CHAPTER 1

INTRODUCTION

1.1Computer Graphics

Computer Graphics deals with the study of technology and techniques for generating and displaying images of the natural and synthetic objects. It is an exciting field with a wide range of application including entertainment, graphical user interfaces, industrial modelling, molecular modelling, surgery planning, virtual reality and visualization.

Computer graphics is concerned with all aspects of producing images using a computer. It concerns with the pictorial synthesis of real or imaginary objects from their computer-based models. A computer image is usually represented as a discrete grid of picture elements called pixels. The number of pixels determines the resolution of the image.

Application of Computer Graphics:

- Allows you to create rough freehand drawings.
- The images are stored as bit maps and can easily be edited.
- Enables you to sequence a series of image to simulate movement. Each image is like a frame in a movie. It can be defined as a simulation of movement created by displaying a series of picture or frames.
- A cartoon on television is one example of animation. Animation on computer is one of the ingredients of multimedia presentations.
- Enables architects and engineers to draft design (CAD).

1.2 OpenGL

- OpenGL is a software interface to graphics hardware. This interface consists of about 150 distinct commands that you to specify the objects and operations needed to produce interactive three-dimensional applications.
- OpenGL is open Graphics Library. It is an open industry-standard API for hardware accelerated graphics drawing. It is implemented by graphics-card vendors.
- OpenGL is designed as a streamlined, hardware-independent interface to be implemented on many different hardware platforms.
- OpenGL's main purpose is to render two- and three-dimensional objects into a frame buffer. These objects are described as sequences of vertices (which define geometric objects) or pixels (which define images).
- OpenGL is a state-machine. Remember state-machines. Once a state is set, it remains active until the state is changed via a transition. A transition in OpenGL equals a function-call. A state in OpenGL is defined by the OpenGL-object which are current.
- A sophisticated library that provides these features could certainly be built on top of OpenGL.
- The libraries are

  AGL, GLX, WGL – interaction between OpenGL and windowing system.

  GLU (OpenGL Utility Library) – part of OpenGL.

  GLUT (OpenGL utility Toolkit) – portable windowing API. It is not officially part of OpenGL.

Characteristics

- 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.

Computer Graphics Library Organisation

OpenGL stands for Open Graphics Library. Graphics Library is a collection of APIs (Application Programming Interfaces).

Graphics Library functions are divided in three libraries. They are as follows-

i. GL Library (OpenGL in Windows)
ii. GLU (OpenGL Utility Library)
iii. GLUT (OpenGL Utility Toolkit)

Functions in main GL library name function names that begin with the letter 'gl'.

- GLU library uses only GL functions but contains code for creating objects and simplify viewing.
- To interface with the window system and to get input from external devices GLUT library is used, which is a combination of three libraries GLX for X windows, 'wgl' for Windows and 'agl' for Macintosh.
- These libraries are included in the application program using pre-processor directives. E.g.: #include<GL/glut.h>
- The following figure shows the library organization in OpenGL.

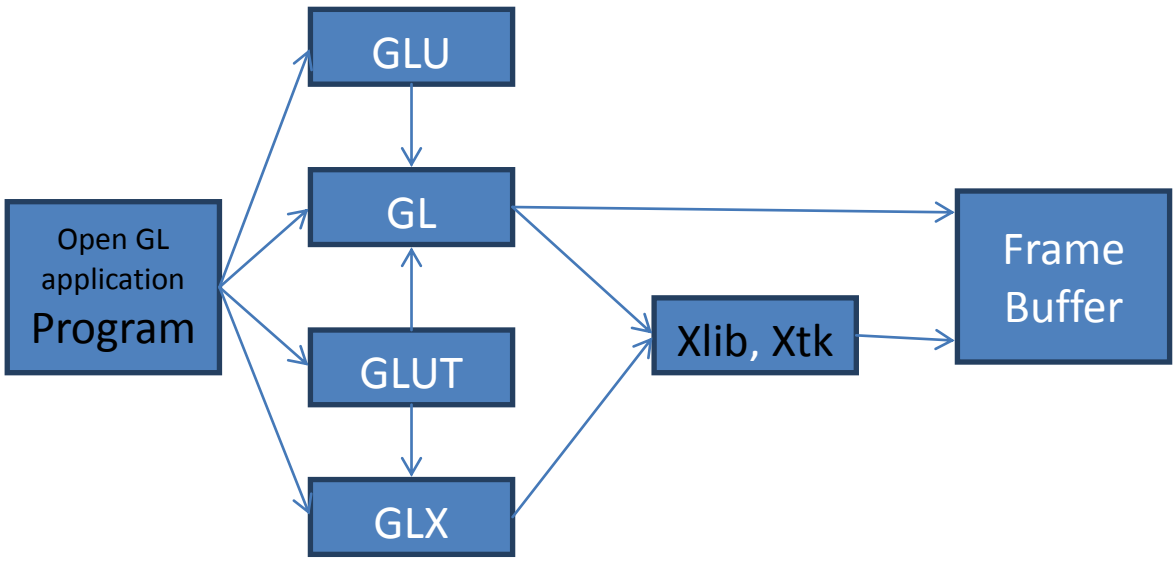

Fig 1.1: Library Organization

1.3 ABOUT PROJECT:

Computer Graphics is concerned with all aspects of producing pictures or images using a computer.

