VISVESVARAYA TECHNOLOGICAL UNIVERSITY

BELAGAVI-590018

A Computer Graphics Mini Project Report

on

WINDMILL

Submitted in partial fulfillment of the requirements for the VI semester

Computer Science and Engineering of Visvesvaraya Technological University, Belagavi

Submitted by:

KASHISH	*1RN20CS064*
POOJITHA C	*1RN20CS098*

Under the Guidance of:

Dr. A N Ramya Shree
Associate Professor
Dept. of CSE

ESTD : 2001
An Institute with a Difference

Department of Computer Science and Engineering
(Accredited by NBA up to 30/6/2025)

RNS Institute of Technology

Channasandra, Dr.Vishnuvardhan Road, Bengaluru-560 098

2023

RNS INSTITUTE OF TECHNOLOGY

Channasandra, Dr.Vishnuvardhan Road, Bengaluru-560098

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

(Accredited by NBA up to 30/6/2025)

ESTD : 2001

An Institute with a Difference

CERTIFICATE

This is to certify that the mini project work entitled **WINDMILL** has been successfully carried out by **KASHISH** bearing USN **1RN20CS064** and **POOJITHA C** bearing USN **1RN20CS098**, bonafide students of **RNS Institute of Technology** in partial fulfillment of the requirements for the 6th semester **Computer Science and Engineering of Visvesvaraya Technological University"**, Belagavi, during academic year 2023. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The project report has been approved as it satisfies the CG laboratory requirements of 6th semester BE, CSE.

Signature of the Guide	Signature of the HoD	Signature of the Principal
Dr. A N Ramya Shree	**Dr. Kiran P**	**Dr. Ramesh Babu H S**
Associate Professor	Professor & Head	Principal
Dept. of CSE	Dept. of CSE	

External Viva:

Name of the Examiners	Signature with Date
1.	
2.	

Acknowledgement

The successful completion of any achievement is not solely dependent on individual efforts but also on the guidance, encouragement, and cooperation of intellectuals, elders, and friends. We would like to take this opportunity to express our heartfelt gratitude to all those who have contributed to the successful execution of this project.

First and foremost, we extend our profound thanks to **Sri. Satish R Shetty**, Managing Trustee of R N Shetty Trust and Chairman of RNS Group of Institutions, and **Sri. Karan S Shetty** , CEO of RNS Group of Institutions, Bengaluru, for providing a conducive environment that facilitated the successful completion of this project.

We would also like to express our sincere appreciation to our esteemed Director, **Dr. M K Venkatesha**, for providing us with the necessary facilities and support throughout the duration of this work.

Our heartfelt thanks go to our respected Principal, **Dr. Ramesh Babu H S**, for his unwavering support, guidance, and encouragement that played a vital role in the completion of this project.

We would like to extend our wholehearted gratitude to our HOD, **Dr. Kiran P**, Professor, and Head of the Department of Computer Science & Engineering, RNSIT, Bangalore, for his valuable suggestions and expert advice, which greatly contributed to the success of this endeavor.

A special word of thanks is due to our project guide,**Dr. A N Ramya Shree**, Associate Professor in the Department of CSE, RNSIT, Bangalore, for her exceptional guidance, constant encouragement, and unwavering assistance throughout the project.

We would also like to express our sincere appreciation to all the teaching and non-teaching staff of the Department of Computer Science & Engineering, RNSIT, for their consistent support and encouragement.

Once again, we express our deepest gratitude to everyone involved, as their support and cooperation were instrumental in the successful completion of this project.

Abstract

The aim of this project is to develop the computer graphics project using OpenGL to demonstrate the working of windmills in production of wind energy. For power to be created, the blades of a wind mill must first catch wind and be moved by it. Wind is infinite that it is the cheapest and most useful power source. Therefore there is usage of simple OpenGL functions to demonstrate rotating fans which generate electricity when rotated. It can also perform operations on selected objects like translation.It provides most of the features that a model should have. The change in direction of rotation of fan in windmill in clockwise and anticlockwise both generate electricity as the result of this project.The interface provided here has a menu when right clicked which is user friendly.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview of Computer Graphics

The term computer graphics has been used in a broad sense to describe almost everything on computers that is not text or sound. Typically, the term computer graphics refers to several different things:

- The representation and manipulation of image data by a computer.
- The various technologies used to create and manipulate images.
- The sub-field of computer science which studies methods for digitally synthesizing and manipulating visual content.

