs of road construction and maintenance.

# CHAPTER 2

# SYSTEM ANALYSIS

## 2.1 Existing System

The graphics project is designed using the graphics Utility library, which contains graphics manipulation and creation APIs, which are implemented as part of project. The object created during the course of development of this software consists of many geometric figures like lines, rectangle, square, polygon, sphere, cube, etc. All the figures taken to be an array of X and Y coordinates, similarly rectangles are stored as a set of 4 points etc. System analysis and design is a very important process in any software development process. This process is called as requirement analysis. Once all the requirements are collected, we can then proceed with the actual designing of the system. During design process, we are involved in actual implementation of the system.

## 2.2 Proposed System

The proposed system is developed by using python OpenGL. It is implemented on Windows. The 3D graphics packages designed here provides an interface for the user handling the display. The proposed system provides the visual simulation to understand reflection, impact of lighting surface with smoothness and shadow observation. On Command with Key clicks we could zoom in & out, fill add & remove.

## 2.3 Aim

The main aim is to develop a graphic program that utilizes the features of the computer graphics. The objective is to graphically illustrate the properties like reflection, smooth shading in a real 3D environment. The project aim is to observe these Properties when a primitive object is under the impact of a fixed light source.

# CHAPTER 3

# REQUIREMENT SPECIFICATIONS

## 3.1 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.

## 3.2 Software Requirements

- Any Windows OS
- Editor: Visual Studio Code or Sublime text.
- Language: OpenGL

## 3.3 Hardware Requirements

- Processor: Pentium or Higher. (Preferred x64 bit – Intel i3 processor 1.6GHz)
- Ram: 4GB
- Hard Disk: 500GB (At least 80GB)
- Display: Intel HD graphics (128MB)
- Keyboard, Mouse

# CHAPTER 4

## IMPLEMENTATION

### 4.1 Built-In Functions

4.1.1 **glutInit(&argc, argv):** Used to initialize the glut.

4.1.2 **glutCreateWindow("TOLL COLLECTION BOOTH"):** Creates a window to display the images.

4.1.3 **glutInitDisplayMode() :** In OpenGL the main function of this function is to specify the type of display mode when creating a window.

4.1.4 **glutMainLoop() :** It is the main function the keeps calling and calling the displaying functions and which also keeps our window actually opened.

4.1.5 **glflush():** Forces previously issued OpenGL commands to begin execution.

4.1.6 **glMatrixMode( 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 . Additionally, if the ARB_imaging extension is supported, GL_COLOR is also accepted.

4.1.7 **gluOrtho2D(0, 500, 0, 500):** The function gluOrtho2D can be used to set the clipping area of 2D orthographic view. Objects outside the clipping area will be clipped away and cannot be seen.

4.1.8 **glLoadIdentity():** The glLoadIdentity function replaces the current matrix with the identity matrix.

## 4.2 Header files/imports

```
#include<GL/glut.h>

#include<string.h>

#include<stdio.h>

#include<math.h>
```

## 4.3 Variable Declaration.

### 4.3.1 Display variables

GLfloat w = 800, h = 800, zy1 = 65, zy2 = 430

### 4.3.2 Car variables

GLfloat ay = 0, by1 = 480, by2 = 480, by3 = 460, by4 = 460, by5 = 460, by6 = 460, by7 = 420, by8 = 420, by9 = 420, by10 = 420, by11 = 400, by12 = 400, ax = 0, bx = 0;

GLfloat by13 = 403, by14 = 403, by15 = 416, by16 = 416, by17 = 403, by18 = 403, by19 = 416, by20 = 416, by21 = 463, by22 = 463, by23 = 476, by24 = 476, by25 = 463, by26 = 463, by27 = 476, by28 = 476