Here we use a particular graphics system, OpenGL which has become a widely accepted standard for developing graphics applications.

A 2D OpenGL animation that demonstrates the use of glPushMatrix and glPopMatrix to implement hierarchical modeling.

This project is titled **A SCENIC VIEW OF WINDMILL**.

It is coded such that Windmill keeps rotating, and rotation speed can be varied in three steps, car keeps moving from left to right and one car can be stopped. The day and night view can be experienced.

1.4 ORGANIZATION OF REST OF THE PROJECT

- In requirements specification, we give a brief description of the project and also the features included in the project.
- In system design phase, we give the overview of project and how the features mentioned in the requirements specification are implemented. We give the brief system architectural design and the description of components. We have also included the OPENGL function calls used.
- In source code phase, we specify the data structures used in this project along with data type definitions. We also give the detailed description of each component.
- In testing phase, we test each component and integrate those components and test the system as a whole. We provide the snapshots of the expected output.
- In the conclusion and scope for the further enhancements, we provide ways using which some more features can be implemented.
- In the reference phase, the books and the webistes reffered to accomplish the completion of this project are given.

CHAPTER 2

SYSTEM REQUIREMENTS

2.1 Hardware Requirements

- Hard disk : 40GB Hard disk or higher.
- Processor : Intel i3 core/AMD Athlon or higher.
- Memory : 1 GB or higher.
- Monitor : Mono/Colour
- Keyboard : Low profile, dispatchable type.
- Graphics Card : 1280x800 display with qualified hardware acceleration.

2.2 Software Requirements

- Microsoft Visual Studio 2008/10 package.
- OpenGL Interface.
- Windows OS XP/7/8/10.
- Glut library functions.

CHAPTER 3

DESIGN

3.1 Initialization

Initialize the interaction with the windows. Initialize the display mode- double buffer and depth buffer. Initialize the various callback functions for drawing and redrawing the polygon, for mouse and keyboard interface, for movement of the image in different directions. Initialize the window position and size and create the window to display the output.

3.2 Flow of control

The flow of control in the above flow chart is respected to the Texture Package. For any of the program flow chart is compulsory to understand the program. We consider the flow chart for the texture project in which the flow starts from start and proceeds to the main function after which it comes to the initialization of call back functions and further it proceeds to mouse and keyboard functions after all the function, the flow comes to quit which is the end of the flow chart.