Today, computers and computer-generated images touch many aspects of daily life. Computer images is found on television, in newspapers, for example in weather reports, in all kinds of medical investigation and surgical procedures. A well-constructed graph can present complex statistics in a form that is easier to understand and interpret. In the media such graphs are used to illustrate papers, reports, thesis, and other presentation material.Many powerful tools have been developed to visualize data. Computer generated imagery can be categorized into several different types: 2D, 3D, 4D, 7D, and animated graphics.

As technology has improved, 3D computer graphics have become more common. Computer graphics has emerged as a sub-field of computer science which studies methods for digitally synthesizing and manipulating visual content. Over the past decade, other specialized fields have been developed like information visualization, and scientific visualization more concerned with the visualization of three dimensional phenomena (architectural, meteorological, medical, biological, etc.), where the emphasis is on realistic renderings of volumes, surfaces, illumination sources, and so forth, perhaps with a dynamic component.

1.2 History of Computer Graphics

In 1959, the TX-2 computer was developed at MIT's Lincoln Laboratory. The TX-2 integrated a number of new man-machine interfaces. A light pen could be used to draw sketches on the computer using Ivan Sutherland's revolutionary Sketchpad software.Using a light pen, Sketchpad allowed one to draw simple shapes on the computer screen, save them and even recall them later. The light pen itself had a small photoelectric cell in its tip. This cell emitted an electronic pulse whenever it was placed in front of a computer screen and the screen's electron gun fired directly at it. By simply timing the electronic pulse with the current location of the electron gun, it was easy to pinpoint exactly where the pen was on the screen at any given moment. Once that was determined, the computer could then draw a cursor at that location.

In 1961 another student at MIT, Steve Russell, created the first video game, E. E. Zajac, a scientist at Bell Telephone Laboratory (BTL), created a film called "Simulation of a two-giro gravity attitude control system" in 1963. During 1970s, the first major advance in 3D computer graphics was created at University of Utah by these early pioneers, the hidden-surface algorithm. In order to draw a representation of a 3D object on the screen, the computer must determine which surfaces are "behind" the object from the viewer's perspective, and thus should be "hidden" when the computer creates (or renders) the image.

In the 1980s, artists and graphic designers began to see the personal computer, particularly the Commodore Amiga and Macintosh, as a serious design tool, one that could save time and draw more accurately than other methods. In the late 1980s, SGI computers were used to create some of the first fully computer-generated short films at Pixar. The Macintosh remains a highly popular tool for computer graphics among graphic design studios and businesses. Modern computers, dating from the 1980s often use graphical user interfaces (GUI) to present data and information with symbols, icons and pictures, rather than text. Graphics are one of the five key elements of multimedia technology.

3D graphics became more popular in the 1990s in gaming, multimedia and animation. In 1996, Quake, one of the first fully 3D games, was released. In 1995, Toy Story, the first full-length computer-generated animation film, was released in cinemas worldwide. Since then, computer graphics have only become more detailed and realistic, due to more powerful graphics hardware and 3D modeling software.

1.3 Applications of Computer Graphics

The applications of computer graphics can be divided into four major areas:

- Display of information
- Design
- Simulation and animation
- User interfaces

1.3.1 Display of information

Computer graphics has enabled architects, researchers and designers to pictorially interpret the vast quantity of data. Cartographers have developed maps to display the celestial and geographical information. Medical imaging technologies like Computerized Tomography (CT), Magnetic Resonance Imaging (MRI), Ultrasound, Positron Emission Tomography (PET) and many others make use of computer graphics.

1.3.2 Design

Professions such as engineering and architecture are concerned with design. They start with a set of specification; seek cost-effective solutions that satisfy the specification. Designing is an iterative process. Designer generates a possible design, tests it and then uses the results as the basis for exploring other solutions. The use of interactive graphical tools in Computer Aided Design (CAD) pervades the fields including architecture, mechanical engineering, and the design of very-large-scale integrated (VLSI) circuits and creation of characters for animation.

