CHAPTER 1

INTRODUCTION

1.1 AIM

The main aim of the transport Simulation Computer Graphics Mini Project is to illustrate the concepts and usage of pre-built functions in OpenGL. Simulation of a bus is being done using computer graphics. Car Simulation is a project where we can also change the color, name, of the bus and other graphics. We can also interchange the boy or girl picking and dropping by the bus

1.2 INTRODUCTION TO OPENGL

OpenGL (Open Graphics Library[3]) is a cross-language, cross-platform application programming interface (API) for rendering 2D and 3D vector graphics. The API is typically used to interact with a graphics processing unit (GPU), to achieve hardware-accelerated rendering.

OpenGL (Open Graphics Library) is a software interface to graphics hardware. The interface consists of over 250 different function calls which can be used to draw complex two and threedimensional scenes from simple geometric primitives.

OpenGL draws primitives—points, line segments, or polygons—subject to several selectable modes. You can control modes independently of each other; that is, setting one mode doesn't affect whether other modes are set. Primitives are specified, modes are set, and other OpenGL operations are described by issuing commands in the form of function calls.

Primitives are defined by a group of one or more vertices.

A vertex defines a point, an endpoint of a line, or a corner of a polygon where two edges meet. Data is associated with a vertex, and each vertex and its associated data are processed independently, in order, and in the same way.

The type of clipping depends on which primitive the group of vertices represents.

Commands are always processed in the order in which they are received, although there may be an indeterminate delay before a command takes effect. This means that each primitive is drawn completely before any subsequent command takes effect. It also means that statequerying commands return data that's consistent with complete execution of all previously issued OpenGL commands.

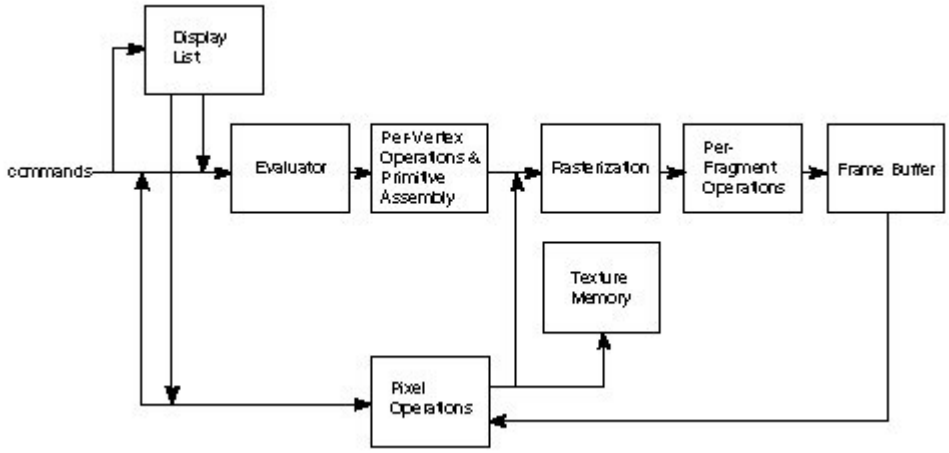


Figure 1.2.1 openGL

1.3 PROJECT RELATED CONCEPTS

OpenGL is an application program interface (API) offering various functions to implement primitives, models and images. This offer functions to create and manipulate render lighting, coloring, viewing the models. OpenGL offers different coordinate system and frames.

OpenGL offers translation, rotation and scaling of objects. Most of our applications will be designed to access OpenGL directly through functions in three libraries.

They are:

1. Main GL: Library has names that begin with the letter gl and are stored in a library usually referred to as GL
2. . OpenGL Utility Library (GLU): This library uses only GL functions but contains code for creating common objects and simplifying viewing.
3. OpenGL Utility Toolkit (GLUT): This provides the minimum functionality that should be accepted in any modern windowing system.

The basic interfaces in this project is:

i) User Interface: User able to control the movement of the bus by using keyboard arrow keys

ii) Selection: User can change the type of student picking up from either male or female.

iii) Pickup/drop: Whenever a bus stop signs arrive the bus automatically picks and drops the student

1.4 INTERFACE

1) Mouse_Interface:

   Left-click button: this action is used in the beginning of the simulation in order to start to the next phase we have to click the left-button of mouse

2) Keyboard Interface:

   i) Left-arrow key: This action is used to move the bus in only one direction from left to right

CHAPTER 2

REQUIREMENT SPECIFICATION

2.1 SOFTWARE REQUIREMENTS:

Programming language – C/C++ using OpenGL.