FLOWCHART

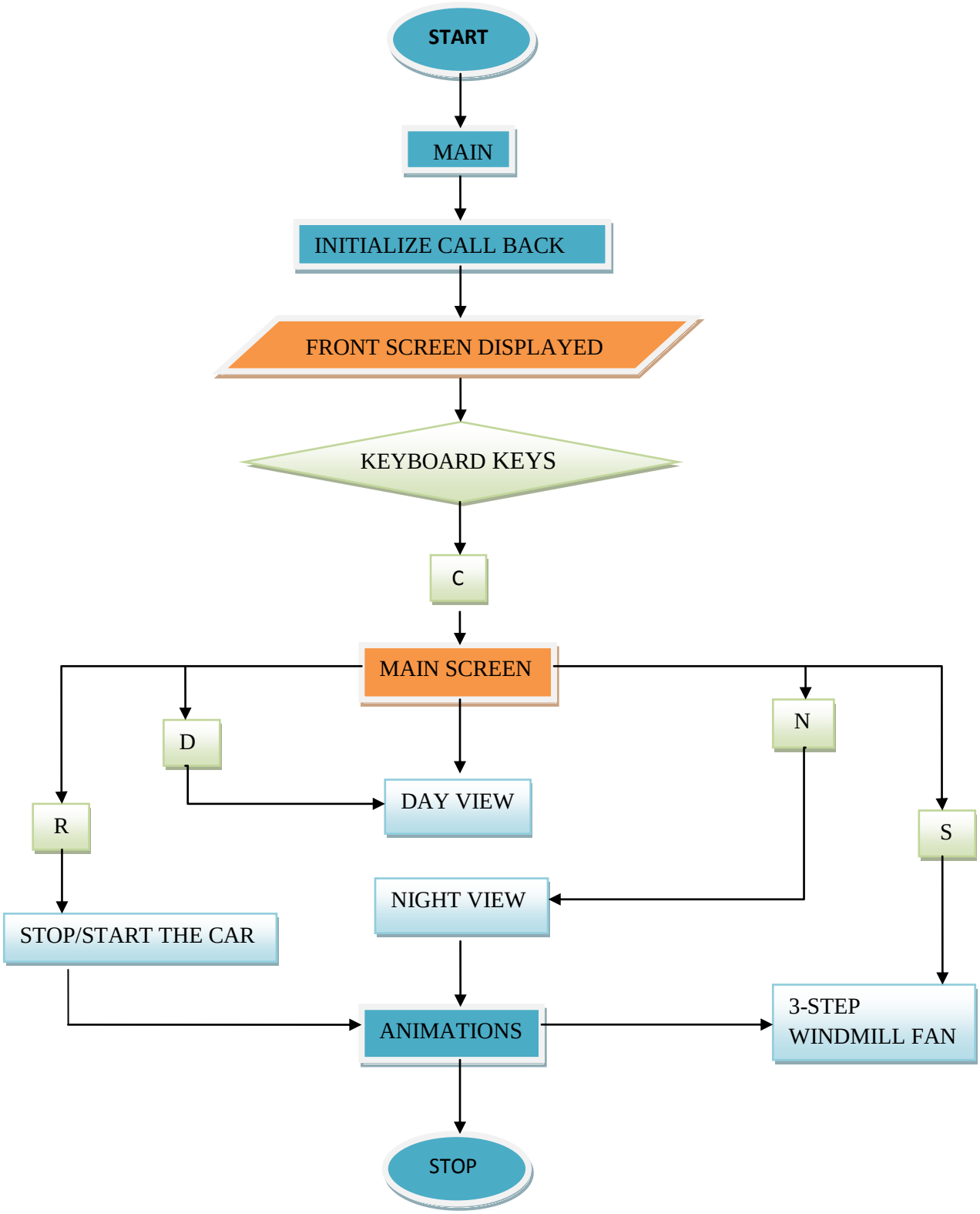

Figure 3.1 Flow Chart for scenic view of Windmill

CHAPTER 4

IMPLEMENTATION

4.1 Built-in Functions:

- **void glutInit(int *argc, char **argv);**

  Initialises GLUT. The arguments passed from main can be used by the applications

- **void glutInitDisplayMode(unsigned int mode);**

  Requests a display with properties in mode. The value of the mode is determined by logical OR operation. Mode values used are GLUT_DOUBLE, GLUT _RGB, GLUT_DEPTH.

- **void glutInitWindowSize(int width, int height);**

  Specifies the initial width and height of window in pixels.

- **int glutCreateWindow(const char *title);**

  Gives a name to the window which is created.

- **void glutMainLoop(void);**

  The glutMainLoop enters the GLUT event processing loop

- **void glutDisplayFunc(void (*func)(void));**

  The glutDisplayFunc sets the display callback for the current window specified by 'func'.

- **void glFlush(void);**

  Empties all of these buffers, causing all issued commands to be executed as quickly they are accepted by the actual rendering engine.

- **void glLoadIdentity(void);**

  Replaces the current matrix with the identity matrix.

- **void glRotatef(GLfloat angle, GLfloat x, GLfloat y, GLfloat z);**

  Produces a rotation of angle degrees around vector x,y,z.

4.2 User-defined Functions:

- **void myInit(void);**
  Responsible for the initial initialisation of point attributes, clearing the screen etc.

- **void Display(void);**
  This function is called when the image needs to be redrawn.
  It is installed by main() as the GLUT display function.
  It draws the current frame of the animation.

- **void main(int argc, char**argv);**
  Main function from which execution begins and which calls all other functions.

- **void keys(unsigned char key, int x, int y);**
  Keyboard function where few keys are given

- **void Write(double x,double y,double z,double scale,char *s);**

- **void circle1(GLfloat x, GLfloat y, GLfloat radius);**
  Using Circle1 function draw the circle with some specified radius.

- **void drawWheel();**
  Draw a wheel, centered at (0,0) and with radius 1. The wheel has 15 spokes that rotate in a clockwise direction as the animation proceeds.

- **void drawCart();**
  Draw a cart of required by using GL_POLYGONS. And insert its wheels using push and pop matrix functions.

- **void drawSun();**
  Draw a sun with radius 0.5 centered at (0,0). There are also 13 rays which Extend outside from the sun for another 0.25 units.

- **void drawWindmill();**

  Draw a windmill, consisting of a pole and three vanes. The pole extends from the point (0,0) to (0,3). The vanes radiate out from (0,3). A rotation that depends on the frame number is applied to the whole set of vanes, which causes the windmill to rotate as the animation proceeds.