## 4.4 Function Declaration

### 4.4.1 Draw a car using glu function:

```
void myCar(GLUquadricObj * object, GLdouble outerRadius, GLdouble innerRadius, GLdouble lenght)
{
glPushMatrix(); gluCylinder(object, outerRadius,
outerRadius, lenght, 20, 1); glPushMatrix(); glRotatef(180,
```

```
0.0, 1.0, 0.0);  gluDisk(object, innerRadius, outerRadius, 20,
1);  glPopMatrix();  glTranslatef(0.0, 0.0, lenght);
gluDisk(object, innerRadius, outerRadius, 20, 1);
glPopMatrix();
}
```

4.4.2 Draw a wheel.

```
 void draw_piston(void)
{  glPushMatrix();  glColor4f(0.3,
0.6, 0.9, 1.0);  glPushMatrix();
glRotatef(90, 0.0, 1.0, 0.0);
glTranslatef(0.0, 0.0, -0.07);
myCylinder(obj, 0.125, 0.06,
0.12);  glPopMatrix();  glRotatef(-
90, 1.0, 0.0, 0.0);  glTranslatef(0.0,
0.0, 0.05);  myCar(obj, 0.06, 0.0,
0.6);  glTranslatef(0.0, 0.0, 0.6);
myCar(obj, 0.2, 0.0, 0.5);
glPopMatrix();
}
```

4.4.3 Draw the road:

```
void draw_road(void)
{  glPushMatrix();
glColor4f(0.9, 0.9, 0.9,
1.0);  myBox(0.5, 3.0, 0.5);
glColor3f(0.5, 0.1, 0.5);
glRotatef(90, 0.0, 1.0, 0.0);
glTranslatef(0.0, 0.9, -0.4);
myCar(obj, 0.1, 0.0, 2);
glPopMatrix();
}
```

4.4.4 Draws the barriers:

```
void draw_barriers(void)
{
glPushMatrix();  glColor4f(0.5,
1.0, 0.5, 0.1);  glRotatef(90, 1.0,
0.0, 0.0);  glTranslatef(0, 0.0,
0.4);  glRotatef(head_angle, 1, 0,
0);  glTranslatef(0, 0.0, -0.4);
myCar(obj, 0.23, 0.21, 1.6);
glRotatef(180, 1.0, 0.0, 0.0);
gluDisk(obj, 0, 0.23, 20, 1);
glPopMatrix();
}
```

4.5 DISPLAY FUNCTION:

4.5.1 **Car color**

```
glColor3f(0.0, 0.0, 1.0);
if (color1 == 1)
glColor3f(0.7, 0.2, 1.5);
```

4.5.2 **Car movement**

```
void mov()
{
if (flagl == 0)
{
if (by9 > 380 || by10 > 380 || by11 > 380 || by12 > 380)
{
by1 = by1 - sizer;
by2 = by2 - sizer;
```

```
by3 = by3 - sizer;
by4 = by4 - sizer;
by5 = by5 - sizer;
by6 = by6 - sizer;
by7 = by7 - sizer;
by8 = by8 - sizer;
by9 = by9 - sizer;
by10 = by10 - sizer;
by11 = by11 - sizer;
by12 = by12 - sizer;
by13 = by13 - sizer;//wheels
by14 = by14 - sizer;
by15 = by15 - sizer;
by16 = by16 - sizer;
by17 = by17 - sizer;
by18 = by18 - sizer;
by19 = by19 - sizer;
by20 = by20 - sizer;
by21 = by21 - sizer;
by22 = by22 - sizer;
by23 = by23 - sizer;
by24 = by24 - sizer;
by25 = by25 - sizer;
by26 = by26 - sizer;
by27 = by27 - sizer;
by28 = by28 - sizer;
```

```
zy2 = zy2 - sizer;
}
}
glutPostRedisplay();
}
```

