VISVESVARAYA TECHNOLOGICAL UNIVERSITY
"Jnana Sangama", Belagavi-590018, Karnataka

BANGALORE INSTITUTE OF TECHNOLOGY
K. R. Road, V. V. Puram, Bengaluru-560 004

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Computer Graphics Laboratory With Mini Project Report-18CSL67
on

"Wind Energy Simulation"

Submitted By

1BI19CS146	**Shriya Khar**
1BI19CS193	**Zainab Drabu**

for the academic year 2022-23

Under the guidance of

Prof. DR Nagamani
Assistant Professor

VISVESVARAYA TECHNOLOGICAL UNIVERSITY
"Jnana Sangama", Belagavi-590018, Karnataka

BANGALORE INSTITUTE OF TECHNOLOGY
K. R. Road, V. V. Puram, Bengaluru-560 004

Department of Computer Science & Engineering

Certificate

This is to certify that the implementation of **Computer Graphics Laboratory With Mini Project (18CSL67)** entitled **"WIND ENERGY SIMULATION"** has been successfully completed by

1BI19CS146 **Shriya Khar**

of VI semester B.E. for the partial fulfillment of the requirements for the Bachelor's degree in **Computer Science & Engineering** of the **Visvesvaraya Technological University** during the academic year **2022-2023**.

Lab In charge:

Prof. DR Nagamani
Assistant Professor
Dept. of CSE, BIT

Dr. Girija J.
Professor and Head
Dept. of CSE, BIT

Examiners: 1) 2)

ACKNOWLEDGEMENT

The knowledge & satisfaction that accompany the successful completion of any task would be incomplete without mention of people who made it possible, whose guidance and encouragement crowned my effort with success. I would like to thank all and acknowledge the help I have received to carry out this Mini Project.

I would like to convey my sincere thanks to **Dr. Girija J.**, HOD, Department of CS&E, BIT for being kind enough to provide the necessary support to carry out the mini project.

I am most humbled to mention the enthusiastic influence provided by the lab in-charges **Prof. DR Nagamani** and **Prof. Suma L.,** on the project for their ideas, time to time suggestions for being a constant guide and co-operation showed during the venture and making this project a great success.

I would also take this opportunity to thank my friends and family for their constant support and help. I'm very much pleasured to express my sincere gratitude to the friendly co-operation showed by all the **staff members** of Computer Science Department, BIT.

Shriya Khar
1BI19CS146

Table of contents

Chapter -1

INTRODUCTION

1.1 Computer Graphics

Computer graphics is an art of drawing pictures, lines, charts, using computers with the help of programming. Computer graphics is made up of number of pixels. Pixel is the smallest graphical picture or unit represented on the computer screen. Basically, there are 2 types of computer graphics namely,

Interactive Computer Graphics involves a two-way communication between computer and user. The observer is given some control over the image by providing him with an input device. This helps him to signal his request to the computer.

Non-Interactive Computer Graphics otherwise known as passive computer graphics it is the computer graphics in which user does not have any kind of control over the image. Image is merely the product of static stored program and will work according to the instructions given in the program linearly. The image is totally under the control of program instructions not under the user. Example: screen savers.

1.2 Applications of Computer Graphics

Scientific Visualization

Scientific visualization is a branch of science, concerned with the visualization of three-dimensional phenomena, such as architectural, meteorological, medical, biological systems.

Graphic Design

The term graphic design can refer to a number of artistic and professional disciplines which focus on visual communication and presentation

Computer-aided Design

Computer-aided design (CAD) is the use of computer technology for the design of objects, real or virtual. The design of geometric models for object shapes, in particular, is often called computer-aided geometric design (CAGD). The manufacturing process is tied in to the computer description of the designed objects so that the fabrication of a product can be automated using methods that are referred to as CAM, computer-aided manufacturing.

Web Design

Web design is the skill of designing presentations of content usually hypertext or hypermedia that is delivered to an end-user through the World Wide Web, by way of a Web browser.

Digital Art

Digital art most commonly refers to art created on a computer in digital form.

Video Games

A video game is an electronic game that involves interaction with a user interface to generate visual feedback on a raster display device.

Virtual Reality

Virtual reality (VR) is a technology which allows a user to interact with a computer simulated environment. The simulated environment can be similar to the real world. This allows the designer to explore various positions of an object. Animations in virtual reality environments are used to train heavy equipment operators or to analyse the effectiveness of various cabin configurations and control placements.

Computer Simulation

A computer simulation, a computer model or a computational model is a computer program, or network of computers, that attempts to simulate an abstract model of a particular system.

Education and Training

Computer simulations have become a useful part of mathematical modelling of many natural systems in physics, chemistry and biology, human systems in economics, psychology, and social science and in the process of engineering new technology, to gain insight into the operation of those systems, or to observe their behaviour. Most simulators provide screens for visual display of the external environment with multiple panels is mounted in front of the simulator.

Image Processing

The modification or interpretation of existing pictures such as photographs and TV scans, is called image processing. In computer graphics, a computer is used to create a picture. Image processing techniques, on the other hand, are used to improve picture quality, analyse images, or recognize visual patterns for robotics applications

1.3 OpenGL

OpenGL has become a widely accepted standard for developing graphics applications. Most of our applications will be designed to access OpenGL directly through functions in the three libraries. Functions in main GL libraries 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 viewing. All function in GLU can be created from the core GL library. The GLU library is available in all OpenGL implementations. Functions in the GLU library starts with the letters glu.

The third is the OpenGL Utility Toolkit (GLUT). It provides the minimum functionality that should be formulated in modern windowing systems.

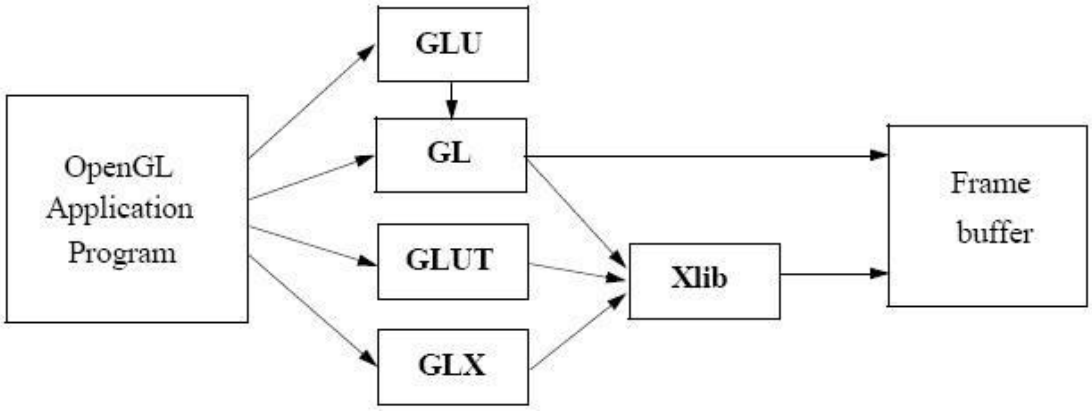

Figure 1.1 Basic block diagram of OpenGL

1.4 Problem Statement

"To implement the simulation of WIND ENERGY using OpenGL framework,graphic primitives, textures and keyboard/mouse interactions."

1.5 Objectives of the Project

- To show the working of the orthographic projections in appearance of the objects used in the scene.
- To show the implementation of Textures for a better appearance of the real-world objects.
- To show the implementation of the OpenGL transformation functions.
- To show the user and programme interaction using input devices.

1.6 Organisation of the Project

The project was organised in a systematic way. First we analysed what are the basic features to be included in the project to make it acceptable. As it is a graphics oriented project, we made the sketches prior, so as to have an idea like how our output must look like. After all these, the source code was formulated as a paper work. All the required software were downloaded. Finally, the successful implementation of the project.