- Operating system – Linux operating system.
- Compiler – C Compiler
- Graphics library – GL/glut.h
- OpenGL 2.0

2.2 HARDWARE REQUIREMENTS:

- Microprocessor: 1.0 GHz and above CPU based on either AMD or INTEL Microprocessor Architecture
- Main memory : 2 GB RAM
- Hard Disk : 40 GB
- Hard disk speed in RPM:5400 RPM
- Keyboard: QWERTY Keyboard
- Mouse :2 or 3 Button mouse
- Monitor : 1024 x 768 display resolution

CHAPTER 3

DESIGN

3.1 Window design :

Totally 3 interfaces are used : They are

- **Introduction window :** It is a window that shows the information regarding the developers of the project.
- **Output window (1) :** Main outer window that holds the content. It includes the first working interface in which the operation takes place. The operation includes the bus picking of the girl from the bus stop and moving ahead.
- **Output window (2) :** The second working interface includes the bus dropping of the girl in front of the college and moving towards the parking slot.

3.2 Operations :

The main operation includes movement of the bus based on the instructions specified. The key used for this operation is **right arrow** which helps in the movement of the bus.

3.3 Animations :

The animations included in the interface contains some of the pre-defined functions such as **glutSolidSphere, glutSolidTorus** etc.. and some of the user defined functions such as **void wheel(), void lamppost()** etc.

3.4 FLOW CHART

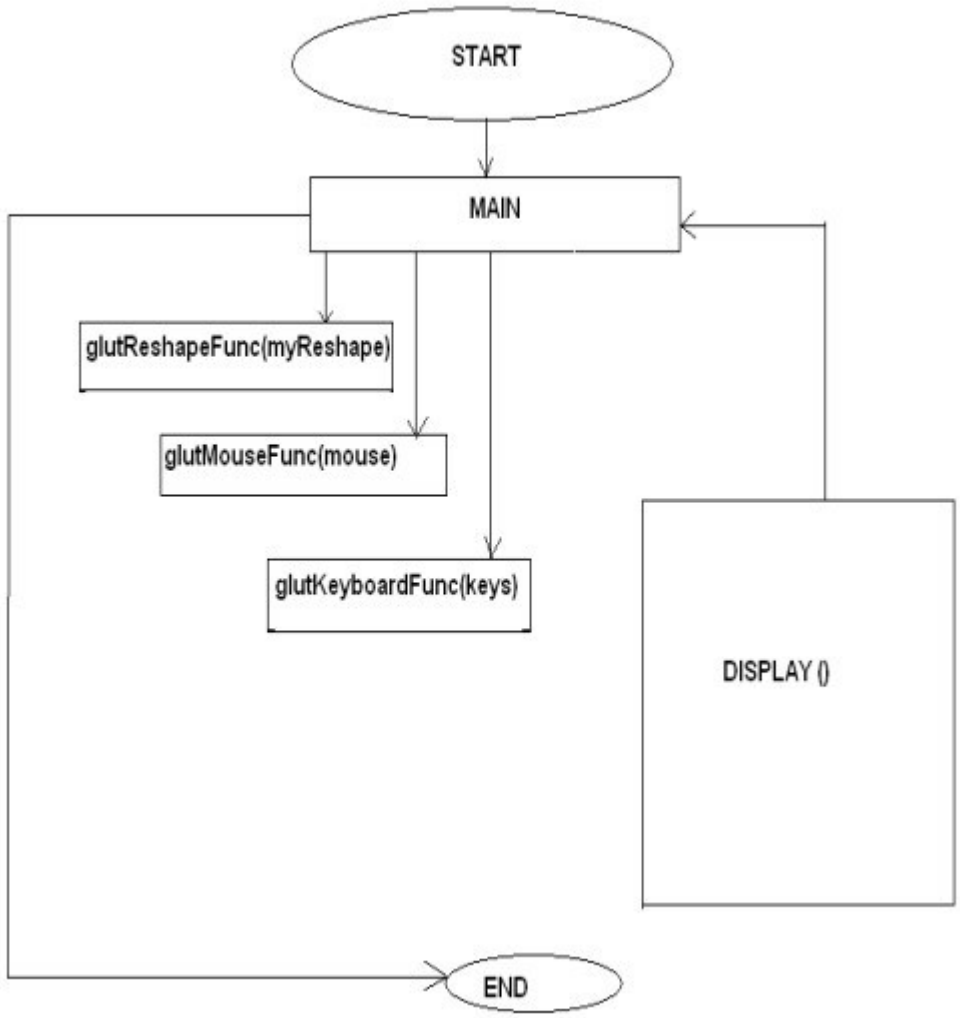


FIGURE 3.4.1

CHAPTER 4

IMPLEMENTATION

4.1 FUNCTIONS USED

1. **glRasterPos3f(x,y,z):** OpenGL maintains a 3-D position in window coordinates. This position, called the raster position, is maintained with subpixel accuracy. The current raster position consists of three window coordinates (*x, y, z*), a clip coordinate *w* value, an eye coordinate distance, a valid bit, and associated color data and texture coordinates

2. **glutBitmapCharacter(a,b) :** It automatically sets the OpenGL unpack pixel storage modes it needs appropriately and saves and restores the previous modes before returning. The generated call to glBitmap will adjust the current raster position based on the width of the character.
3. **glColor3ub(a,b,c) :** glColor3ub is **the version for unsigned char in C/C++, that is a 8bit integer without a sign**. glColor3b is the version for char which is a signed 8-bit integer. 255 = 0xFF is actually -1 when interpreted as a signed 8-bit integer.