4.5.3 void display

```
    void display()
{
glClear(GL_COLOR_BUFFER_BIT);
if (flaga == 0)
{
welcome();
}
else
{
if (flagd == 0)
frontscreen();
else
{
glClearColor(0.654321, 0.54321, 0.321, 1.0);
glColor3f(0.658824, 0.658824, 0.658824);
glBegin(GL_POLYGON);
glVertex2f(100, 500);
glVertex2f(400, 500);
glVertex2f(400, 0);
glVertex2f(100, 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, "Nexa");
// display instruction for left car
glColor3f(0.0, 0.0, 0.0);
drawstring1(ax + 405, 60, 0, "Car Nexa");
drawstring1(ax + 405, 50, 0, "instructions:");
drawstring1(ax + 405, 40, 0, "r:open barrier");
drawstring1(ax + 405, 30, 0, "n:speed up");
```

```
drawstring1(ax + 405, 20, 0, "m:speed down");
//Barrier 1
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(110, 390, 0, "TOLL BOOTH 1");
drawstring(10, 338, 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();
}
```

4.5.4 Welcome page and front screen:

```
void welcome()
{
glClear(GL_COLOR_BUFFER_BIT);
glColor3f(0.0, 0.0, 0.0);
drawstring1(100, 300, 0.0, "CGV MINI PROJECT BY:");
glColor3f(0.0, 0.0, 0.0);
drawstring(180, 260, 0.0, "AFREEN & ANUSHA");
glColor3f(0.0, 0.0, 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, 450, 0.0, "CGV MINI PROJECT  (18CSL67)");
glColor3f(0.0, 0.0, 0.0);
drawstring(140, 410, 0.0, "TOLL COLLECTING BOOTH");
glColor3f(0.0, 0.0, 0.0);
```

```
drawstring1(30, 320, 0.0, "DONE BY:");
glColor3f(0.0, 0.0, 0.0);
glColor3f(0.0, 0.0, 0.0);
drawstring2(50, 265, 0.0, "AFREEN N (1RG18CS002)");
drawstring2(50, 290, 0.0, "ANUSHA K BHAT (1RG18CS003)");
glColor3f(0.0, 0.0, 0.0);
drawstring1(30, 220, 0.0, "Under the Guidance of:");
glColor3f(0.0, 0.0, 0.0);
drawstring2(50, 200, 0.0, "Prof. Geeta Nayak");
glColor3f(0.0, 0.0, 0.0);
drawstring1(300, 90, 0.0, "For Next press : 2");
glFlush();
}
```

4.6 Main Function():

```
int main(int argc, char** argv)
{
glutInit(&argc, argv);
glutInitWindowSize(650, 650);
glutInitWindowPosition(200, 20);
glutCreateWindow("TOLL COLLECTION BOOTH");
init();
glutInitDisplayMode(GLUT_RGB | GLUT_SINGLE);
glutDisplayFunc(display);
glutMouseFunc(mouse);
```

```
glutReshapeFunc(reshape);
glutIdleFunc(mov);
glutKeyboardFunc(key);
glutMainLoop();
}
```

4.7 Key Board Functions:

```
static void key(unsigned char key1, int x, int y)
{
switch (key1)
{
case 'Q':
case 'q':
exit(0);
break;
case 'r':
case 'R':
flagl = 1;
flag2 = 1;
break;
case 'l':
case 'L':
flagr = 1;
flag3 = 1;
break;
case 'z':
case 'Z':
size = size * 2;
```

```
break;
case 'x':
case 'X':
size = size / 2;
break;
case 'n':
case 'N':
sizer = sizer * 2;
break;
case 'm':
case 'M':
sizer = sizer / 2;
break;
case 'v':
case 'V':
sizec = sizec * 2;
break;
case 'b':
case 'B':
sizec = sizec / 2;
break;
case '1':flaga = 1;
break;
case '2':flagd = 1;
break;
case '0':flagc = 1;
break;
}
}
```

4.8 Mouse Functions:

```
void mouse(int btn, int state, int x, int y)
{
if (btn == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
flagr = 1;
flag3 = 1;
if (btn == GLUT_RIGHT_BUTTON && state == GLUT_DOWN)
flagl = 1;
flag2 = 1;
}
```