Chapter -2

SYSTEM SPECIFICATION

2.1 Hardware Requirements

- Main Processor : PENTIUM III
- Processor Speed: 800 MHz
- RAM Size : 128 MB DDR
- Keyboard : Standard qwerty serial or PS/2 keyboard
- Mouse : Standard serial or PS/2 mouse
- Compatibility : AT/T Compatible
- Cache memory : 256 KB
- Diskette drive : 1,44MB,3.5 inches

2.2 Software Requirements

- Operating System: Windows 10 or Linux (Fedora) or macOS
- Hypervisor used : Docker
- Compiler used : g++
- Language used : C++ language
- Editor : Fedora
- Toolkit : GLUT Toolkit

Chapter -3

DESIGN

3.1 Flow Diagram

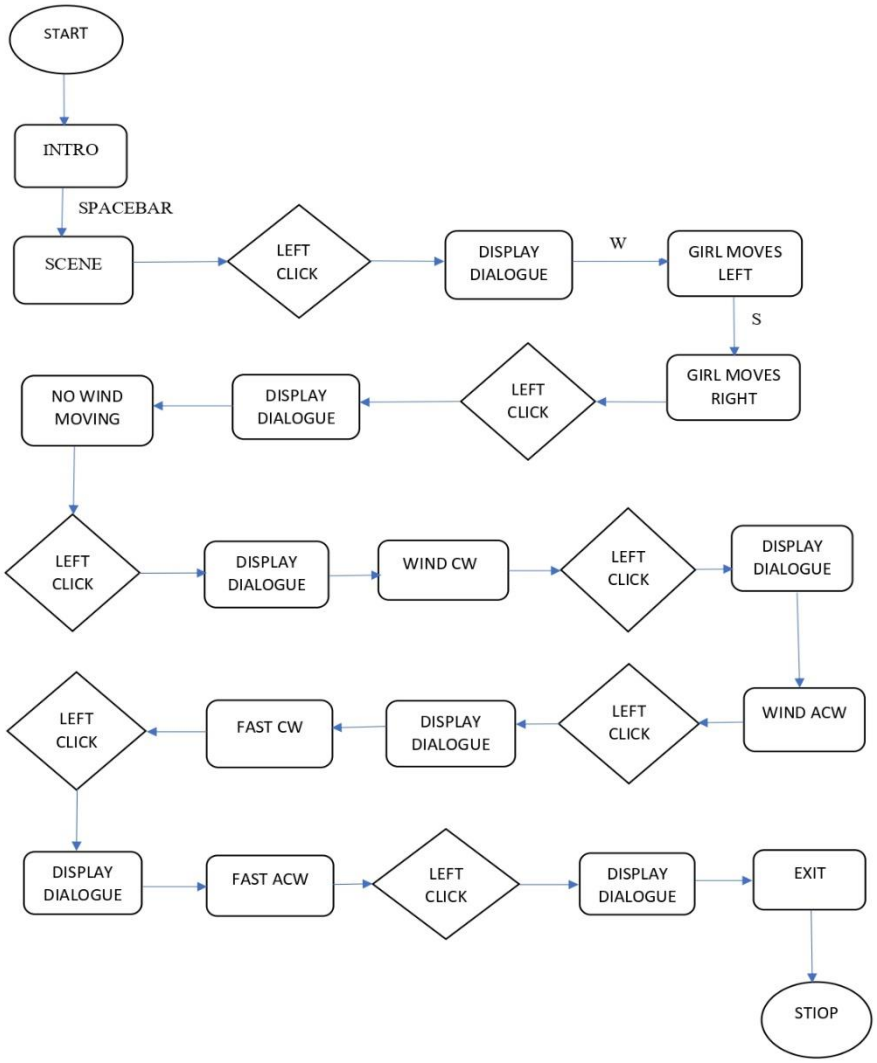

Figure 3.1 : Flow diagram

3.2 Description of Flow Diagram

The description of the flow diagram is as follows:

Step 1: Start.

Step 2: The user is presented with intro and presses ‘Spacebar’ to proceed to next scene.

Step 3: The user presses left click to show dialogues interactively.

Step 4: The user is prompted to press ‘W’ to move the girl in left direction.

Step 5: The user is prompted to press ‘S’ to move the girl in right direction.

Step 6: The user is prompted to press left click to display dialogues interactively.

Step 7: There will be no wind flowing at this moment in the scene.

Step 8: The user is prompted to press left click to display dialogues.

Step 9: Now the wind will be moving slowly in clockwise direction in the scene and energy will be generated in the form of lights in the house and power station.

Step 10: The user is prompted to press left click to display dialogues.

Step 11: Now the wind will be moving slowly in anti-clockwise direction in the scene and energy will be generated in the form of lights in the house and power station.

Step 12: The user is prompted to press left click to display dialogues.

Step 13 Now the wind will be moving fastly in clockwise direction in the scene and energy will be generated not only in the form of lights in the house and power station but as a street light too.

Step 14: The user is prompted to press left click to display dialogues.

Step 15: Now the wind will be moving fastly in anti-clockwise direction in the scene and energy will be generated not only in the form of lights in the house and power station but as a street light too.

Step 16: The user is prompted to press left click to display dialogues.

Step 17: The user must click Exit to leave the scene.

Step 18: Stop.

Chapter -4

IMPLEMENTATION

4.1 Built in Functions

1. glutInit()

glutInit is used to initialize the GLUT library.
Usage: void glutInit (int *argc, char **argv);
Description: glutInit will initialize the GLUT library and negotiate a session with the window system.

2. glutInitDisplayMode()

glutInitDisplayMode sets the initial display mode.
Usage: void glutInitDisplayMode (unsigned int mode);
Mode-Display mode, normally the bitwise OR-ing GLUT display mode bit masks.
Description: The initial display mode is used when creating top-level windows, sub-windows, and overlays to determine the OpenGL display mode for the to-be created window or overlay.

3. glutCreateWindow()

glutCreateWindow creates a top-level window.

Usage: intglutCreateWindow (char *name); Name-ASCII character string for use as window name

Description: glutCreateWindow creates a top-level window. The name will be provided to the window system as the window's name. The intent is that the window system will label the window with the name. Implicitly, the current window is set to the newly created window.

Each created window has a unique associated OpenGL context.

4. glutDisplayFunc()

glutDisplayFunc sets the display callback for the current window.

Description: glutDisplayFunc sets the display callback for the current window. When GLUT determines that the normal plane for the window needs to be redisplayed, the display callback for the window is called. Before the callback, the current window is set to the window needing to be redisplayed and the layer in use is set to the normal plane. The display callback is called with no parameters. The entire normal plane region should be redisplayed in response to the callback.

5. glutMainLoop()

glutMainLoop enters the GLUT event processing loop.

Usage: void glutMainLoop(void);

Description: 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 return. It will call as necessary any callbacks that have been registered.

6. glMatrixMode()

The two most important matrices are the model-view and projection matrix. At many times, the state includes values for both of these matrices, which are initially set to identity matrices. There is only a single set of functions that can be applied to any type of matrix. Select the matrix to which the operations apply by first set in the matrix mode, a variable that is set to one type of matrix and is also part of the state.

7. glTranslate(GLfloat X, GLfloat Y, GLfloat Z)

glTranslate produces a translation by x y z. If the matrix mode is either GL_MODEL_VIEW or GL_PROJECTION, all objects drawn after a call to glTranslate are translated.