4. **glutSolidTorus :** GLUT provides function glutSolidTorus to render a Solid Torus and glutWireTorus to render a Wire Frame Torus. glutSolidTorus and glutWireTorus render a solid or wireframe torus (doughnut) respectively centered at the modeling coordinates origin whose axis is aligned with the Z axis. Here are the syntax of both the above functions.
5. **glutSolidCube** : The glutSolidCube() function draws a solid-shaded cube with side-length given by width. The vertices of the cube are at (+/- width/2, +/- width/2, +/- width/2), so that the cube is centered at the origin.

6. **glutSolidCone :** glutSolidCone and glutWireCone render a solid or wireframe cone respectively oriented along the Z axis. The base of the cone is placed at Z = 0, and the

top at Z = height. The cone is subdivided around the Z axis into slices, and along the Z axis into stacks.

7. **glutSolidSphere :** glutSolidSphere and glutWireSphere render a solid or wireframe sphere respectively. Usage. void glutSolidSphere(GLdouble radius, GLint slices, GLint stacks);

8. **glPushMatrix :** The glPushMatrix function pushes the current matrix stack down by one, duplicating the current matrix. That is, after a glPushMatrix call, the matrix on the top of the stack is identical to the one below it.

9. **glPopMatrix :** The glPopMatrix function pops the current matrix stack, replacing the current matrix with the one below it on the stack. Initially, each of the stacks contains one matrix, an identity matrix.

10. **glRasterPos3f(x,y,z) :** The current raster position consists of three window coordinates (x, y, z), a clip coordinate w value, an eye coordinate distance, a valid bit, and associated color data and texture coordinates.

4.2 EXISTING SYSTEM

Existing system for a graphics is the TC++. This system will support only the 2D graphics. 2D graphics package being designed should be easy to use and understand. It should provide

various options such as free hand drawing, line drawing, polygon drawing, filled polygons, flood fill, translation, rotation, scaling, clipping etc.

Even though these properties were supported, it was difficult to render 2D graphics cannot be very difficult to get a 3 Dimensional object. Even the effects like lighting, shading cannot be provided.

So we go for Microsoft Visual Studio software.

4.3 PROPOSED SYSTEM

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

1. Open GL is designed as a streamlined.
2. It's a hardware independent interface i.e it can be implemented on many different hardware platforms.
3. With Open GL 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 functions.

CHAPTER 5

SOURCE CODE

```
#include <stdlib.h>
```

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

#include<string.h>

#include<stdio.h>


int x=-150,o=0,xd=-150; int x1=0,z=0; float a=0,a1=0,moving,angle=0;

float z5=0,u=0,flag12=0,begin; float k=0,y2=0; int

flag=0,flag55=0,var=0,flag1=0,flag551=0,vari=0,vard=0,varid=0,then=0

; float p=0.0,q=0.0; #define maxx 10

#define maxy 12

#define dx 27.7

#define dy 12

GLfloat xangle=0.0,yangle=0.0,zangle=0.0;   /* axis angles */


void   wheel();

void wheel2();


// FUNCTION cube

void polygon(int a, int b, int c , int d,int E,int f ){

        glBegin(GL_POLYGON);

        glColor3fv(colors[E]);

        glVertex3fv(vertices[a]);
```

```
        glVertex3fv(vertices[b]);
        glVertex3fv(vertices[c]);
        glVertex3fv(vertices[d]);
                if(f!=0)
                 glVertex3fv(vertices[f]);
                 glEnd();
}          void
colorcube(){
        inti;
        wheel1();
         polygon(0,1,5,4,0,0);
         polygon(13,14,18,0,0,0);      polygon(15,16,17,18,2,0);      polygon(16,11,2,1,0,17);
                polygon(0,4,8,13,0,0);          polygon(1,10,9,5,0,0);          polygon(9,10,2,6,1,0);
                polygon(4,5,9,8,0,0);  polygon(8,9,6,12,1,7);          glColor3ub(100,40,50);
                for(i=0;i<=180;i+=45)
         {
                glBegin(GL_LINES);
        glVertex3f(180+i,447,70);
        glVertex3f(180+i,500,70);
                 glEnd();
         }
```

```
        polygon(13,8,7,3,1,0);

        polygon(3,15,14,13,1,0);

        polygon(6,2,11,12,0,0);

        polygon(11,3,7,12,0,0);

wheel2();

} void

bus_stop();

void

road2();

void

text2();

void

text3();

void

text4d();

void

text5d();

void line();