4.3 Source Code:

```
#include<stdio.h>
#include <GL/glut.h>
#include <math.h>
#include<string.h>
const double PI = 3.141592654;
int night=0,stop=0,day=0;
int speed=0;
int frameNumber = 0;
int frameNumberOfCart=0;
int win1,win2;
int storeframe=0;
void circle1(GLfloat x,GLfloat y,GLfloat radius);
void display();
void keys(unsigned char key, int x, int y)
{
        if(key=='n') {
        night=1-night;}
        if(key=='d')
                day=day-1;
        if(key=='s')
                speed=(1+speed)%3;
        if(key=='r'){
        stop=1-stop;}

/*
 * Draw a cart consisting of a rectangular body and two wheels.  The wheels
 * are drawn by the drawWheel() method; a different translation is applied to each
 * wheel to move them into position under the body.  The body of the cart
 * is a red rectangle with corner at (0,-2.5), width 5, and height 2.  The
 * center of the bottom of the rectangle is at (0,0).
 */
}

void Write(double x,double y,double z,double scale,char *s)
{
int i,l=strlen(s);
glPushMatrix();
glTranslatef(x,y,z);
glScalef(scale,scale,scale);
for(i=0;i<l;i++)
glutStrokeCharacter(GLUT_STROKE_ROMAN,s[i]);
```

```
glPopMatrix();
}

void redisplay(void)
{
glClearColor(1.0/256.0,40.0/256.0,180.0/256.0,0.0);
glClear(GL_COLOR_BUFFER_BIT);
glColor3f(1.0,1.0,0.0);
Write(-0.95,0.9,1,0.0007,"Sir M Visvesvaraya Institue of Technology");
Write(-0.4,0.8,1,0.0007,"Dept. of CSE");
glColor3f(1.0,0.0,0.0);
Write(-0.75,0.4,1,0.0007,"  Graphical Implementation of ");
Write(-0.75,0.3,1,0.0007,"OpenGL Hierarchical Modeling 2D");

Write(-0.4,0.6,0.0,0.0007,"Mini Project on");
glColor3f(1.0,1.0,1.0);
Write(-0.4,-0.8,0.0,0.0006,"Press 'C' to continue");
glColor3f(0.5,0.5,0.0);
Write(-1.0,0.1,0.0,0.0007," Submitted BY:");
glColor3f(0.0,0.0,0.0);
Write(-1.0,-0.03,0.0,0.0006," Harish Devathraj: 1MV13CS038 ");
Write(-1.0,-0.2,0.0,0.0006," Gururaj: 1MV13CS037 ");
Write(-1.0,-0.4,0.0,0.0007," Under the guidance of: ");
Write(0.15,-0.415,0.0,0.0006," ELAIYARAJA P");
glFlush();
}


/*
 * This function is set as the glutTimerFunc to drive the animation
 */
void doFrame(int v) {
   frameNumber++;
         frameNumberOfCart++;
   glutPostRedisplay();
   glutTimerFunc(30,doFrame,0);
}

/*
 * This method is called from main() to initialize OpenGL.
 */
void init() {
         glClearColor(0.5f, 0.5f, 1, 1);
```

```
                // The next three lines set up the coordinates system.
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        glOrtho(0, 7, -1, 4, -1, 1);
        glMatrixMode(GL_MODELVIEW);
}


void keyboard1(unsigned char key ,int x ,int y)
{
if(key=='c' || key=='C')
{
        glutDestroyWindow(win1);
         glutInitDisplayMode(GLUT_DOUBLE);
        win2=glutCreateWindow("Windmill");
        init();
        glutDisplayFunc(display);
         glutTimerFunc(200,doFrame,0);
        glutKeyboardFunc(keys);

}
}
void circle1(GLfloat x, GLfloat y, GLfloat radius)
 {
   float angle;
   glBegin(GL_POLYGON);
   for(int i=0;i<100;i++)
        {
      angle = i*2*(PI/100);
      glVertex2f(x+(cos(angle)*radius),y+(sin(angle)*radius));
   }
   glEnd();
 }

void drawDisk(double radius) {
        int d;
        glBegin(GL_POLYGON);
        for (d = 0; d < 32; d++) {
                double angle = 2*PI/32 * d;
                glVertex2d( radius*cos(angle), radius*sin(angle));
        }
        glEnd();
}
```

```
/*
 * Draw a wheel, centered at (0,0) and with radius 1. The wheel has 15 spokes
 * that rotate in a clockwise direction as the animation proceeds.
 */
void drawWheel() {
   int i;
          glColor3f(0,0,0);
          drawDisk(1);
          glColor3f(0.75f, 0.75f, 0.75f);
          drawDisk(0.8);
          glColor3f(0,0,0);
          drawDisk(0.2);
          glRotatef(frameNumber*20,0,0,1);
          glBegin(GL_LINES);
          for (i = 0; i < 15; i++) {
                   glVertex2f(0,0);
                   glVertex2d(cos(i*2*PI/15), sin(i*2*PI/15));
          }
          glEnd();
}

void drawCart() {
          glPushMatrix();
          glTranslatef(-1.5f, -0.1f, 0);
          glScalef(0.8f,0.8f,1);
          drawWheel();
          glPopMatrix();
          glPushMatrix();
          glTranslatef(1.5f, -0.1f, 0);
          glScalef(0.8f,0.8f,1);
          drawWheel();
          glPopMatrix();
          glColor3f(.1,.25,0);


          glBegin(GL_POLYGON);              // start drawing a polygon
  glVertex3f(-3.0f, 3.0f, 0.0f);       // Top left
  glVertex3f(1.2f, 3.0f, 0.0f);