8. glRotatef(GLdouble angle, GLdouble X, GLdouble Y, GLdouble Z)

glRotatef produces a rotation of angle degrees around the vector x y z. If the matrix mode is either GL_MODEL_VIEW or GL_PROJECTION, all objects drawn after glRotatef is called are rotated. Use glPushMatrix() and glPopmatrix() to save and restore the unrotated coordinate system.

9. glPushMatrix()

There is a stack of matrices for each of the matrix mode. In GL_MODELVIEW mode, the stack depth is atleast 32. In other modes, GL_COLOR, GL_PROJECTION, and GL_TEXTURE, the depth is atleast 2. The current matrix in any mode is the matrix on the top of the stack for that mode.

10. glPopMatrix()

glPopMatrix pops the current matrix stack, replacing the current matrix with the one below it on the stack. Initially, each of the stack contains one matrix, an identity matrix. It is an error to push a full matrix stack or pop a matrix stack that contains only a single matrix. In either case, the error flag is set and no other change is made to GL state.

11. glutSwapBuffers()

Usage: void glutSwapBuffers(void);

Description: Performs a buffer swap on the layer in use for the current window. Specifically, glutSwapBuffers promotes the contents of the front buffer. The contents of the back buffer then become undefined

12. glPointSize(GLfloat size)

glPointSize specifies the rasterized diameter of points. This value will be used rasterize points. Otherwise, the value written to the shading language built-in variable gl-PointSize will be used. The point size specified by glPointSize is always returned when GL_POINT_SIZE is queried.

13. glutKeyboardFunc()

Usage: void glutKeyboardFunc(void(*func)(unsigned char key, int x, int y)

Func: The new keyboard callback function

Description: 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.

14. glLineWidth(GLfloat width)

Parameters: width- Specifies the width of rasterized lines. The initial value is 1.
Description: glLineWidth specifies the rasterized width of lines. The actual width is determined by rounding the supplied width to the nearest integer. i pixels are filled in each column that is rasterized, where I is the rounded value of width.

15. glLoadIdentity(void)

glLoadIdentity replaces the current matrix with the identity matrix. It is semantically equivalent to calling glLoadMatrix with the identity matrix.

4.2 User Defined Functions

1. void init()

Used to set Matrix Mode and clipping coordinates and set background color.

2. void title()

Used to give title like street light,power house and electric power generation.

3. void streetlight()

Used to build streetlight.

4. void background()

Used for creating background such as green ground and mid night blue sky.

5. void fan1()

Used for creating turbine fan 1.

6. void fan2()

Used for creating turbine fan 2.

7. void fan3()

Used for creating turbine fan 3.

8. void fan4()

Used for creating turbine fan 4.

9. void wires()

Used for creating wires that will be connected to between fans and houses.

10. void powerstation()

Used for creating the view of powerstation.

11. void road()

Road function to create roads.

12. void clouds()

clouds function to create clouds.

13. void roof()

roof function to create the roof of house.

14. void house()

house function to create the modern house.

15. void fanpole1()

Used to create the pole of fan1.

16. void fanpole()2

Used to create the pole of fan2.

17. void fanpole()3

Function used to create the pole of fan3.

18. void fanhouse()

Function used to create the fanhouse(bottom,roof and upper part).

19. void woman()

Function used to create the woman.

20. void mykeys(unsigned char key,int m,int n)

Function used to set up keyboard interaction.

21. void display(void)

Used to display our output in output window.

22. void spinclockwise(void)

Function to rotate the fan flaps in slow clockwise direction.

23. void spinanticlockwise(void)

Function to rotate the fan flaps in slow anticlockwise direction.

24. void spinclockwise(void)

Function to rotate the fan flaps in fast clockwise direction.

25. void spinanticlockwise(void)

Function to rotate the fan flaps in fast anticlockwise direction.

26. void reshape()

Function is used everytime window is reshaped or resized .

27. void menu()

Function to display menu and process menu events.

28. void processMenu()

Function to display and handle menu options.

4.3 PSUEDOCODE

main.cpp

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

static GLfloat spin=360.0;/*fan rotation variable*/
static GLfloat u=0.45;
static GLfloat v=0.45;
static GLfloat w=0.45;
static GLfloat b=0.45;
static GLfloat c=0.00;
static GLfloat d=0.00;
static GLfloat e=0.00;
static GLfloat a=-40; /*clouds translation variable*/
static int z=0;
GLfloat x=0;
GLfloat y=0;
int m,n;
void declare(char *string)
{
    while(*string)
      glutBitmapCharacter(GLUT_BITMAP_8_BY_13, *string++);
}
void init(void)
{
         glClearColor(1.0,1.0,1.0,1.0);
         glShadeModel(GL_FLAT);
}
void title()
{
    glColor3f(u,v,w);
    glRasterPos2f(0,13);
    declare("POWER HOUSE");
          glRasterPos2f(20,13);
    declare("STREET LIGHT");
} void title1()
{
       glColor3f(0.0,0.0,1.0);
        glRasterPos2f(-22,25);
      declare("ELECTRIC POWER GENERATION THROUGH WIND MILL");
}
   void streetlight
```

```
{
	glPushMatrix();/*1st street light*/
	glLoadIdentity();
	glColor3f(0.2,0.2,0.2);
	glBegin(GL_POLYGON);
	glVertex3f(28.0,-20.0,2.0);
	glVertex3f(29.0,-20.0,3.0);
	glVertex3f(29.0,10.0,4.0);
	glVertex3f(28.0,10.0,2.0);
	glEnd();
	glPopMatrix();
	glPushMatrix();
	  glColor3f(0.3,0.15,0.1);
	glBegin(GL_POLYGON);
	glVertex3f(26.0,6.0,2.0);
	glVertex3f(31.0,7.0,3.0);
	glVertex3f(31.0,6.0,4.0);
	glVertex3f(26.0,7.0,2.0);
	glEnd();
	glPopMatrix();
	glPushMatrix();
	glColor3f(b,b,b);
	glTranslatef(24.5,4.0,1.0);
	glRotatef(260,0,0,1);
	glScalef(1,3.5,1);
	glutSolidCube(2);
	glPopMatrix();
	glPushMatrix();/*2nd street light*/
	glLoadIdentity();
	  glColor3f(0.2,0.2,0.2);
	glBegin(GL_POLYGON);
	glVertex3f(16.1,-10.0,2.0);
	glVertex3f(16.9,-10.0,3.0);
	glVertex3f(16.9,14.0,4.0);
	glVertex3f(16.1,14.0,2.0);
	glEnd();
	glPopMatrix();
	glPushMatrix();
	glColor3f(0.3,0.15,0.1);
	glBegin(GL_POLYGON);
	  glVertex3f(14.5,12.0,2.0);
	glVertex3f(18.5,13.0,3.0);
	glVertex3f(18.5,12.0,4.0)
```

```
	glVertex3f(14.5,13.0,2.0);
	glEnd();
	glPopMatrix();
	glPushMatrix();
	glColor3f(b,b,b);
	glTranslatef(13.5,10.5,1.0);
	glRotatef(260,0,0,1);
	glScalef(1,3.5,1);
	glutSolidCube(1.5);
	glPopMatrix();
}
void background()
{
	glColor3f(0.0,0.1,0.0);
	glBegin(GL_POLYGON);//green ground
	glVertex2i(-250.0,-250.0);
	glVertex2i(250.0,-250.0);
	glVertex2i(250.0,0.0);
	glVertex2i(-250.0,0.0);
	glEnd();
	glColor3f(0.1 ,0.1,0.1);
	glBegin(GL_POLYGON);//mid night blue sky
	glVertex2i(-250.0,0.0);
	glVertex2i(-250.0,250.0);
	glVertex2i(250.0,250.0);
	glVertex2i(250.0,0.0);
	glEnd();