4.9 Reshape Function:

```
void reshape(int w, int h)
{
glViewport(0, 0, w, h);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
if (w <= h)
gluOrtho2D(0.0, 500.0, 0.0 * (GLfloat)h / (GLfloat)w, 500.0 * (GLfloat)h / (GLfloat)w);
else
gluOrtho2D(0.0 * (GLfloat)w / (GLfloat)h, 500.0 * (GLfloat)w / (GLfloat)h, 0.0, 500.0);
glMatrixMode(GL_MODELVIEW);
}
```

# CHAPTER 5

## SCREENSHOTS

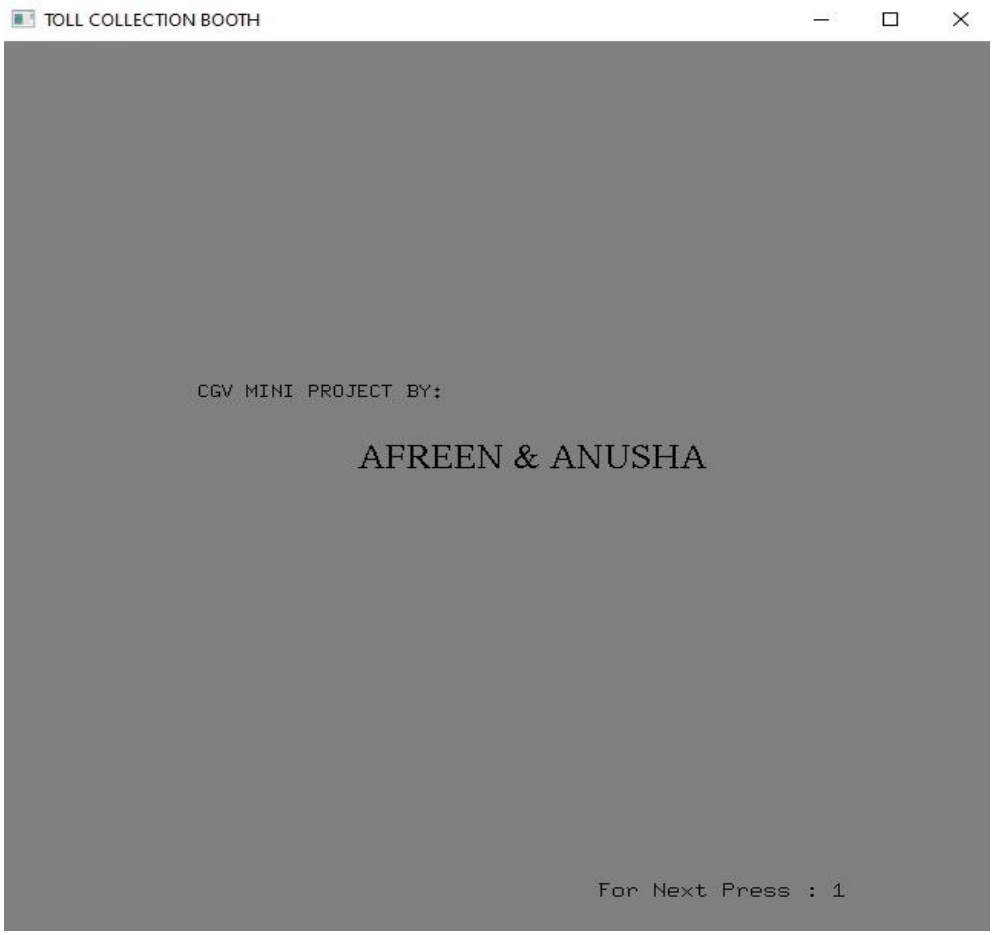