TION Woman

void tree12();

void tree1()
```

```
void

woman();

void man();

void

lamppost();

void

lamppost1();

void

lamppost2();

void

lamppost4();

void

wheel1d();

void

wheel2d();

void

polygond(int

a, int b, int c ,

int d,int E,int

f); wheel2d();

void
```

```
road2d();
void textd();
void text2d();
void text3d();
void
buildingd();
void lined();
void walld();

void bus_move();
{
        if(x<50)
{               x+=3;
                glPushMatrix();
                glTranslatef(-100,0,-90);
                woman();
               glPopMatrix();
       glPushMatrix();
              glTranslatef(x,0,0);
               wheel1();
```

```
        colorcube();
        wheel2();
        line();
        text1();
        glPopMatrix();
}
         if(x>=50)
                  vari=1;
         if(flag55==1 )
        {
        x+=6;
                 glPushMatrix();
        glTranslatef(x,0,0);
                 wheel1();

        colorcube();
        wheel2();
        line();
        text1();
                  glPopMatrix();
         }
```

```
        if(x>=865)
                var=1;
}           void
bus_moved()
{
        if(xd>50)
{
                xd+=3;
           glPushMatrix();
        glTranslatef(-100,0,-90);
           womand();
        glPopMatrix();
        glPushMatrix();
        glTranslatef(xd,0,0);
                wheel1d();

        colorcubed();
        wheel2d();
        lined();
              text1d();
                glPopMatrix();
```

```
}
        if(flag551==1)
{
                xd+=5;
               glPushMatrix();
        glTranslatef(xd,0,0);
               wheel1d();

        colorcubed();
        wheel2d();
        lined();
        text1d();
        glPopMatrix();
         }
         if(xd>50)
                varid=1;
}
//static void SpecialKeyFunc( int Key, int x, int y );
static void SpecialKeyFunc( int Key, int x, int y )
{
        switch ( Key )
```

```
{
        case GLUT_KEY_UP:                       /*move to right */
                //bus_move();
           //p+=50;
glutPostRedisplay();
                break;
        case GLUT_KEY_RIGHT:
                //rota();                       /
glutPostRedisplay();
                break;
        }
}           void
display(void)
{
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();       glOrtho(0,
1000, 10.0, 650,-2000,1500);
         glMatrixMode(GL_MODELVIEW);
         glClearColor(1.0, 1, 1.0, 1.0);
    glClear( GL_DEPTH_BUFFER_BIT | GL_COLOR_BUFFER_BIT);
        if(then==0)
```

```
{
        intro();
           texti();
        }
        if(then==1)
        {
        if(flag)
        {
                  glPushMatrix();
                 glTranslatef(-1.0,0.0,-3.5);
       glRotatef(xangle+25,1.0,0.0,0.0);
       glRotatef(yangle,0.0,1.0,0.0);
             glRotatef(zangle,0.0,0.0,1.0);
                  road2();
                 glPushMatrix();
       glTranslatef(0,00,-50);
                  bus_stop();
                  glPopMatrix();
                  tree1();
                  tree12();
    man();
```

```
                lamppost();


        lamppost1();
        lamppost2();


                text2();
        bus_move();
        // text1();
  glPopMatrix();        }
        else {
        road2();
        bus_stop();
        text();
       tree1();
                 tree12();
                man();
                 lamppost();
                 lamppost1();
                 lamppost2();


        lamppost4();
```

```
        text2();
        bus_move();
        flag55=1;
        }
  if(vari==1)
        {
                text3();
        if(x==865)
                        vari=0;
        }
        if(var==1)
        {
      if(flag1)
        {
                glPushMatrix();
            glTranslatef(-1.0,0.0,-3.5);
     glRotatef(xangle+25,1.0,0.0,0.0);
         glRotatef(yangle,0.0,1.0,0.0);
         glRotatef(zangle,0.0,0.0,1.0);
```

```
                road2d();
                buildingd();
                walld();
            text2d();
                gated();
                treed();
                tree1d();
                tree2d();
            shrubd();
                shrub1d();
                shrub2d();
                shrub3d();
                shrub4d();
                stopd();
                text3d();
                text4d();
                 bus_moved();
              glPopMatrix();
        }
        else
{
```

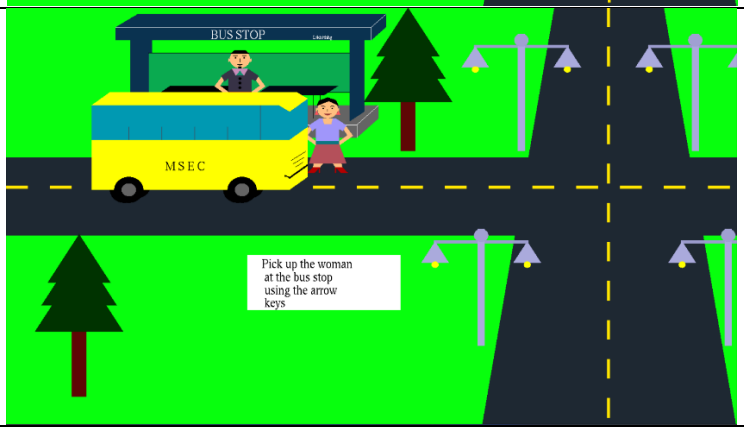
```
            road2d();
            textd();
            buildingd();
            walld();
            text2d();
            gated();
            treed();
            tree1d();
            tree2d();
            shrubd();
            shrub1d();
            shrub2d();
            shrub3d();
            shrub4d();
            stopd();
           text3d();
             text4d();