   glColor3f(1.0f,1.5f,1.6f);
  glVertex3f(3.0f, 1.2f, 0.0f);

           // Set The Color To Green
```

```
  glVertex3f( 3.0f,0.0f, 0.0f);       // Bottom Right
  glVertex3f(-3.0f,0.0f, 0.0f);


        glEnd();
}
void drawCart1() {
        glPushMatrix();
        glTranslatef(-1.5f, -0.1f, 0);
        glScalef(0.8f,0.8f,1);
        drawWheel();
        glPopMatrix();
        glPushMatrix();
        glTranslatef(1.5f, -0.1f, 0);
        glScalef(0.8f,0.8f,1);
        drawWheel();
        glPopMatrix();
        glColor3f(0.3,0.2,0);


        glBegin(GL_POLYGON);            // start drawing a polygon
        glVertex2f(-2.5f,0);
        glVertex2f(2.5f,0);
        glColor3f(1,0.2,0);
        glVertex2f(2.5f,2);
        glVertex2f(-2.5f,2);
        glEnd();
}

/*
 * Draw a sun with radius 0.5 centered at (0,0).  There are also 13 rays which
 * extend outside from the sun for another 0.25 units.
 */
void drawSun() {
        int i;

                glColor3f(1,1,0);
                for (i = 0; i < 13; i++) { // Draw 13 rays, with different rotations.
                        glRotatef( 360 / 13, 0, 0, 1 ); // Note that the rotations accumulate!
                        glBegin(GL_LINES);
                        glVertex2f(0, 0);
                        glVertex2f(0.75f, 0);
                        glEnd();
```

```
                }

        drawDisk(0.5);
}

void drawSun1() {
        if(night==1){
                glColor3f(1.0,1.0,1.0);
                                drawDisk(0.35);}

}
void drawSun2() {

        if(night==1){

                  glColor3f(0.0,0.0,0.0);
                  drawDisk(0.35);}

}
/*
 * Draw a windmill, consisting of a pole and three vanes.  The pole extends from the
 * point (0,0) to (0,3).  The vanes radiate out from (0,3).  A rotation that depends
 * on the frame number is applied to the whole set of vanes, which causes the windmill
 * to rotate as the animation proceeds.

void drawWindmill() {
        int i;
        glColor3f(0.8f, 0.8f, 0.9f);
        glBegin(GL_POLYGON);
        glVertex2f(-0.05f, 0);    //thick ness of the windmill pole
        glVertex2f(0.05f, 0);
        glVertex2f(0.05f, 3);
        glVertex2f(-0.05f, 3);
        glEnd();
        glTranslatef(0, 3, 0);
        glRotated(frameNumber * (180.0/(10*(5*speed+4))), 0, 0, 1);
        glColor3f(0.4f, 0.4f, 0.8f);
        for (i = 0; i < 3; i++) {
                glRotated(120, 0, 0, 3);  // Note: These rotations accumulate.
                glBegin(GL_POLYGON);
                glVertex2f(0,0);
                glVertex2f(0.5f, 0.1f);
                glVertex2f(1.5f,0);
```

```
                glVertex2f(0.5f, -0.1f);
                glEnd();
        }
}

/*
 * This function is called when the image needs to be redrawn.
 * It is installed by main() as the GLUT display function.
 * It draws the current frame of the animation.
 */
void display() {


        glClear(GL_COLOR_BUFFER_BIT); // Fills the scene with blue.
        glLoadIdentity();
        if(night==1){
                glClearColor(0, 0, 0, 1);

                glColor3f(1.0,1.0,1.0);
   circle1(1.87,2.85,0.01);
        circle1(1.9,2.65,0.01);
        circle1(1.56,3.5,0.02);

        circle1(0.85,3.5,0.01);
        circle1(1.345,2.5,0.01);
        circle1(2.02,2.69,0.01);
        circle1(1.8,3.75,0.02);
        circle1(0.99,3.45,0.01);
        circle1(1.345,2.5,0.01);
        circle1(7.40,2.8,0.01);
        circle1(6.120,2.5,0.01);
        circle1(4.90,7.5,0.01);
        circle1(8.0,3.5,0.01);
        circle1(6.40,3.5,0.01);
        circle1(4.55,3.5,0.01);
        circle1(7.65,3.5,0.01);

        circle1(8.23,3.1,0.01);
        circle1(6.2,2.5,0.01);
        circle1(4.95,2.85,0.01);
        circle1(6.15,2.75,0.01);
        }
        else
```

```
        glClearColor(0.4,0.7,1.0,1);
if(night==1)
/* Draw three green triangles to form a ridge of hills in the background */
glColor3f(0,0.2f,0.0f);
else
        glColor3f(0, 0.6f, 0.2f);
glBegin(GL_POLYGON);
glVertex2f(-3,-1);
glVertex2f(1.5f,1.65f);
glVertex2f(5,-1);
glEnd();
glBegin(GL_POLYGON);
glVertex2f(-3,-1);
glVertex2f(3,2.1f);
glVertex2f(7,-1);
glEnd();
glBegin(GL_POLYGON);
glVertex2f(0,-1);
glVertex2f(6,1.2f);
glVertex2f(20,-1);
glEnd();


/* Draw a bluish-gray rectangle to represent the road. */
if(night==0)
glColor3f(0.4f, 0.4f, 0.5f);
else
        glColor3f(0.3f,0.3f,0.4f);
glBegin(GL_POLYGON);
glVertex2f(0,-0.4f);
glVertex2f(7,-0.4f);
glVertex2f(7,0.4f);