Fig 1: Welcome page

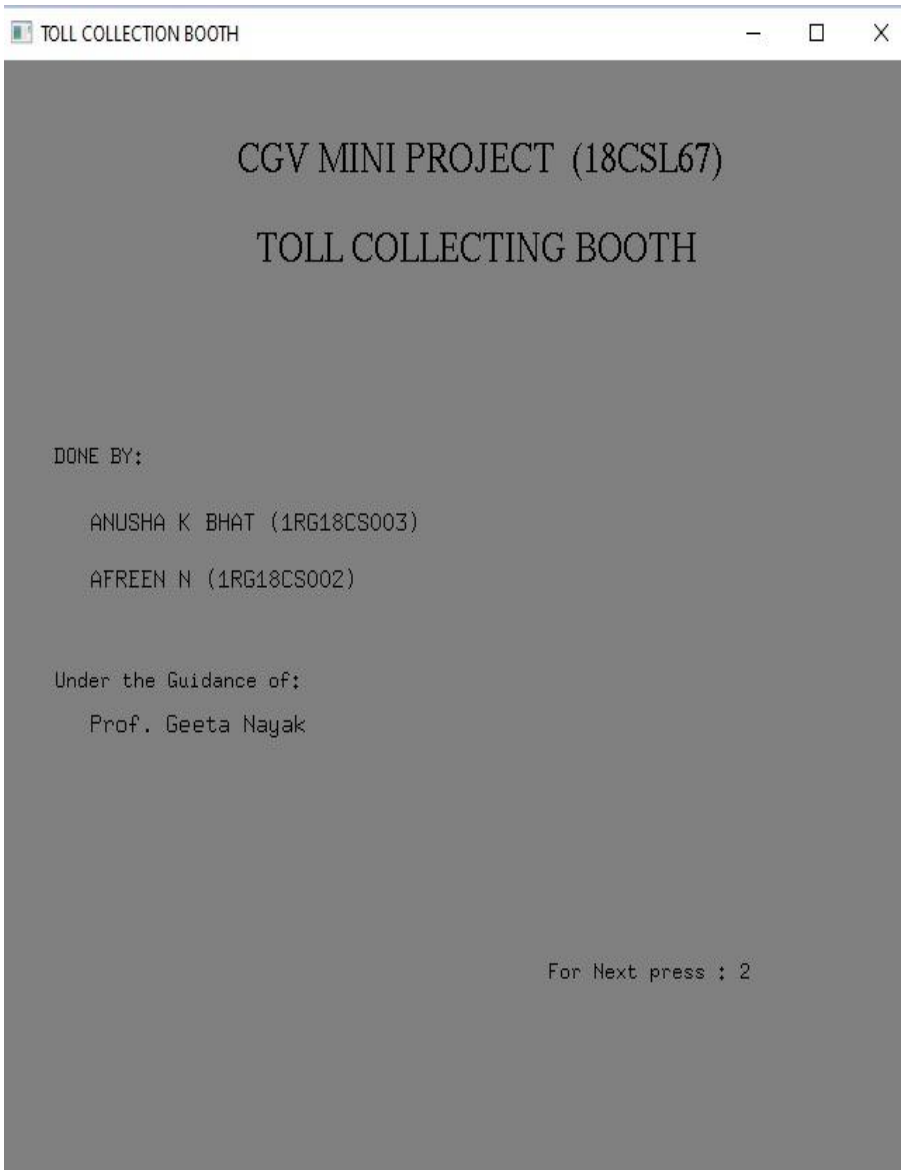



Fig 2: Front Screen

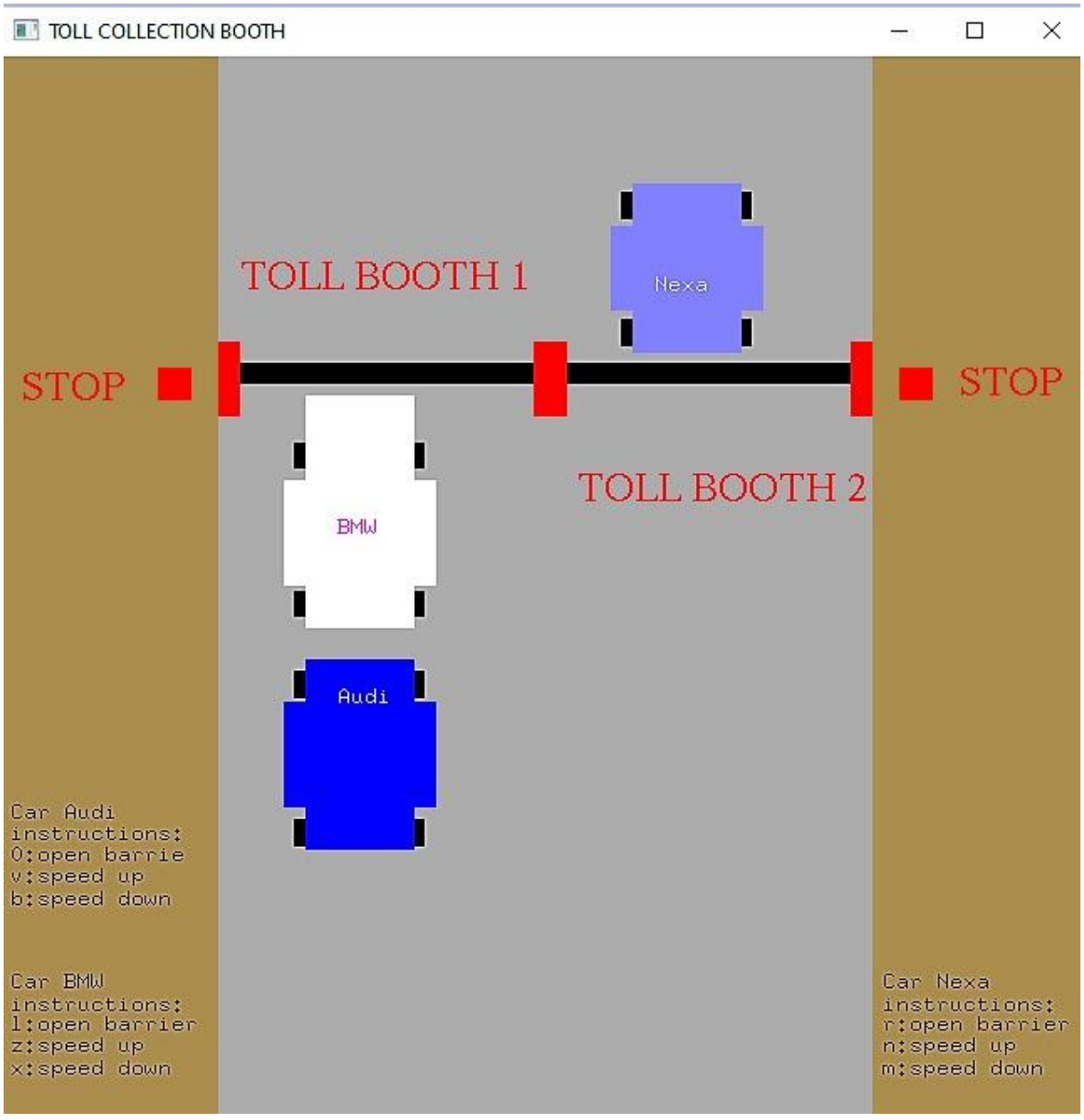



Fig 3: Main Page (Both the barriers are closed)