    bus_moved();
       flag551=1;
```

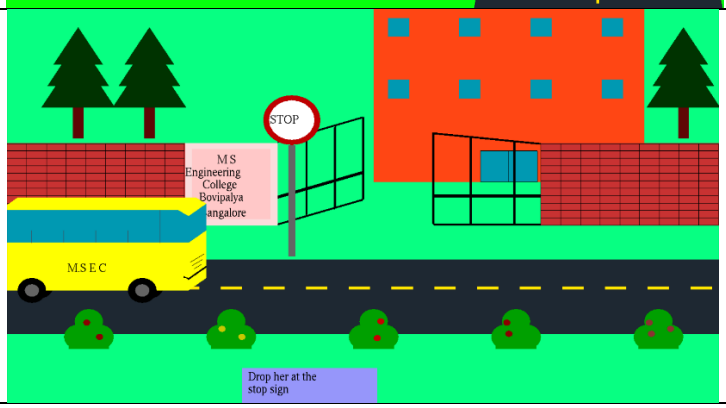
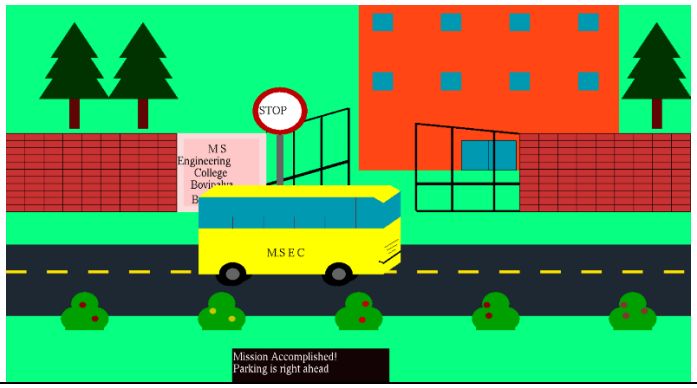
```
        }
        if(varid==1)
                text5d();
         }
         }
        glFlush();
        glutSwapBuffers();
}   void   myreshape(int
w,int h)
{
        glViewport(0,0,w,h);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
         if(w<=h)
                 glOrtho(-2.0,2.0,-2.0*(GLfloat)h/(GLfloat)w,2.0*(GLfloat)h/(GLfloat)w,-
10.0,10.0);
         else
              glOrtho(-2.0*(GLfloat)w/(GLfloat)h,2.0*(GLfloat)w/(GLfloat)h,-2.0,2.0,-
10.0,10.0);
         glMatrixMode(GL_MODELVIEW);
}
```

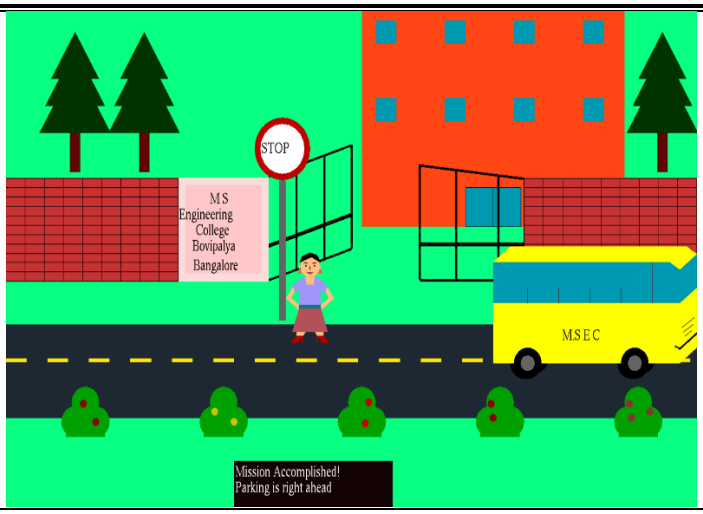
```
/*****    main    ****/  int

main(int       argc,       char

**argv)

 {

        glutInit(&argc, argv);            glutInitDisplayMode(GLUT_DOUBLE |

GLUT_RGBA | GLUT_DEPTH );              glutInitWindowSize(1000,650);

        glutInitWindowPosition(0,0);            glutCreateWindow("BUS STOP");