1.3.3 Simulation and Animation

Once the graphics system evolved to be capable of generating sophisticated images in real time, engineers and researchers began to use them as simulators. Graphical flight simulators have proved to increase the safety and to reduce the training expenses. The field of virtual reality (VR) has opened many new horizons. A human viewer can be equipped with a display headset that allow him/her to see the images with left eye and right eye which gives the effect of stereoscopic vision. This has further led to motion pictures and interactive video games.

1.3.4 User interfaces

Computer graphics has led to the creation of graphical user interfaces (GUI) using which even naive users are able to interact with a computer. Interaction with the computer has been dominated by a visual paradigm that includes windows, icons, menus and a pointing device such as mouse. Millions of people are internet users; they access the internet through the graphical network browsers such as Microsoft internet explorer and Mozilla Firefox.

Chapter 2

OpenGL

OpenGL (Open Graphics Library) is a standard specification defining a cross-language, cross-platform API for writing applications that produce 2D and 3D computer graphics. The interface consists of over 250 different function calls which can be used to draw complex three-dimensional scenes from simple primitives. OpenGL was developed by Silicon Graphics Inc. (SGI) in 1992 and is widely used in CAD, virtual reality, scientific visualization, information visualization, and flight simulation. OpenGL provides a set of commands to render a three dimensional scene. That means you provide the data in an OpenGL-useable form and OpenGL will show this data on the screen (render it). It is developed by many companies and it is free to use. You can develop OpenGL-applications without licensing. OpenGL is a hardware- and system-independent interface. An OpenGL-application will work on every platform, as long as there is an installed implementation. Because it is system independent, there are no functions to create windows etc., but there are helper functions for each platform. A very useful thing is GLUT.

2.1 OpenGL Libraries

Computer Graphics are created using OpenGL, which became a widely accepted standard software system for developing graphics applications. As a software interface for graphics hardware, 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 performs several processing steps on this data to convert it to pixels to form the final desired image in the frame buffer. OpenGL stands for 'open graphics library' graphics library is a collection of API's (Applications Programming Interface). Graphics library functions are:

- GL library (OpenGL in windows) – Main functions for windows.

- GLU (OpenGL utility library) - Creating and viewing objects.
- GLUT (OpenGL utility toolkit)- Functions that help in creating interface of windows

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 (although many modes may interact to determine what eventually ends up in the frame buffer). Primitives are specified, modes are set, and other OpenGL operations are described by issuing commands in the form of function calls. These libraries are included in the application program using preprocessor directives OpenGL User Interface Library (GLUI) is a C++ user interface library based on the OpenGL Utility Toolkit (GLUT) which provides controls such as buttons, checkboxes, radio buttons, and spinners to OpenGL applications. It is window and operating system independent, relying on GLUT to handle all system-dependent issues, such as window and mouse management. The OpenGL Utility Library (GLU) is a computer graphics library. It consists of a number of functions that use the base OpenGL library to provide higher-level drawing routines from the more primitive routines that OpenGL provides. It is usually distributed with the base OpenGL package.

2.2 OpenGL Contributions

It is very popular in the video games development industry where it competes with Direct3D (on Microsoft Windows).OpenGL is also used in CAD, virtual reality, and scientific visualization programs.OpenGL is very portable. It will run for nearly every platform in existence, and it will run well. It even runs on Windows NT 4.0 etc. The reason OpenGL runs for so many platforms is because of its Open Standard. OpenGL has a wide range of features, both in its core and through extensions. Its extension feature allows it to stay immediately current with new hardware features, despite the mess it can cause.

2.3 Limitations

- OpenGL is case sensitive.
- Line Color, Filled Faces and Fill Color not supported.
- Shadow plane is not supported.

Chapter 3

Resource Requirements

3.1 Hardware Requirements

The Hardware requirements are very minimal and the program can be run on most of the machines.Table 3.1 gives details of hardware requirements.

Table 3.1: Hardware Requirements

Processor	Intel Core i3 processor
Processor Speed	1.70 GHz
RAM	4 GB
Storage Space	40 GB
Monitor Resolution	1024*768 or 1336*768 or 1280*1024

3.2 Software Requirements

The software requirements are description of features and functionalities of the system.Table 3.2 gives details of software requirements.