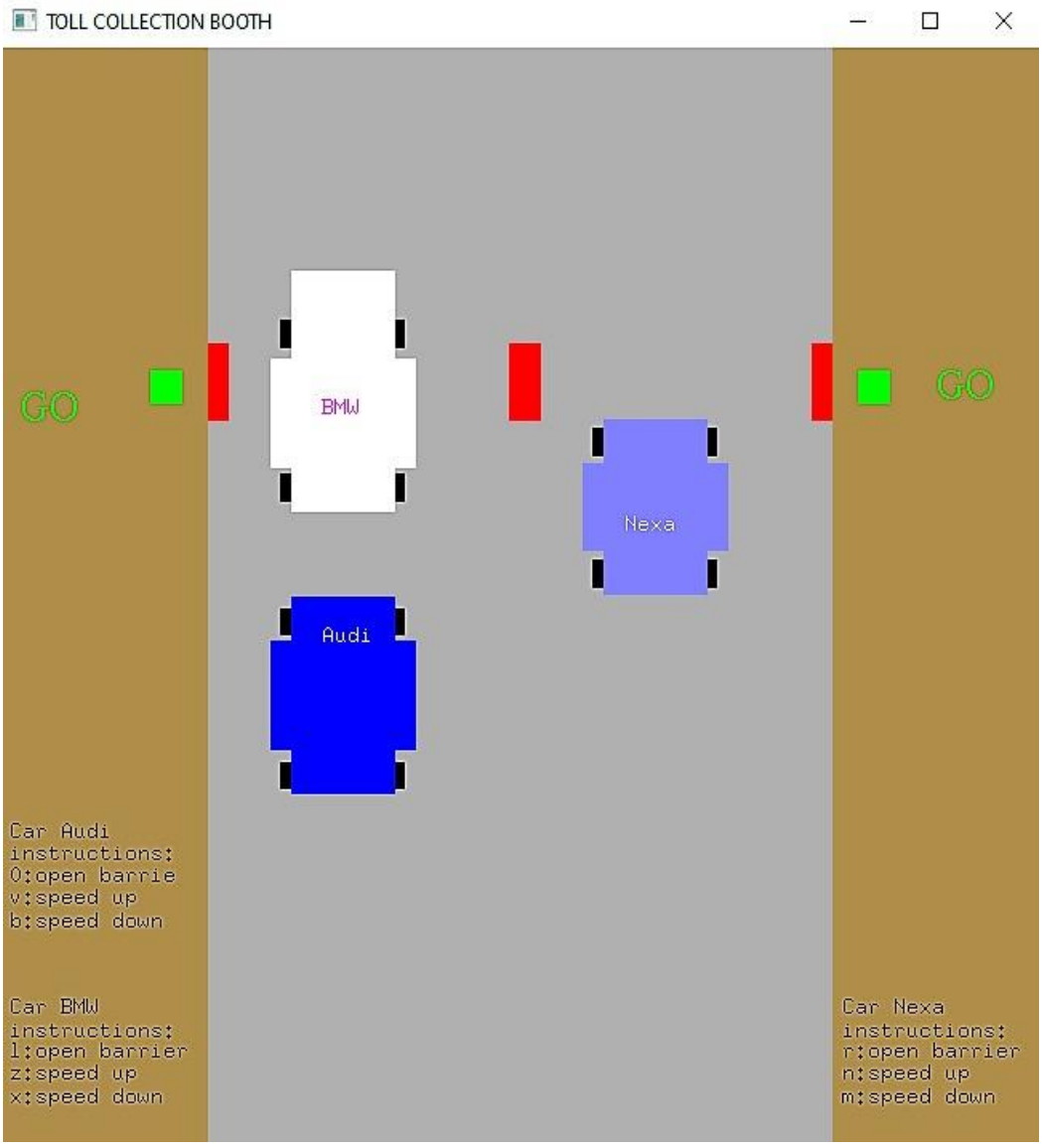



Fig 4: Main Page (Both the barriers are opened)