}
void fan1()
{
	glPushMatrix();
	glLoadIdentity();
	glColor3f(1,1,1);
	glTranslatef (-8.0,20.0, 2.0);/*rotation about fixed point*/
	glRotatef(spin,0.0,0.0,1.0);
	glTranslatef (8.0,-20.0, -2.0);
	glBegin(GL_TRIANGLES);/*1st fan*/
	glVertex3f(-8.0,20.0,2.0);
	glVertex3f(-12.0,16.0,3.0);
	glVertex3f(-12.0,18.0,4.0);
	  glVertex3f(-8.0,20.0,2.0);
	glVertex3f(-4.0,24.0,3.0);
	glVertex3f(-4.0,22.0,4.0);
```

```
	glEnd();
	glPopMatrix();
}
void fan2()
{
	glPushMatrix();
	glLoadIdentity();
	glTranslatef (-20.0, 20.0, 2.0);/*rotation about fixed point*/
	glRotatef(spin,0.0,0.0,1.0);
	glTranslatef (20.0, -20.0,-2.0);
	glColor3f(1,1,1);
	glBegin(GL_TRIANGLES);/*2nd fan*/
	glVertex3f(-20.0,20.0,2.0);
	glVertex3f(-25.0,17.0,3.0);
	glVertex3f(-25.0,19.0,4.0);
	  glVertex3f(-20.0,20.0,2.0);
	glVertex3f(-15.0,23.0,3.0);
	glVertex3f(-15.0,21.0,4.0);
	glEnd();
	glPopMatrix();
}
void fan3()
{
	glPushMatrix();
	glLoadIdentity();
	glTranslatef (-32.0, 20.0, 2.0);/*rotation about fixed point*/
	glRotatef(spin,0.0,0.0,1.0);
	glTranslatef (32.0, -20.0, -2.0);
	glColor3f(1,1,1);
	glBegin(GL_TRIANGLES);/*2nd fan*/
	glVertex3f(-32.0,20.0,2.0);
	glVertex3f(-36.0,16.0,3.0);
	glVertex3f(-36.0,18.0,4.0);
	glVertex3f(-32.0,20.0,2.0);
	glVertex3f(-28.0,24.0,3.0);
	glVertex3f(-28.0,22.0,4.0);
	glEnd();
	glPopMatrix();
}
void fan4()
{
	glPushMatrix();
	glLoadIdentity();
	glColor3f(1,1,1);
```

```
	glTranslatef (28.0,25.0, 2.0);/*rotation about fixed point*/
	glRotatef(spin,0.0,0.0,1.0);
	glTranslatef (-28.0,-25.0, -2.0);
	glBegin(GL_TRIANGLES);/*4th fan*/
	glVertex3f(28.0,25.0,2.0);
	glVertex3f(24.0,21.0,3.0);
	glVertex3f(24.0,23.0,4.0);
	glVertex3f(28.0,25.0,2.0);
	glVertex3f(32.0,29.0,3.0);
	glVertex3f(32.0,27.0,4.0);
	glEnd();
	glPopMatrix();
}
void wires()
{
	glColor3f(.7,.5,.7);
	glEnable(GL_LINE_STIPPLE);
	glLineStipple(1,0x00FF);
	glBegin(GL_LINES);
	glVertex2f(-8.0,7.0);
	glVertex2f(-32.0,7.0);
	glVertex2f(-8.0,10.0);
	glVertex2f(1.5,10.0);
	glVertex2f(26.5,7.0);
	glVertex2f(14.5,12.0);
	glVertex2f(31.0,7.0);
	glVertex2f(18.0,12.0);
	glEnd();
	glDisable(GL_LINE_STIPPLE);
}
void powerstation()
{
	GLint ax=1.5,ay=8;
	glColor3f(1.0,0.25,0.1);
	glBegin(GL_POLYGON);//bottom part of power station home(anti
clkwise)
	glVertex2i(ax,ay);//a
	glVertex2i(ax-2,ay-2);//b
	glVertex2i(ax-2,ay-8);//c
	glVertex2i(ax+2,ay-8);//d
	glVertex2i(ax+2,ay-2);//e
	glEnd();
	glColor3f(0.7,0.5,0.3);
	glBegin(GL_POLYGON);//roof or upper part (from a)
```

```
	glVertex2i(ax,ay+3);//a
	glVertex2i(ax-3,ay-3);//b
	glVertex2i(ax+3,ay-3);//e
	glEnd();
	glColor3f(v,v,w);
	glBegin(GL_POLYGON);/* door */
	glVertex2i(ax-1,ay-5.0);//top left
	glVertex2i(ax-1.0,ay-8.0);//bottom left
	glVertex2i(ax+1.0,ay-8.0);// bottom right
	glVertex2i(ax+1.0,ay-5.0);//top right
	glEnd();
}
void road()//road
{
	glColor3f(0.0,0.0,0.0);
	glBegin(GL_POLYGON);
	glVertex2f(-1,0);
	glVertex2f(4,0);
	glVertex2f(43,-39);
	glVertex2f(37,-40);
	glEnd();
}
void clouds()
{
	glPushMatrix();
	glColor3f (0.4, 0.7,0.9);
   glLoadIdentity ();             /* clear the matrix */
           //viewing transformation
   glTranslatef(a+1, 40.0, -9.0);
   glScalef (2.0, 1.0, 1.0);  /* modeling transformation */
   glutSolidSphere (2.0,50,56);
   glLoadIdentity();
   glTranslatef(a-2.0,40.0, -9.0);
   glScalef (2.0, 1.0, 1.0);
   glutSolidSphere (2.0,50,56);
   glLoadIdentity();
   glTranslatef(a+7.0,40.0, -9.0);
   glScalef (2, 1.0, 1.0);
   glutSolidSphere (2.0,50,56);
   glLoadIdentity();
   glTranslatef(a-7.0,40.0, -9.0);
   glScalef (2, 1.0, 1.0);
   glutSolidSphere (2.0,50,56);
   glLoadIdentity();
```

```
   glTranslatef(a+18.0,40.0, -9.0);
   glScalef (2, 1.0, 1.0);
   glutSolidSphere (2.0,50,56);
   glLoadIdentity();
   glTranslatef(a+25.0,40.0, -9.0);
   glScalef (2, 1.0, 1.0);
   glutSolidSphere (2.0,50,56);
    glLoadIdentity();
   glTranslatef(a+36.0,40.0, -9.0);
   glScalef (3.0, 1.0, 1.0);
   glutSolidSphere (2.0,50,56);
   glLoadIdentity();
   glTranslatef(a+50.0,40.0, -9.0);
   glScalef (2, 1.0, 1.0);
   glutSolidSphere (2.0,50,56);
   glLoadIdentity();
   glTranslatef(a+56.0,40.0, -9.0);
   glScalef (2, 1.0, 1.0);
   glutSolidSphere (2.0,50,56);
   glPopMatrix();
}
void roof(GLint rux,GLint ruy,GLint rdx,GLint rdy)
{
      glPushMatrix();
      glColor3f(1,0.25,0.1);
      glBegin(GL_LINE_STRIP);
      glVertex2i(rux,ruy);//roof up
      glVertex2i(rdx,rdy);//roof down
      glEnd();
      glPopMatrix();
}
void house(GLint rux,GLint ruy,GLint rdx,GLint rdy)//for modern home
{
      GLint blx=rdx,bly=rdy-9,brx=rdx+10,bry=rdy-9,kx=rdx-8,ky=rdy+1;
      GLfloat i;
      for(i=0;i<440;i=i+1)
            roof(rux+i/40,ruy,rdx+i/40,rdy);/* draw straws */
      glColor3f(0.3,0.25,0.1);
      glBegin(GL_POLYGON);/* front wall */
      glVertex2i(rdx,rdy);//roof left
      glVertex2i(rdx+10,rdy);//roof right
      glVertex2i(brx,bry);//base right
      glVertex2i(blx,bly);//base left
      glEnd();
```