glVertex2f(0,0.4f);
glEnd();

/* Draw a white line to represent the stripe down the middle
 * of the road. */

glLineWidth(4);
// Set the line width to be 6 pixels.
if(night==1) glColor3f(0.9,0.9,0.9);
else
        glColor3f(1,1,1);
```

```
glBegin(GL_LINES);
glVertex2f(0,0);
glVertex2f(0.5,0);
glEnd();

glBegin(GL_LINES);
glVertex2f(0.75,0);
glVertex2f(1.25,0);
glEnd();

glBegin(GL_LINES);
glVertex2f(1.5,0);
glVertex2f(2.0,0);
glEnd();

glBegin(GL_LINES);
glVertex2f(2.25,0);
glVertex2f(2.75,0);
glEnd();

glBegin(GL_LINES);
glVertex2f(3.0,0);
glVertex2f(3.5,0);
glEnd();

glBegin(GL_LINES);
glVertex2f(3.75,0);
glVertex2f(4.25,0);
glEnd();

glBegin(GL_LINES);
glVertex2f(4.5,0);
glVertex2f(5.0,0);
glEnd();

glBegin(GL_LINES);
glVertex2f(5.25,0);
glVertex2f(5.75,0);
glEnd();

glBegin(GL_LINES);
glVertex2f(6.0,0);
glVertex2f(6.5,0);
```

```
        glEnd();

        glBegin(GL_LINES);
        glVertex2f(6.75,0);
        glVertex2f(7.25,0);
        glEnd();
        glBegin(GL_LINES);
        glVertex2f(7.5,0);
        glVertex2f(8.0,0);
        glEnd();

        glColor3f(1,1,1);
        glBegin(GL_LINES);
        glVertex2f(0,0.33);
        glVertex2f(7,0.33);
        glEnd();

        glColor3f(1,1,1);
        glBegin(GL_LINES);
        glVertex2f(0,-0.33);
        glVertex2f(7,0-0.33);
        glEnd();
        glLineWidth(1);  // Reset the line width to be 1 pixel.

        /* Draw the sun.  The drawSun method draws the sun centered at (0,0).  A 2D
translation
         * is applied to move the center of the sun to (5,3.3).   A rotation makes it rotate*/

        glPushMatrix();
        if(night==0)
                glTranslated(5.8,3,0);
        else
                glTranslated(100.8,300,0);
        glRotated(-frameNumber*0.7,0,0,1);
        drawSun();
        glPopMatrix();

        glPushMatrix();
                glTranslated(2.8,3.5,0);
                drawSun1();
                glPopMatrix();

                glPushMatrix();
```

```
                glTranslated(2.75,3.68,0);
                drawSun2();
                glPopMatrix();

                if(night==0)
                        glColor3f(0.5,0.5,0.5);
                else
                        glColor3f(0.1,0.1,0.1);

        circle1(1.6,3.13,.200);
        circle1(1.75,3.23,.300);
        circle1(1.1,3.28,.400);
        circle1(1.5,3.33,.300);

        /* Draw three windmills.
        glPushMatrix();
        glTranslated(0.75,1,0);
        glScaled(0.6,0.6,1);
        drawWindmill();
        glPopMatrix();

        glPushMatrix();
        glTranslated(2.2,1.6,0);
        glScaled(0.4,0.4,1);
        drawWindmill();
        glPopMatrix();

        glPushMatrix();
        glTranslated(5.7,0.8,0);
        glScaled(0.7,0.7,1);
        drawWindmill();
        glPopMatrix();

        /* Draw the cart.  The drawCart method draws the cart with the center of its base at
         * (0,0).  The body of the cart is 5 units long and 2 units high.  A scale is first
         * applied to the cart to make its size more reasonable for the picture.  Then a
         * translation is applied to move the cart horizontally.  The amount of the
translation
         * depends on the frame number, which makes the cart move from left to right
across
         * the screen as the animation progresses.  The cart animation repeats every 300
         * frames.  At the beginning of the animation, the cart is off the left edge of the
         * screen. */
```

```
        glPushMatrix();
        if(stop==0){
                glTranslated(-3 + 13*(frameNumberOfCart % 500) / 300.0, 0, 0);
                storeframe=frameNumberOfCart;
                //printf("%d\n",frameNumberOfCart);
        }
        else{
                glTranslated(-3 + 13*(storeframe % 500) / 300.0, 0, 0);
                frameNumberOfCart=storeframe;
        }
        glScaled(0.22,0.22,1);
        drawCart();
        glPopMatrix();

        glPushMatrix();
        glTranslated(-3 + 13*(frameNumber % 300) / 200.0, 0, 0);
        glScaled(0.3,0.3,1);
        drawCart1();
        glPopMatrix();
        glFlush();
        glutSwapBuffers();
}  // end display

void main(int argc, char** argv) {
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);

        glutInitWindowSize(700,500);
        glutInitWindowPosition(100,100);
        win1=glutCreateWindow("INTRODUCTION");
        glutDisplayFunc(redisplay);
        glutKeyboardFunc(keyboard1);
        glutMainLoop();
}
```