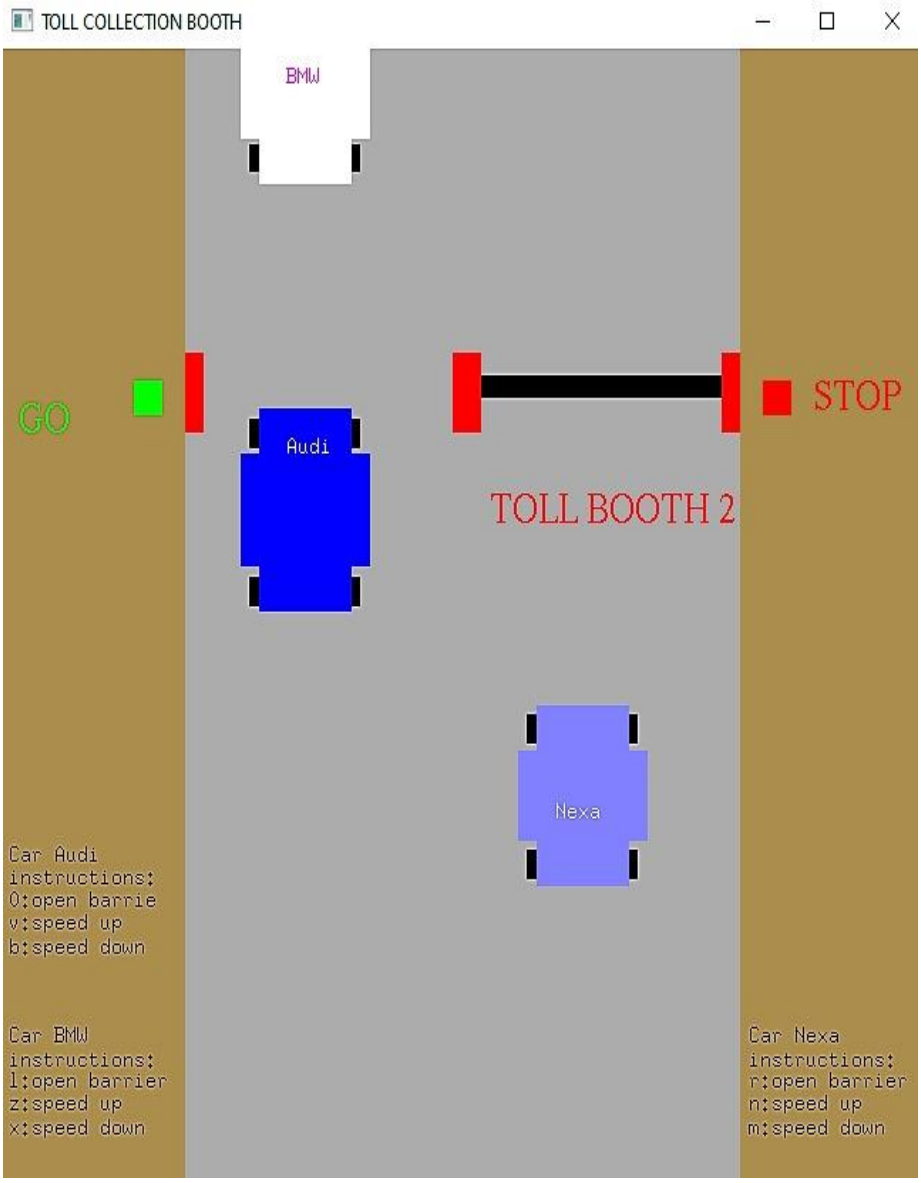



Fig 5: Main Page (Barrier 1 is open for the car Audi)

# CHAPTER 6

## CONCLUSION

This project allows the user to understand the basic purpose of toll booth collection. The user can open and close the barrier accordingly when the car moves. The user can also speed up and down of the car.

## FUTURE ENHANCEMENT:

In the future, the user can implement a display board where it displays the number of cars passed by and also can display the maintenance cost. The user can also add different types of vehicles like trucks, buses, etc.

## BIBLIOGRAPHY

Textbook Referred:

- Interactive Computer Graphics, A Top-Down Approach with OpenGL – Edward Angel, 5th Edition, Addison- Wesley- 2008
- The offical Guide to Learning OpenGL, by Jackie Nedier, Tom Davis, Mason Woo (THE RED BOOK)
- OpenGL Super Bible by Richard S, Wright Jr and Michael Sweet.
- Donald Hearn & Pauline Baker: Computer Graphics with OpenGL Version, 3rd or 4th Edition, Pearson Education, 2011

Website Referred:

- www.stackoverflow.com
- www.youtube.com
- http://www.opengl.org/ online man pages.
- www.google.com