        glutDisplayFunc(display);     glutMouseFunc(mouse);

        glutSpecialFunc( SpecialKeyFunc );

        glutReshapeFunc(myreshape);

 glutMainLoop();

return 1;                  }
```

CHAPTER 6

TEST CASES

Test No	Snapshot	Description
1		Introduction
2		Bus stop, Bus, boy, girl, street lights, trees, roads
3		While picking the girl by bus

4		After Picking the girl
5		Leaving the bus stop, and travelling to the college
6		Arrival of bus to the college
7		Dropping the girl at college

8		Bus leaving the college to the parking zone.

CHAPTER 7

SNAPSHOTS

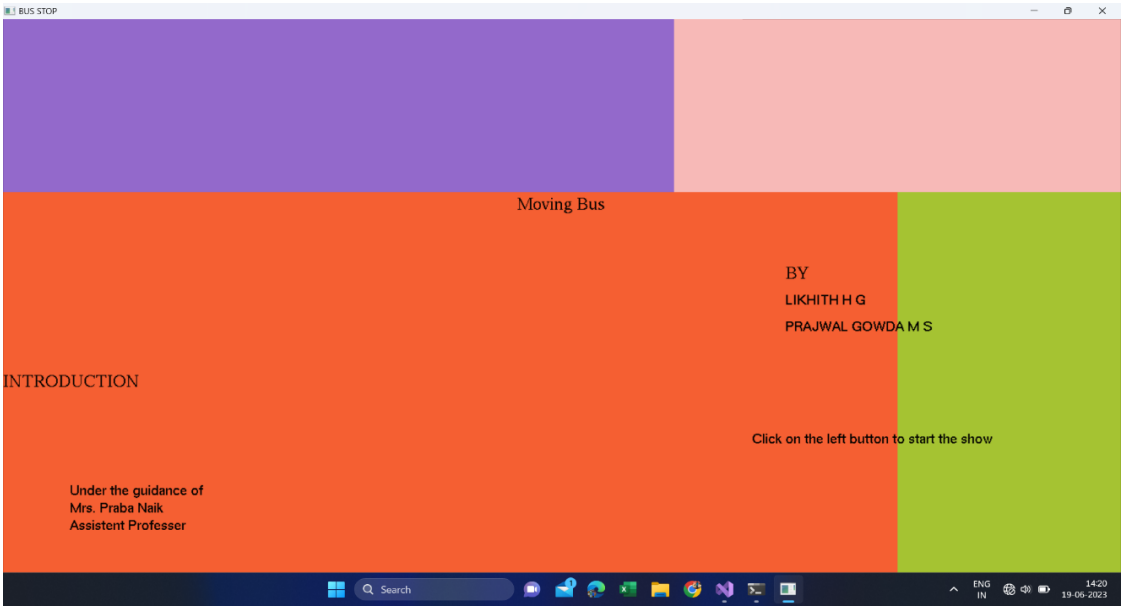


FIGURE 7.1.1 INITIAL DISPLAY

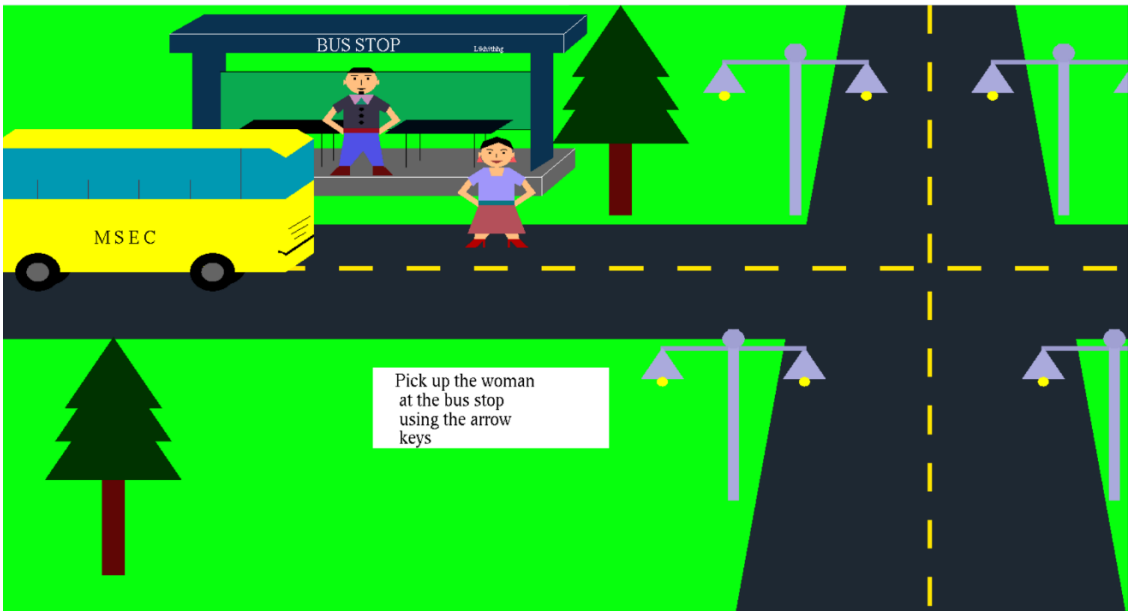