```
        glColor3f(0.3,0.15,0.1);
      glBegin(GL_POLYGON);/* side wall */
      glVertex2i(rux,ruy);//roof up
      glVertex2i(kx,ky);//backK
      glVertex2i(kx,ky-6);//backK base
      glVertex2i(blx,bly);
      glVertex2i(rdx,rdy);
      glEnd();
      glColor3f(v,v,w);
      glBegin(GL_POLYGON);/* window */
      glVertex2i(kx+2,ky-2.5);//top left
      glVertex2i(kx+2,ky-5.5);//bottom left
      glVertex2i(blx-3,bly+3.5);//bottom right
      glVertex2i(blx-3,bly+6.9);//top right
      glEnd();
        glColor3f(v,v,w);
      glBegin(GL_POLYGON);/* door */
      glVertex2i(blx+3.5,bly+5);//top left
      glVertex2i(blx+3.5,bly);//bottom left
      glVertex2i(brx-3.5,bry);//bottom right
      glVertex2i(brx-3.5,bry+5);//top right
      glEnd();
}
void fanpole1()
{
      glColor3f(1.0,1.0,1.0);
      glBegin(GL_TRIANGLE_STRIP);
      glVertex2f(-8.1,20.0);
      glVertex2f(-7.9,20.0);
      glVertex2f(-8.5,0.0);
      glVertex2f(-7.5,0.0);
      glEnd();
}
void fanpole2()
{
      glColor3f(1.0,1.0,1.0);
      glBegin(GL_TRIANGLE_STRIP);
      glVertex2f(-20.1,20.0);
      glVertex2f(-19.9,20.0);
      glVertex2f(-20.5,0.0);
      glVertex2f(-19.5,0.0);
      glEnd();
```

```
void fanpole3()
{
       glColor3f(1.0,1.0,1.0);
       glBegin(GL_TRIANGLE_STRIP);
       glVertex2f(-32.1,20.0);
       glVertex2f(-31.9,20.0);
       glVertex2f(-32.5,0.0);
       glVertex2f(-31.5,0.0);
       glEnd();
}
void fanhouse()// fan house
{   GLint ax=28,ay=30;
       glColor3f(0.7,0.5,0.3);
       glBegin(GL_POLYGON);//bottom part of fan house (anti clkwise)
       glVertex2i(ax,ay);//a
       glVertex2i(ax-3,ay-2);//b
       glVertex2i(ax-3,ay-9);//c
       glVertex2i(ax+3,ay-9);//d
       glVertex2i(ax+3,ay-2);//e
       glEnd();
       glColor3f(0.0,0.3,0.3);
       glBegin(GL_POLYGON);//roof or upper part of fan house(from a)
       glVertex2i(ax,ay+3);//a
       glVertex2i(ax-4,ay-3);//b
       glVertex2i(ax+4,ay-3);//e
       glEnd();
         glColor3f(0.3,0.15,0.1);
       glBegin(GL_TRIANGLE_STRIP);
       glVertex2f(30.1,21.0);
       glVertex2f(29.9,21.0);
       glVertex2f(30.5,0.0);
       glVertex2f(29.5,0.0);
       glEnd();
       glBegin(GL_TRIANGLE_STRIP);
       glVertex2f(26.1,21.0);
       glVertex2f(25.9,21.0);
       glVertex2f(26.5,0.0);
       glVertex2f(25.5,0.0);
       glEnd();
         glColor3f(0.0,0.3,0.3);
       glBegin(GL_TRIANGLE_STRIP);
       glVertex2f(30.0,22.0);
       glVertex2f(29.5,22.0);
       glVertex2f(29.0,19.0);
```

```
glVertex2f(28.5,19.0);
glEnd();
glBegin(GL_TRIANGLE_STRIP);
glVertex2f(26.5,22.0);
glVertex2f(26.0,22.0);
glVertex2f(25.5,19.0);
glVertex2f(25.0,19.0);
glEnd();
  glColor3f(0,0,d);
glEnable(GL_LINE_STIPPLE);
glLineStipple(1,0x00FF);
glBegin(GL_LINES);
glVertex2f(25.5,19.0);
glVertex2f(25.5,-1.0);
glVertex2f(25.0,19.0);
glVertex2f(25.0,-1.0);
glVertex2f(25.25,19.0);
glVertex2f(25.25,-1.0);
glEnd();
glColor3f(0,0,e);
glBegin(GL_LINES);
glVertex2f(28.5,19.0);
glVertex2f(28.5,-1.0);
glVertex2f(29.0,19.0);
  glVertex2f(29.0,-1.0);
glVertex2f(28.75,19.0);
glVertex2f(28.75,-1.0);
glEnd();
glDisable(GL_LINE_STIPPLE);
glColor3f(0,0,1);
glBegin(GL_POLYGON);
glVertex2f(25.0,-1.0);
glVertex2f(29.5,-1.0);
glVertex2f(29.5,-2.0);
glVertex2f(25.0,-2.0);
glEnd();
glColor3f(0.3,0.15,0.1);
glBegin(GL_POLYGON);
glVertex2f(24.5,-1.0);
glVertex2f(25.0,-1.0);
glVertex2f(25.0,-2.0);
glVertex2f(24.5,-2.0);
  glEnd();
glBegin(GL_POLYGON);
```

```
	glVertex2f(24.5,-2.0);
	glVertex2f(30.0,-2.0);
	glVertex2f(30.0,-3.0);
	glVertex2f(24.5,-3.0);
	glEnd();
	glBegin(GL_POLYGON);
	glVertex2f(29.5,-1.0);
	glVertex2f(30.0,-1.0);
	glVertex2f(30.0,-2.0);
	glVertex2f(29.5,-2.0);
	glEnd();
}
void woman()
{
	glClearColor(0.48,0.5,0.5,0.0);
	glColor3f(1.0,0.0,0.0);

	glBegin(GL_POLYGON);//veil
		glVertex2f(21.0+x,-17.0+y);
		glVertex2f(22.0+x,-17.0+y);
		glVertex2f(22.5+x,-20.0+y);
		glVertex2f(20.5+x,-20.0+y);
	glEnd();
	glColor3f(0.3,0.15,0.1);
	glBegin(GL_POLYGON);//face
		glVertex2f(21.0+x,-18.0+y);
		glVertex2f(21.0+x,-17.0+y);
		glVertex2f(22.0+x,-17.0+y);
		    glVertex2f(22.0+x,-18.0+y);
	glEnd();
	glBegin(GL_POLYGON);//neck
		glVertex2f(21.5+x,-17.0+y);
		glVertex2f(21.6+x,-17.0+y);
		glVertex2f(21.6+x,-18.5+y);
		glVertex2f(21.5+x,-18.5+y);
	glEnd();
	glColor3f(0.97,0.45,0.84);
	glBegin(GL_POLYGON);//body
		glVertex2f(21.0+x,-18.5+y);
		glVertex2f(22.1+x,-18.5+y);
		glVertex2f(22.1+x,-20.0+y);
		glVertex2f(21.0+x,-20.0+y);
	glEnd();
	glColor3f(0.59,0.137,0.985);
```