Table 3.2: Software Requirements

Operating System	Windows 8.1
IDE	Code Blocks with c/c++ version 17.0 and higher
OpenGL libraries	glut.h,glu32.lib,opengl32.lib,glut32.lib glu32.dll,glut32.dll,opengl32.dll.

Chapter 4

System Design

4.1 Inbuilt Functions

- **void glutInitDisplayMode(unsigned int mode):**
  This function requests a display with the properties in mode. The value of mode is determined by the logical OR of options including the color model (GLUT_RGB, GLUT_INDEX) and buffering (GLUT_SINGLE, GLUT_DOUBLE).

- **void glutInitWindowPosition(int x, int y):** This specifies the initial position of top-left corner of the windows in pixels.

- **void glutInitWindowSize(int width, int height):** This function specifies the initial height and width of the window in pixels.

- **void glutCreateWindow(char *title):** This function creates a window on the display the string title can be used to label the window. The return value provides a reference to the window that can be used when there are multiple windows.

- **void glutDisplayFunc(void (*func)(void)):** This function registers the display func that is executed when the window needs to be redrawn.

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

- **glutBitmapCharacter():** This function renders the character with ASCII code char at the current raster position using the raster font given by font.
- **glClearColor():** This function sets the color value that is used when clearing the color buffer.
- **glRasterPos2f():** Specifies the raster position for pixel operations.
- **glPushMatrix():** This 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.
- **glLoadIdentity():** This function replaces the current matrix with the identity matrix.
- **glPopMatrix():** This ffunction pops the current matrix stack, replacing the current matrix with the one below it on the stack.
- **glRotatef():** This function computes a matrix that performs a counterclockwise rotation of angle degrees about the vector from the origin through the point (x, y, z).
- **glutSolidSphere():** Renders a sphere centered at the modeling coordinates origin of the specified radius. The sphere is subdivided around the Z axis into slices and along the Z axis into stacks.

4.2 User Defined Functions

- **void streetlight():** This function displays the structure of the streetlight.
- **void background():** This function displays the ground and the sky as the background.
- **void fan1():** This function displays the structure of first fan of the windmill.
- **void fan2():** This function displays the structure of second fan of the windmill.
- **void fan3():** This function displays the structure of third fan of the windmill.
- **void fan4():** This function displays the structure of fourth fan of the windmill.
- **void wires():** This function displays the electrical wires which carries the current generated by the windmill.

- **void house():** This function displays the structure of the house where the electrical energy is being used.
- **void fanpole1():** This function displays the pole of fan1.
- **void fanpole2():** This function displays the pole of fan2.
- **void fanpole3():** This function displays the pole of fan3.
- **void fanhouse():** This function displays the structure of the fan house.
- **void spinclockwise(void):** This function shows the slow movement of windmill in clockwise direction.
- **void anticlockwise(void):** This function shows the slow movement of windmill in anti-clockwise direction.
- **void spinclockwise1(void):** This function shows the fast movement of windmill in clockwise direction.
- **void anticlockwise1(void):** This function shows the fast movement of windmill in anti-clockwise direction.
- **void menu():** This function displays the menu option of the following options:no wind,wind CW,wind ACW,fast wind CW,fast wind ACW,quit.

Chapter 5

Implementation

```
// #include<windows.h>
#include<stdio.h>
#include<GL/glut.h>
static GLfloat spin=360.0;
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;
static int z=0;
GLfloat x=0;
GLfloat y=0;
int m,n;
// Function Prototypes
 ----------------------------------------------------------------------------
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 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);
  glRotatef(spin,0.0,0.0,1.0);
  glTranslatef (8.0,-20.0, -2.0);
  glBegin(GL_TRIANGLES);
  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();
}
//similarly for fan2 and fan3
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();
}
//similarly for fanpole2 and fanpole3
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);
   powerstation();
   wires();
   streetlight();
   fanpole1();
   fanpole2();
   fanpole3();
   fan1();
   fan2();
   fan3();
   fan4();
   glutSwapBuffers();
   glFlush();
   }
}
void spinclockwise(void)// 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 )// 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 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:exit(0);
   }
}
```

Chapter 6

Testing

In unit testing, the program modules that make up the system are tested individually. Unit testing focuses to locate errors in the working modules that are independent to each other. This enables to detect errors in coding and the logic within the module alone. This testing is also used to ensure the integrity of the data stored. The various routines were checked by passing the inputs and the corresponding output is tested.Table 6.1 gives details of validation. Test cases used in the project as follows:

Table 6.1: Test Case Validation

Test Case No.	**Metric**	**Description**	**Observation**
1.	Mouse Function	Displays a menu to select menu for no wind, clockwise rotation, anti-clockwise rotation and quit option.	Results are obtained as expected.
2	Display Function	Displays the various objects like windmill, house, power station and the streetlight.	Results are obtained as expected.
3	Animation	Displays the rotation of windmill either in clockwise or anticlockwise and shows the electricity generated in the household.	Results obtained as expected

Chapter 7

Results & Snapshots

The Figure:7.1 shows the snapshot having the windmill, power station, streetlight,midnight sky, ground and the house.

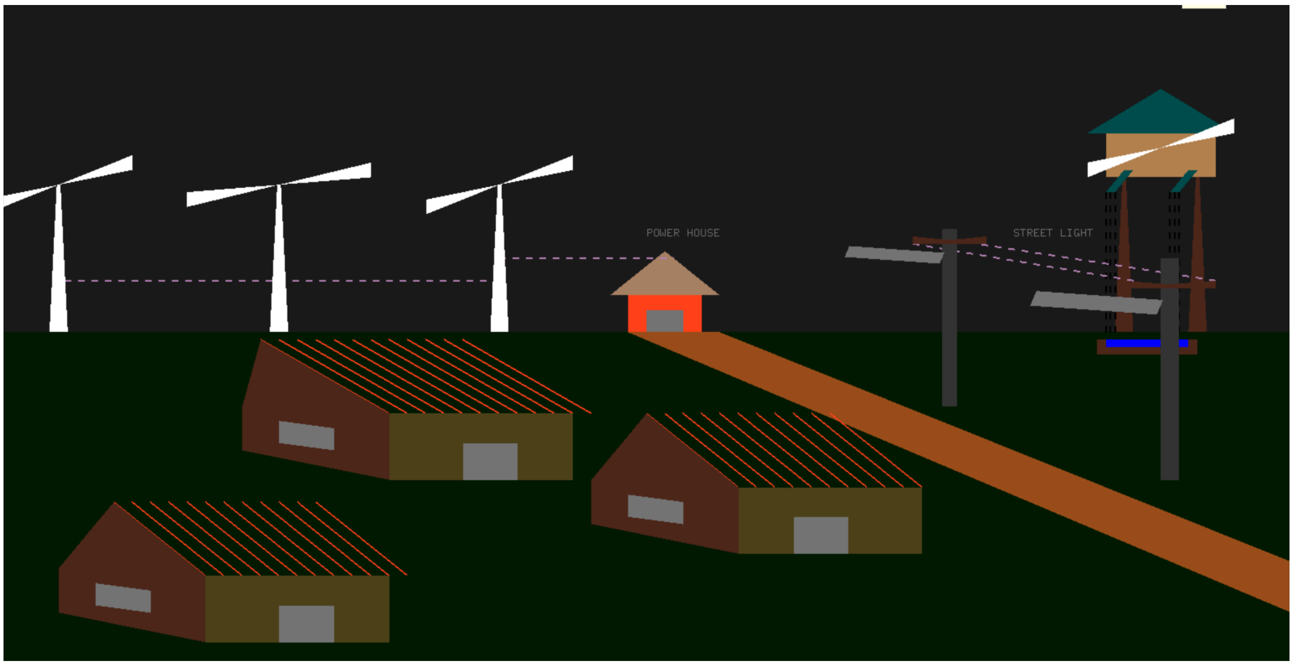