CHAPTER 5

SYSTEM SNAPSHOTS

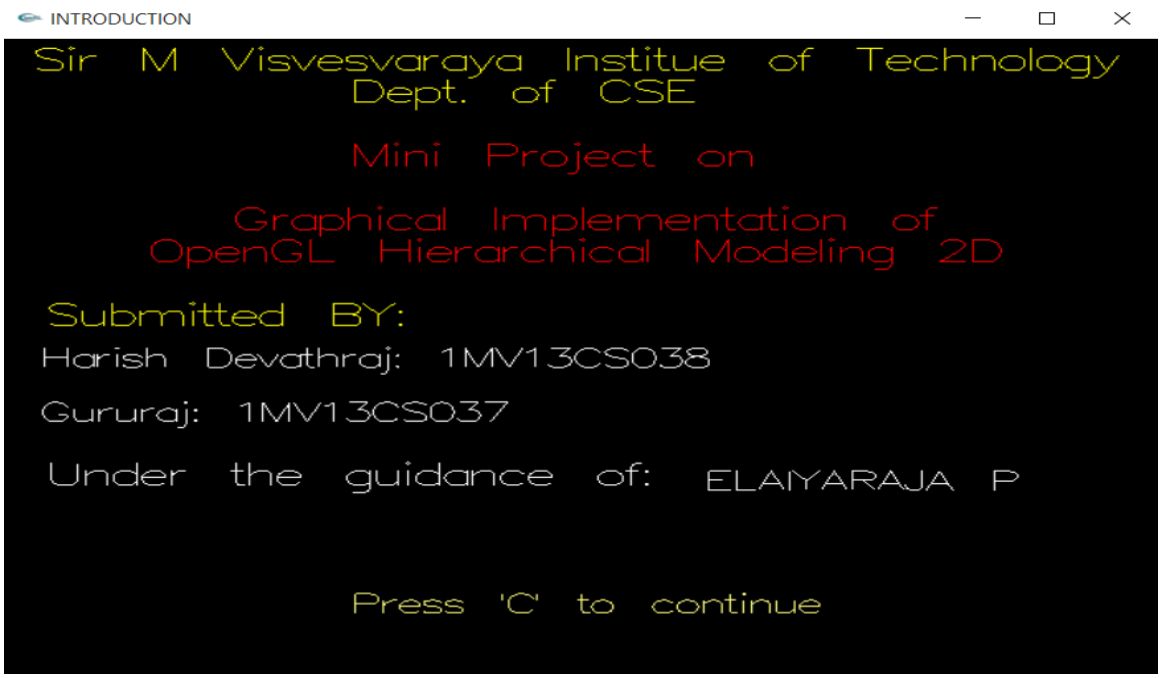

Figure 5.1: Front Screen

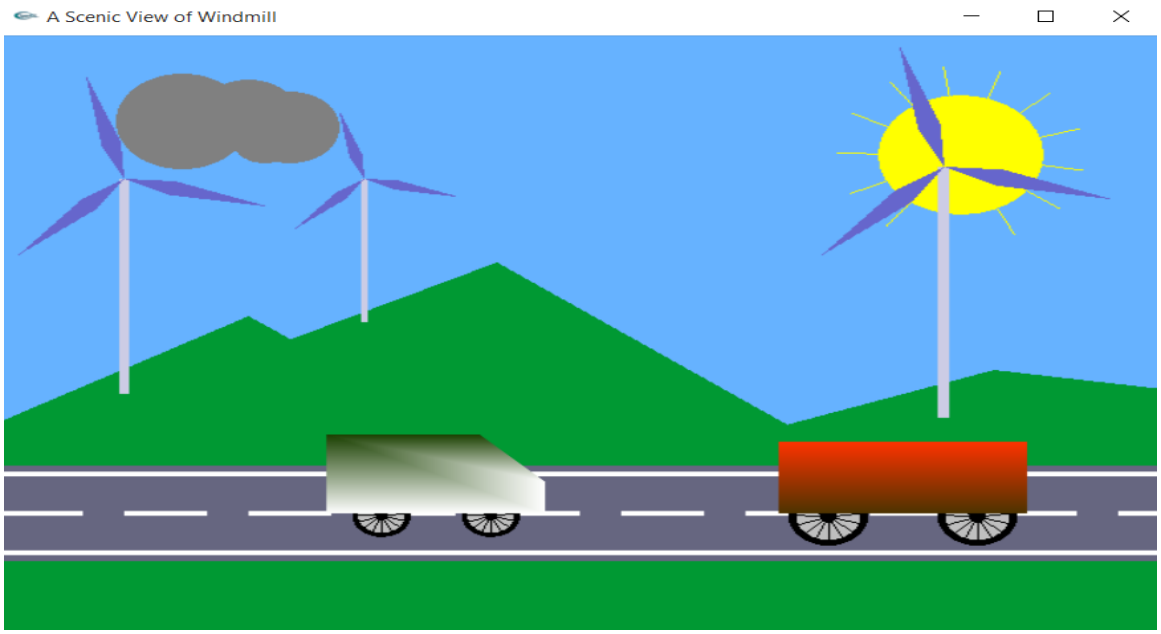

Figure 5.2: Day View

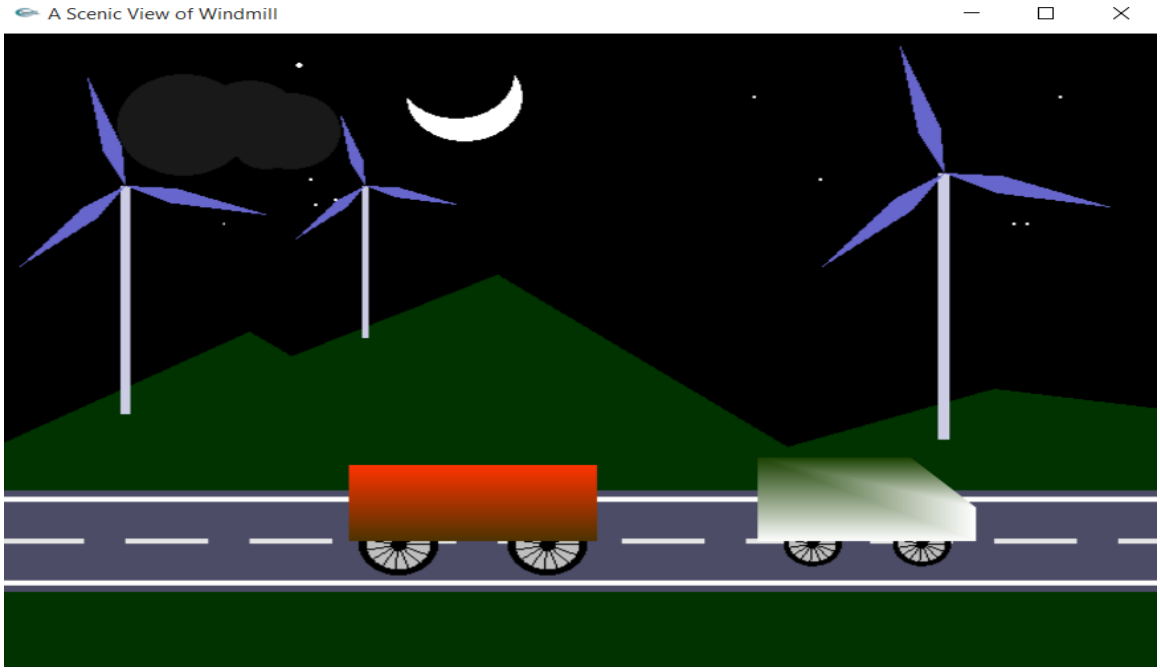

Figure 5.3: Night View

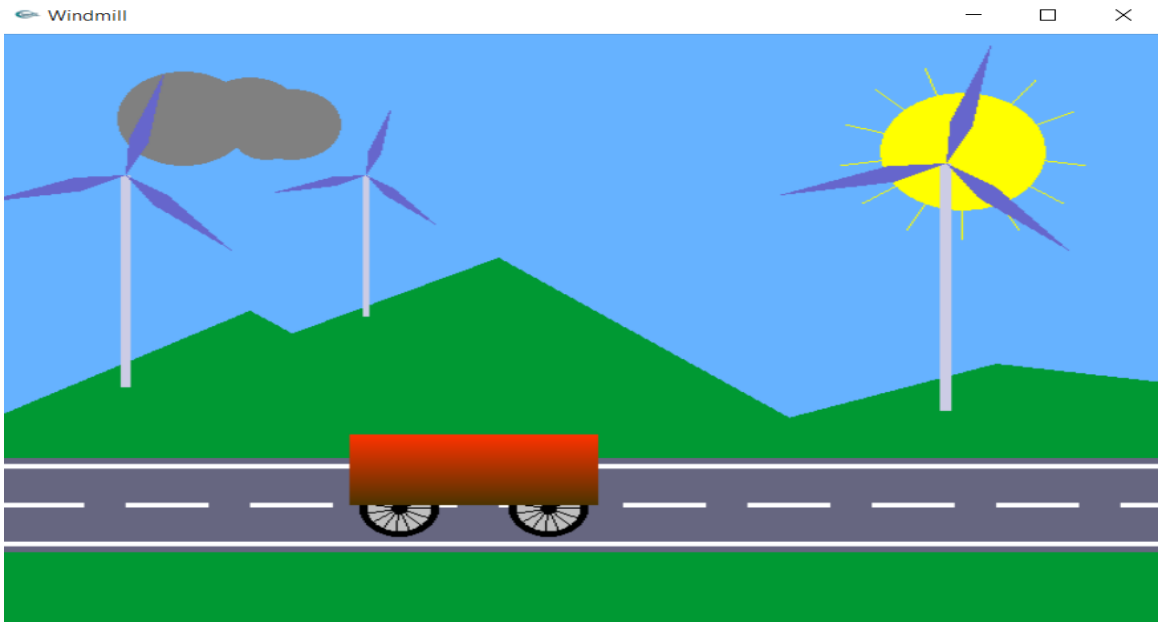

Figure 5.4: Car has Stopped

CHAPTER 6

CONCLUSION

An attempt has been made to develop an OpenGL graphics package, which meets the necessary requirements of the user successfully. It enables us to learn about the basic concepts in OpenGL graphics and know standard library graphics functions and also to explore some other functions. OpenGL graphics is a huge library which consist of numerous functions. These function have been used creatively to build the programs which may be to draw figures of various shapes at lower level or to stimulate any real thing, animation etc at higher level.

This project has given us an insight into the use of computer graphics. As we have had to use many built-in and user defined functions, we have managed to get a certain degree of familiarity with these functions and have generally understood the power of these functions and were able to comprehend the true nature of the most powerful tool graphics in OpenGL and also have understood to a reasonable extent the reason why Graphics is so powerful for game programmers. We can now converse with a certain degree of confidence about Graphics in OpenGL and finally we have successfully completed the implementation of this project using OpenGL.

SCOPE FOR FUTURE ENHANCEMENTS

This project has been designed using C++, which works on the windows platform. The project can be designed using other languages and better graphical interfaces. The following features could have been incorporated.

- This 2D design can be converted in 3D for more better look.
- Various lighting elements can be implemented to make it more realistic.
- Design of Car, Mountain etc can be more realistic by using shadow techniques.

BIBLIOGRAPHY

BOOKS

The books that helped us in implementing this project are as follows:

- **Interactive Computer Graphics**- A top down approach with OpenGL
  -Edward angel,5th edition,Addison-WESLEY-2009
- **Computer Graphics Using OpenGL**
  -F.S.Hill,jr.2nd edition, Pearson Education,2001
- **Computer Graphics-OpenGL version**
  -Donald Hearn and Pauline Baker,2nd edition, Pearson Education,2003

REFERENCES

The list of references are as follows:

- www.opengl.org
- www.wikipedia.org
- www.computergraphics.com