```
glBegin(GL_POLYGON);//skirt
         glVertex2f(21.0+x,-20.0+y);
      glVertex2f(22.1+x,-20.0+y);
      glVertex2f(22.7+x,-21.0+y);
      glVertex2f(20.5+x,-21.0+y);
glEnd();
glColor3f(0.0,0.0,0.0);
glPointSize(1.4);
glBegin(GL_POINTS);//eyes
      glVertex2f(21.3+x,-17.30+y);
      glVertex2f(21.8+x,-17.30+y);
glEnd();
glBegin(GL_LINES);//nose
      glVertex2f(21.6+x,-17.6+y);
      glVertex2f(21.6+x,-17.3+y);
glEnd();
glBegin(GL_LINES);//smile
      glVertex2f(21.5+x,-17.8+y);
      glVertex2f(21.8+x,-17.8+y);
      glEnd();
 glColor3f(0.3,0.15,0.1);
glBegin(GL_POLYGON);//hand 1
      glVertex2f(21.0+x,-18.5+y);
      glVertex2f(20.5+x,-20.0+y);
      glVertex2f(21.0+x,-19.0+y);
glEnd();


glBegin(GL_POLYGON);//hand 2
      glVertex2f(22.1+x,-18.5+y);
      glVertex2f(22.7+x,-19.0+y);
      glVertex2f(22.1+x,-19.0+y);
glEnd();

glColor3f(1,1,0.4);
glBegin(GL_POLYGON);//torch
      glVertex2f(22.7+x,-19.0+y);
      glVertex2f(22.4+x,-19.0+y);
      glVertex2f(22.5+x,-19.7+y);
      glVertex2f(23.0+x,-19.7+y);
glEnd();
  glColor3f(c,c,c);
glBegin(GL_POLYGON);//torch light
      glVertex2f(22.7+x,-19.2+y);
```

```
            glVertex2f(22.4+x,-19.2+y);
            glVertex2f(22.5+x,-19.7+y);
            glVertex2f(23.0+x,-19.7+y);
      glEnd();

      glColor3f(c,c,c);
      glEnable(GL_LINE_STIPPLE);
      glLineStipple(1,0x00FF);//torch light line
      glBegin(GL_LINES);
            glVertex2f(22.5+x,-19.2+y);
            glVertex2f(24.0+x,-25.0+y);
            glVertex2f(22.6+x,-19.2+y);
            glVertex2f(25.5+x,-25.0+y);
            glVertex2f(22.7+x,-19.2+y);
            glVertex2f(27.0+x,-25.0+y);
      glEnd();

}
void mykey(unsigned char key,int m,int n)
{
    if(key=='w')   y+=.1,x-=0.1;
      if(key=='s')    y-=.1,x+=0.1;
      glutPostRedisplay();
}
void display(void)
{
      int b=0;
      glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
      if(z<50)
      {
            for(z=0;z<=1500;z++)
            {
                  title1();
                  glutPostRedisplay();
                  glutSwapBuffers();
                  glFlush();
            }
      }
        else
      {

      background();
      fanhouse();
      title();
```

```
       road();
       house(-29,-23,-24,-33);
       house(0,-11,5,-21);
       house(-21,-1,-14,-11);
       clouds();
       powerstation();
         wires();
       streetlight();
       woman();

       fanpole1();
       fanpole2();
       fanpole3();


       fan1();
       fan2();
       fan3();
       fan4();
       glutSwapBuffers();
       glFlush();
       }
}
void spinclockwise(void)//slow clockwise move

{
       w=0.3;u=0;v=1;b=0.5;c=1;d=1;e=0;
        a=a+0.1;
           if(a>40)
        a-=100.0;
        spin=spin-1.0;
        if(spin<0)
       spin=spin+360.0;

       glutPostRedisplay();

 }
void anticlockwise(void )//slow ant clockwise move
{
       u=0;w=.3;v=1;b=0.5;c=1;d=1,e=0;
        if(a==40)
        a=40;
        a=a-0.1;
          if(a<-100
```

```
         a+=100.0;
         if(spin==360.0)
       spin=spin-360;
       spin=spin+1.0;
       if(spin>360.0)
         spin=spin-360.0;
       glutPostRedisplay();

}
void spinclockwise1(void)//fast  clockwise move

{
       u=0;w=.4;v=1;b=1;c=0.0;d=e=1;
        a=a+0.3;
       if(a>40)
        a-=100.0;
       spin=spin-10.0;
       if(spin<0)
           spin=spin+360.0;
       glutPostRedisplay();
}
void anticlockwise1(void )//fast  ant clockwise move

{
       u=0;w=.4;v=1;b=1;c=0.0;d=e=1;
        if(a==40)
        a=40;
         a=a-0.3;
   if(a<-100)
         a+=100.0;
   if(spin==360.0)
         spin=spin-360;
       spin=spin+10.0;
       if(spin>360.0)
             spin=spin-360.0;
       glutPostRedisplay();
}
void reshape(int w, int h)
{
glViewport(0, 0, (GLsizei) w, (GLsizei) h);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(-35.0, 35.0, -45.0, 45.0, -20.0, 20.0);
glMatrixMode(GL_MODELVIEW);
```

```
glLoadIdentity();
}
void menu(int id )
{
      switch(id)
      {
      case 1: u=v=w=b=0.45;c=d=e=1;
                  glutIdleFunc(display);
                  break;
      case 2: glutIdleFunc(spinclockwise);
                  break;
      case 3: glutIdleFunc(anticlockwise);
                  break;
      case 4: glutIdleFunc(spinclockwise1);
                  break;
      case 5: glutIdleFunc(anticlockwise1);
                  break;
      case 6:exit(0);
      }
}
int main(int argc,char *argv[])
{

printf("************************************************************\n");
    printf("                          GROUP MEMBERS                  ");

printf("\n************************************************************");

printf("\n************************************************************");

printf("\n************************************************************");
    printf("\n**   1.  Shriya         Khar              1BI19CS146");
    printf("\n**   2.  Zainab         Drabu              1BI19CS193");

printf("\n************************************************************");
    glutInit(&argc,argv);
      glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB|GLUT_DEPTH);
      glutInitWindowSize(500,500);
      glutInitWindowPosition(100,100);
      glutCreateWindow("WIND ENERGY");
        glutDisplayFunc(display);
      glutReshapeFunc(reshape);
      glutKeyboardFunc(mykey);
      init();
```