Figure 7.1: Initial screen without any rotation

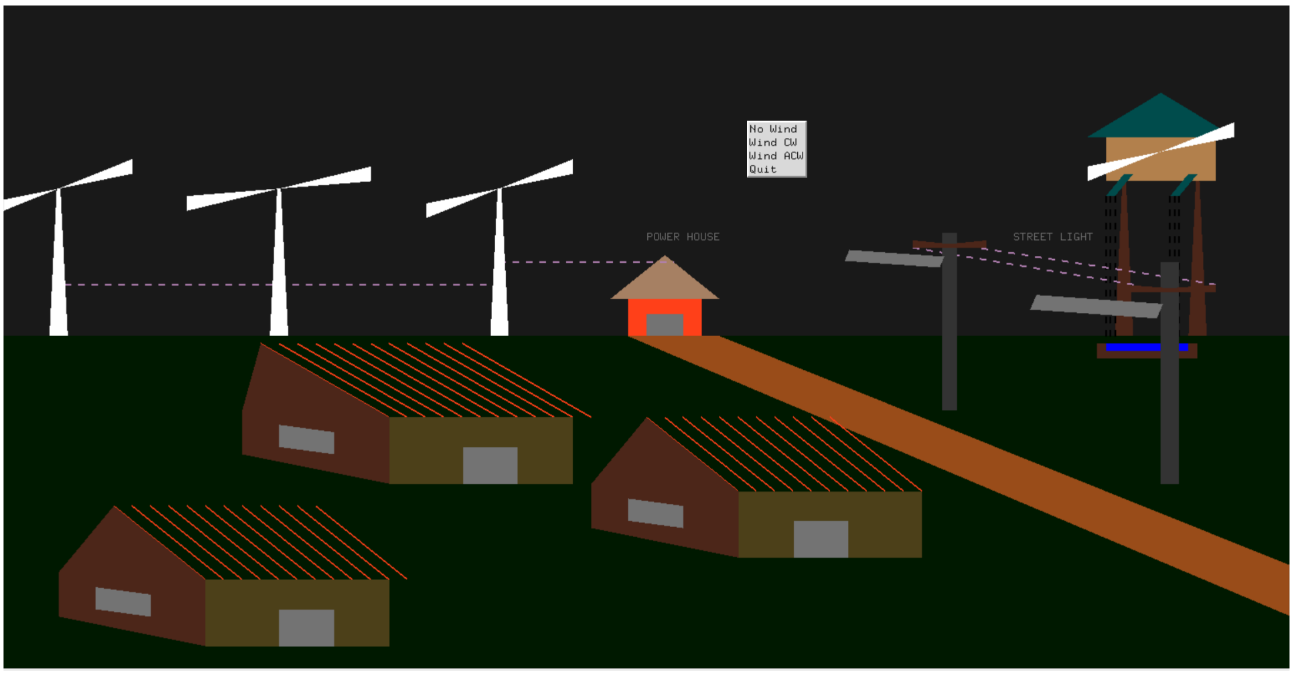

Figure 7.2: Displays menu when right clicked

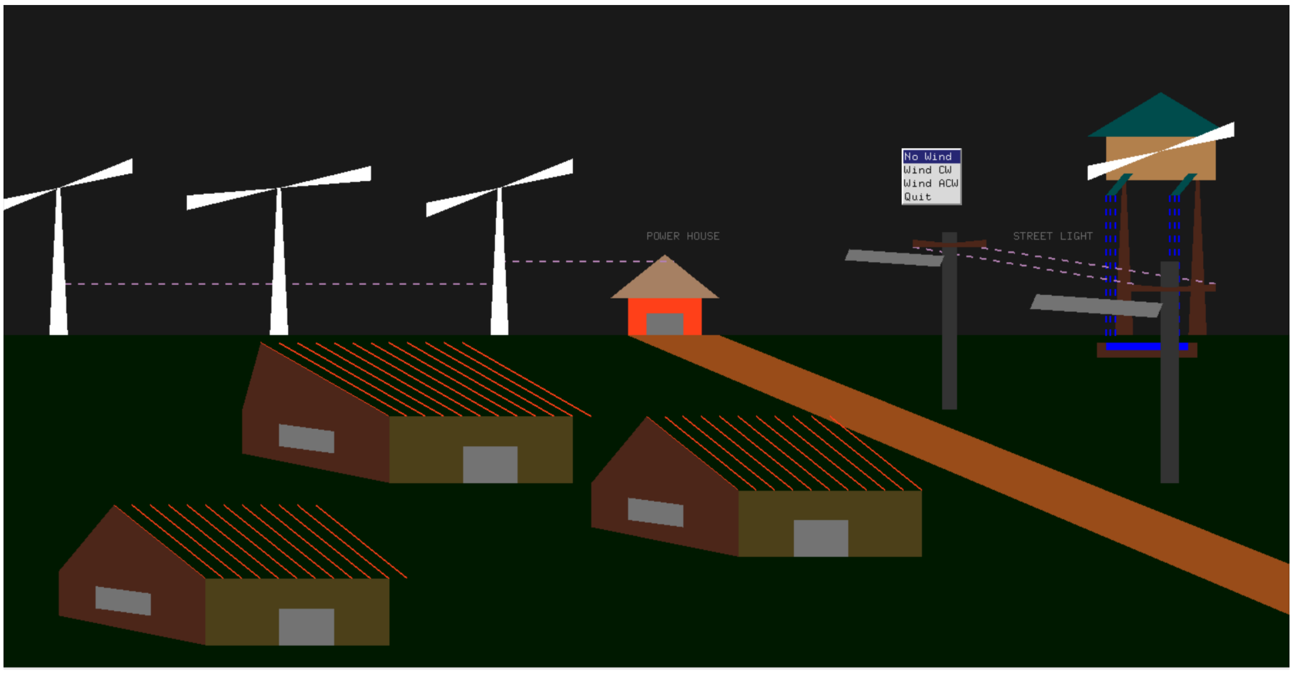

Figure 7.3: Displays snapshot with no wind movement.

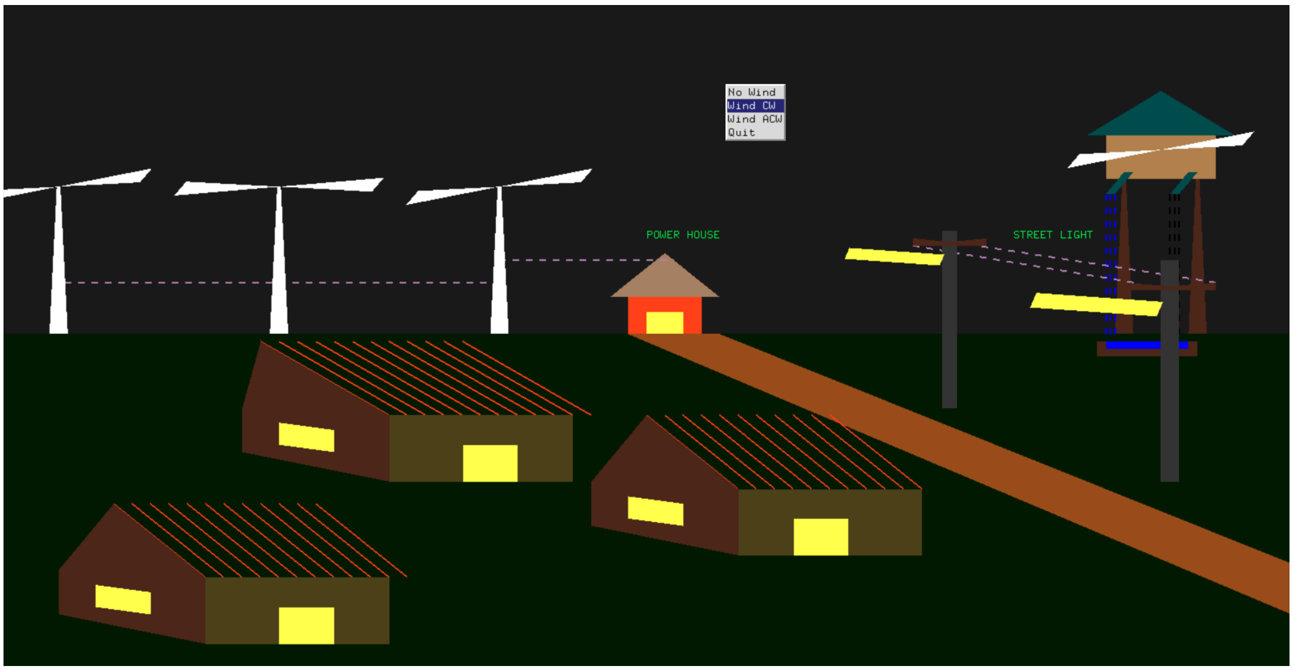

Figure 7.4: Displays snapshot with windmill rotating in clockwise direction.