FIGURE 7.1.2 BUS ARRIVING AT BUS STOP

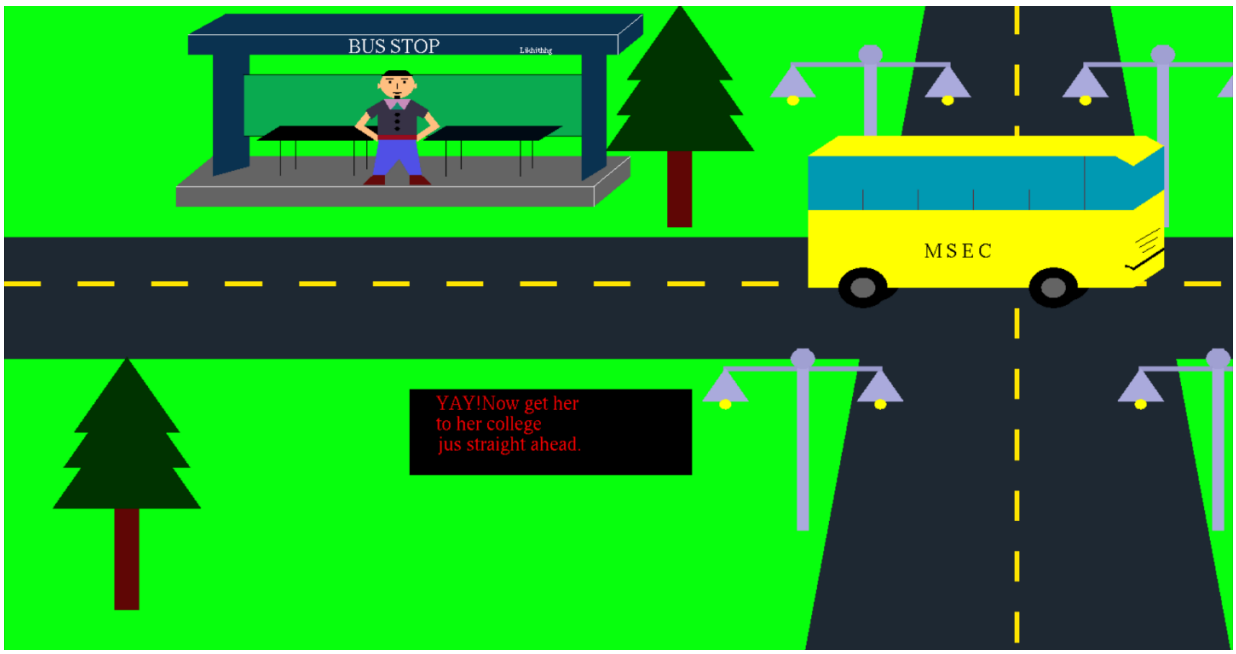


FIGURE 7.1.3 BUS LEAVING BUS STOP

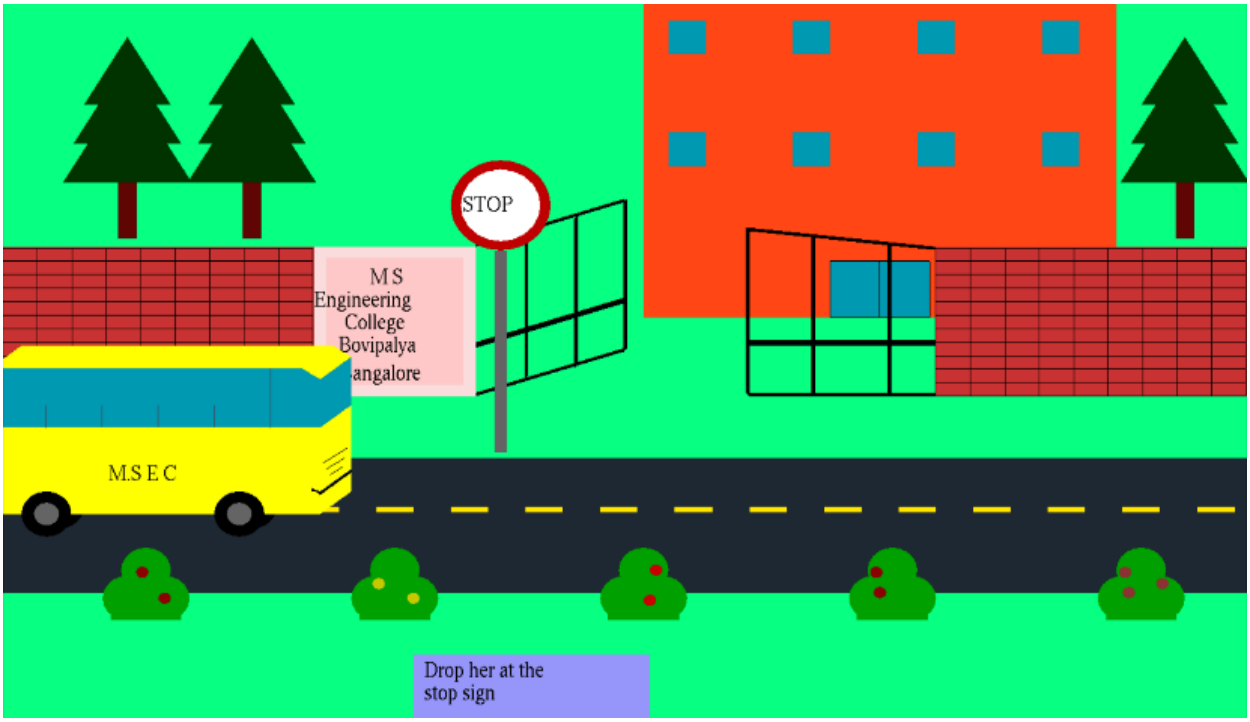


FIGURE 7.1.4 BUS ARRIVING AT SVIT COLLEGE

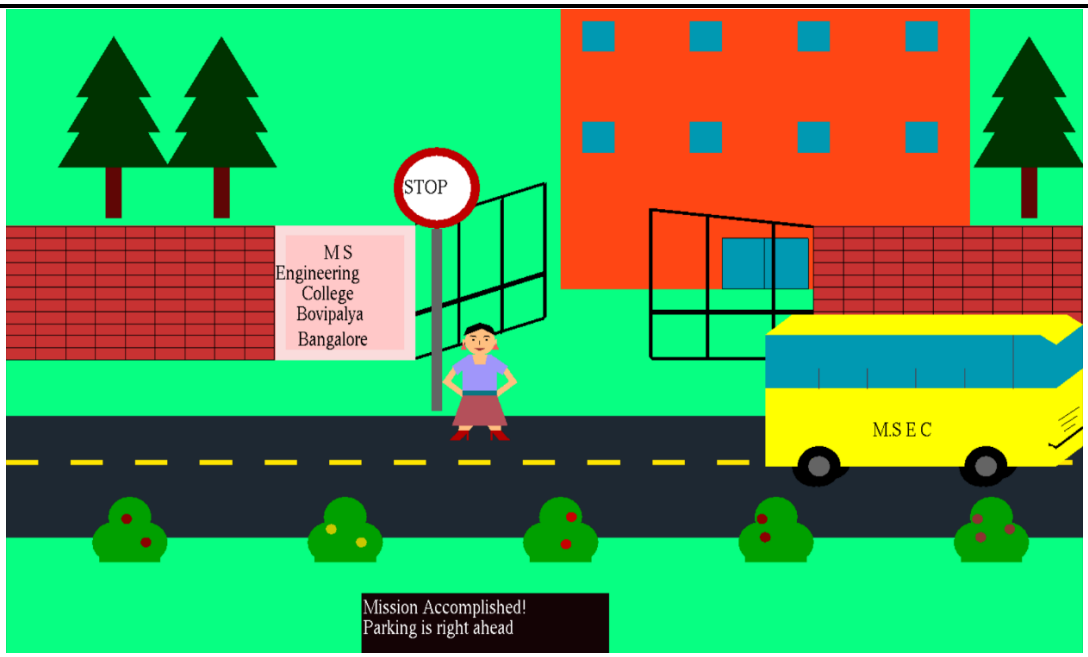


FIGURE 7.1.5 BUS LEAVING AT SVIT COLLEGE

CONLCUSION

An attempt has been made to develop an OpenGL package which meets necessary requirements of the user successfully. Since it is user friendly, it enables the user to interact efficiently and easily.

The development of the mini project has given us a good exposure to OpenGL by which we have learnt some of the technique which help in development of animated pictures, gaming.

Hence it is helpful for us even to take up this field as our career too and develop some other features in OpenGL and provide as a token of contribution to the graphics world.

REFERENCES

1 https://en.wikipedia.org/wiki/OpenGL

2 https://www.javatpoint.com/application-of-computer-graphics

3 https://cglabprojects.blogspot.com/2014/04/moving-bus-animation.html

4 https://github.com/abd-abdullah/Computer-Graphics-BUS-STOP

5 https://devfolio.co/@sowmya

6 https://www.researchgate.net/publication/344285614_A_PROJECT_ON_3D_CAR_SIMULATION/link/5f63657f299bf1b53edba413/download

7 https://solutionsbyabhishek.blogspot.com/2016/12/computer-graphics-bus-stop.html