```
       glutCreateMenu(menu);
       glutAddMenuEntry("No Wind",1);
       glutAddMenuEntry("Wind CW",2);
       glutAddMenuEntry("Wind ACW",3);
       glutAddMenuEntry ("Fast Wind CW",4);
       glutAddMenuEntry("Quit",6);
       glutAttachMenu(GLUT_RIGHT_BUTTON);
       glutMainLoop();
       return 0;
}
```

Chapter -5

SNAPSHOTS

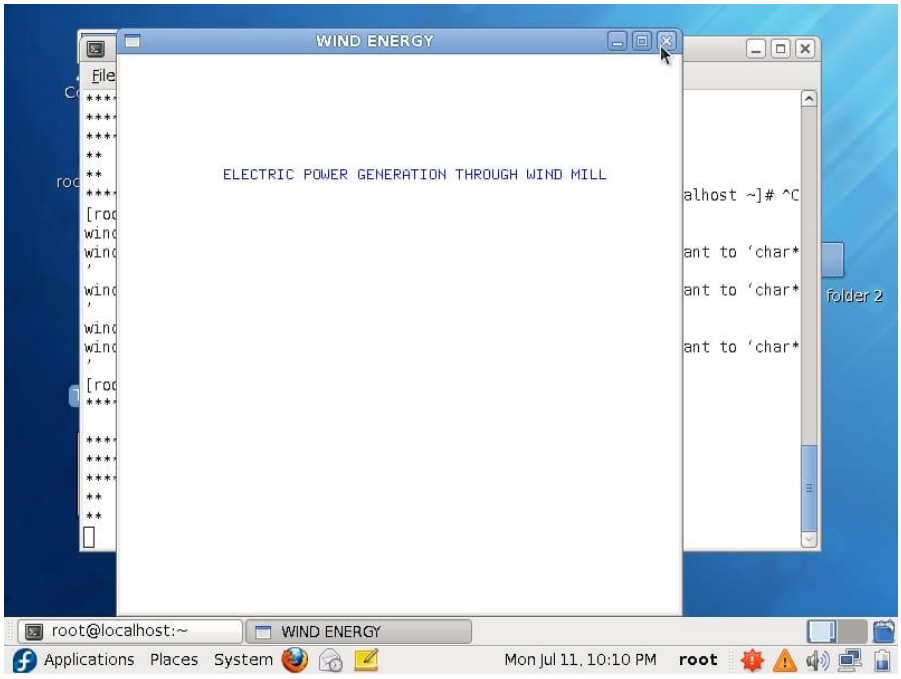

Figure 5.1 : Intro Page

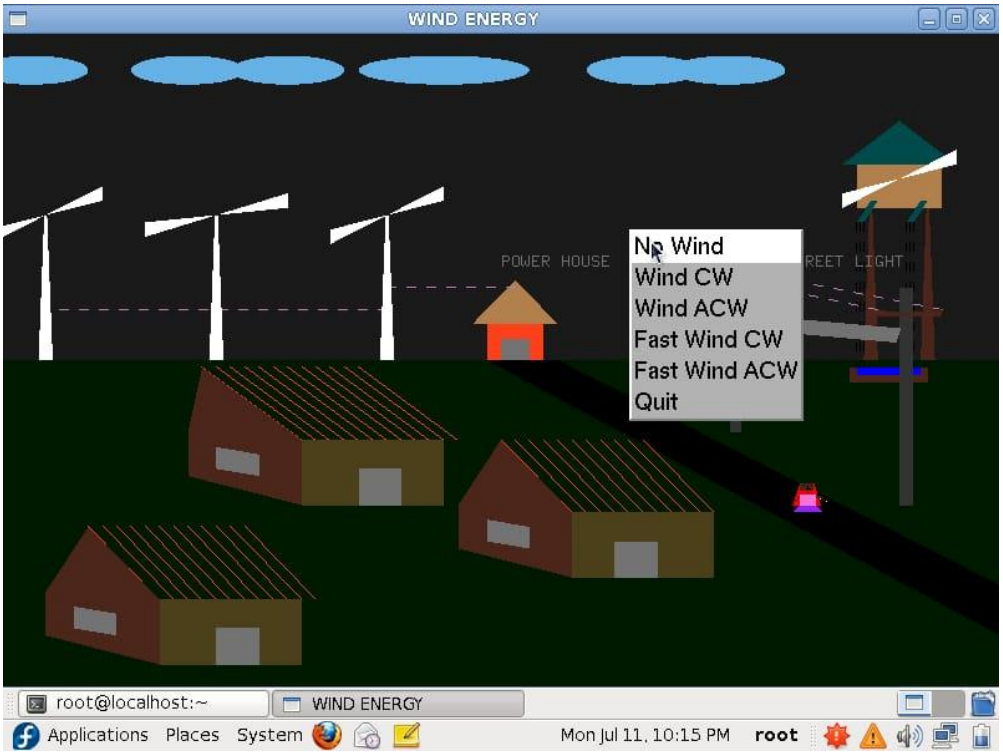

Figure 5.2: No Wind

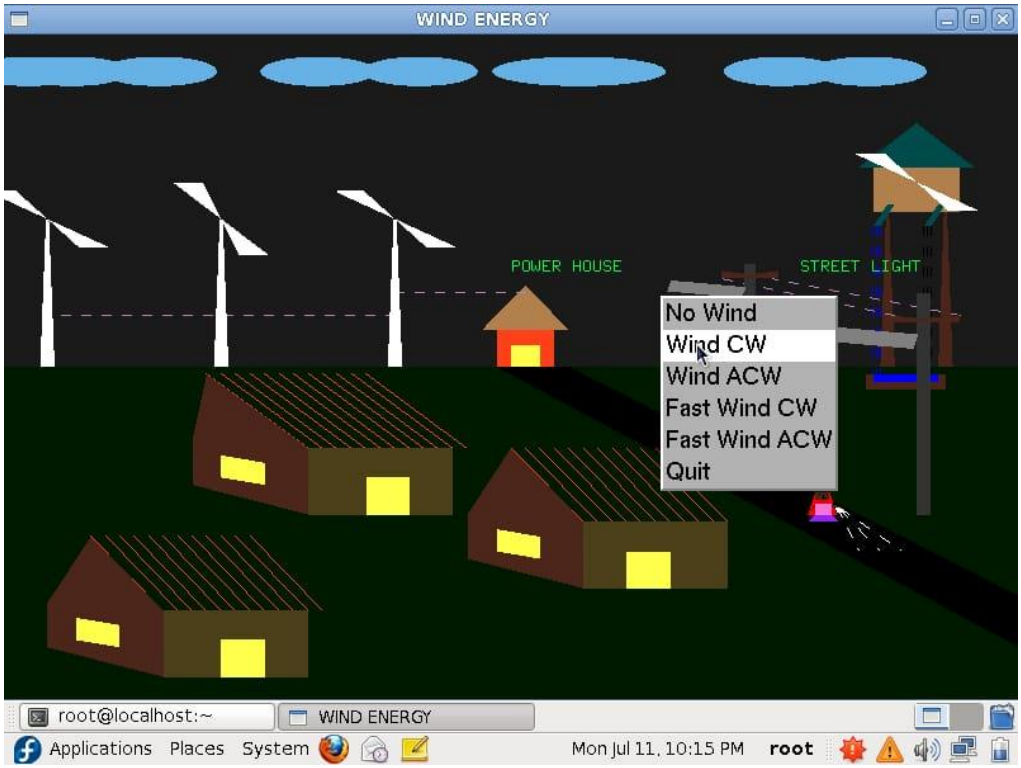

Figure 5.3: Wind moving slowly in clockwise direction

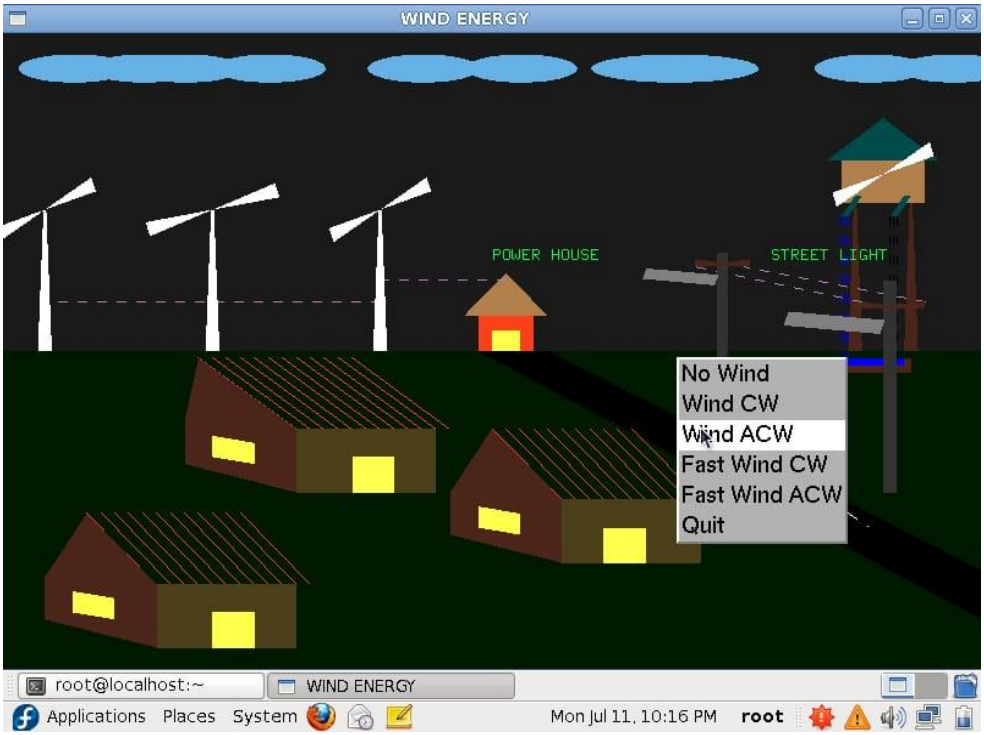

Figure 5.4: Wind moving slowly in anti-clockwise direction

Figure 5.5: Wind moving fastly in clockwise direction

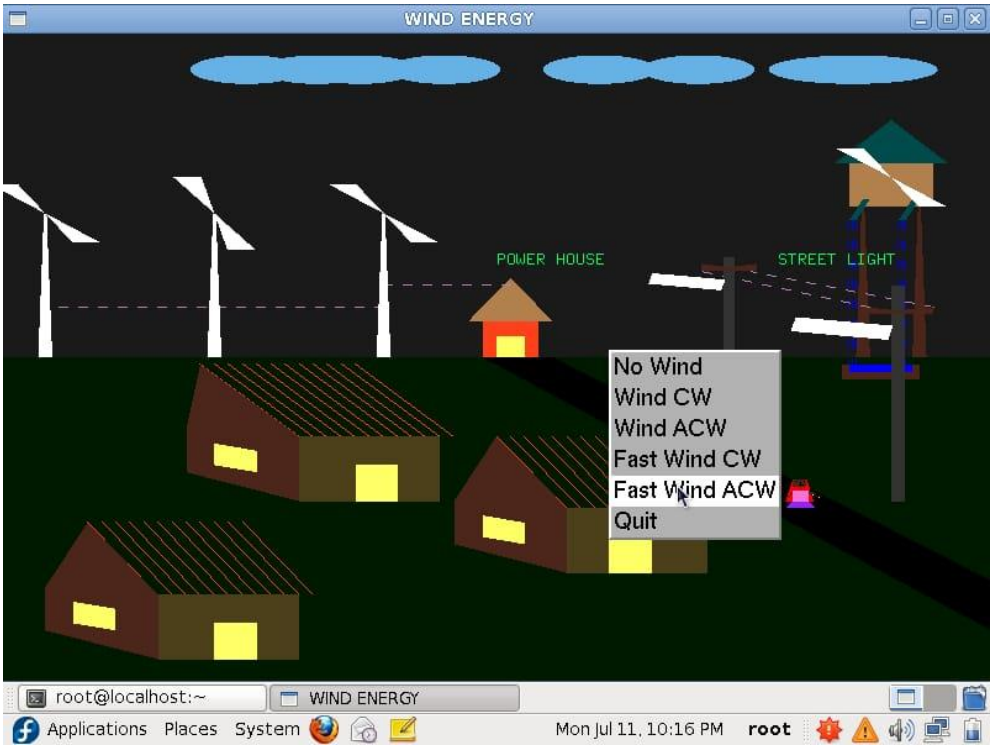

Figure 5.6:Wind flowing fastly in anti-clockwise direction

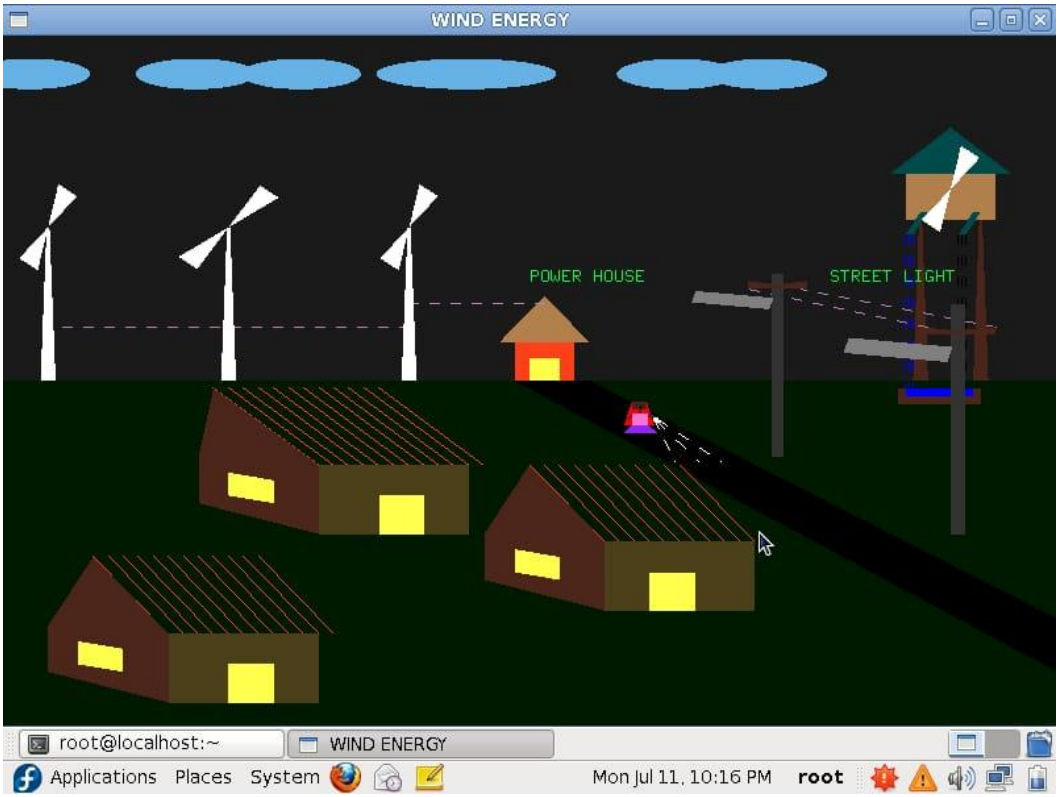

Figure 5.7: Scene3-Girl moving in left direction

Figure 5.8: Girl moving in right direction

Chapter -6

CONCLUSION

"Wind Energy Simulation" project was done in a perspective of understanding the openGL software toolkit. Through this project ,we have acquired a much deeper knowledge of the openGL constraints and computer graphics as a whole.

We thus would like to emphasize the importance of this project to many other perspectives of Technical, mathematical, graphical and software concepts which we were unaware of.

6.1 Future Enhancements

- In future the same project can be enhanced in such a way that we can interact more with the project. Also the project can be implemented in 3D Space.
- A vast amount of future work can be possible by following investigations and strategies.
- More features can be included and can be modified in a more versatile way.

BIBLIOGRAPHY

Reference Books

[1] Donald Hearn & Pauline Baker: Computer Graphics with OpenGL Version, 3rd/4th Edition, Pearson Education, 2011

[2] Edward Angel: Interactive Computer Graphics- A Top Down approach with OpenGL, 5th Edition, Pearson Education

[3] James D Foley, Andries Van Dam, Steven K Feiner, John F Huges Computer Graphics with OpenGL, Pearson Education

[4] Macro Cantu: Mastering Delphi

Websites

[1] https://www.opengl.org/

[2] https://learnopengl.com/