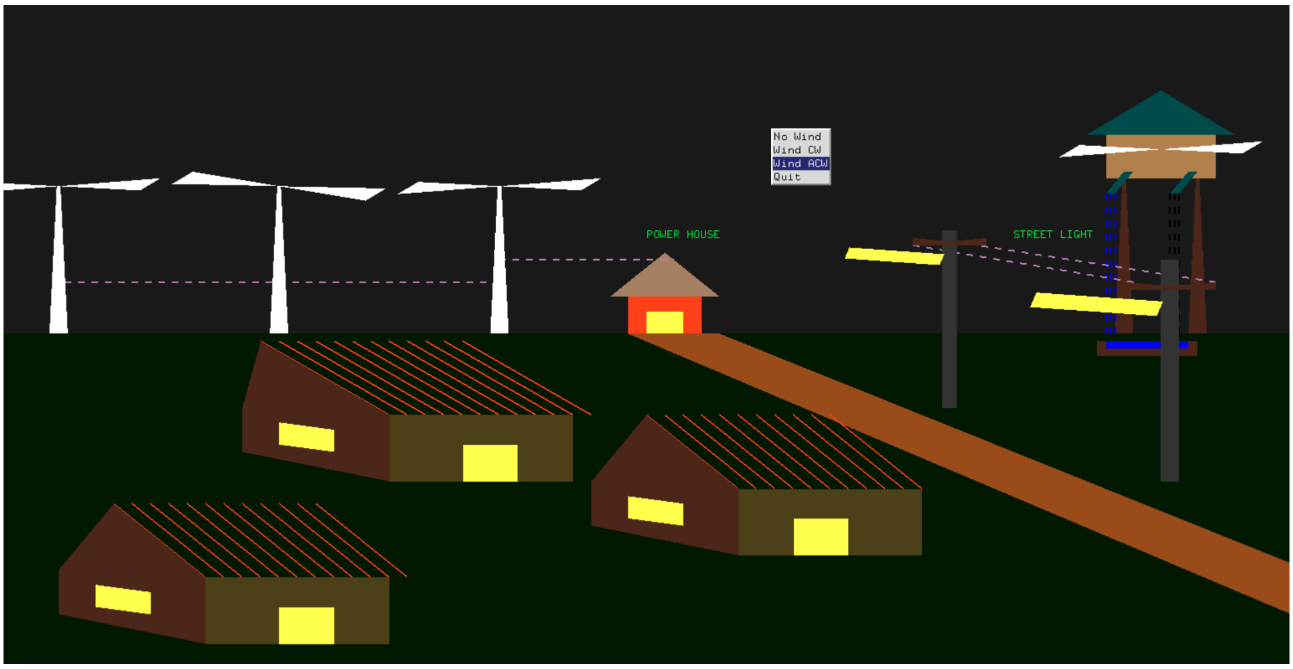

Figure 7.5: Displays snapshot with windmill rotating in anticlockwise direction.

Chapter 8

Conclusion & Future Enhancements

8.1 Conclusion

This project is implemented using OpenGL with C++. The simple OpenGL functions is used to draw the rotating fans in both clockwise and anticlockwise direction for the generation of electricity. Wind power is the conversion of wind energy into a useful form of energy, such as using rotating fans we can generate the electricity which is used for daily routines. This product has been demonstrated to fulfill the requirements. The functionality of all the modules and the module level integration is found to be satisfactory.

This graphics model is very useful, one showing the rotational changes implementation in both clockwise and anticlockwise direction. We have tried our best to make this modeling very realistic, so that the user does not face any trouble when seeing over from any real life to this highly useful graphics model.

8.2 Future Enhancements

This project has been designed using C++using OpenGL which works on the windows platform. The following features can bee incorporated to enhance the project.

- Circular and priority queues can be implemented.
- Different shading effects can be added.
- The concept of transparency and fogging to the objects.

References

[1] Edward Angel, *“Interactive Computer Graphics A Top-Down Approach With OpenGL”* 5th Edition, Addison-Wesley, 2008.

[2] F.S. Hill,*“Computer Graphics Using OpenGL”*, 2nd Edition, Pearson Education, 2001.

[3] James D.Foley, Andries Van Dam, Steven K. Feiner, John F Hughes, *”Computer Graphics”*, Second Edition, Addison-Wesley Professional, August 14,1995.

[4] @online OpenGL Official ,`https://www.opengl.org/`

[5] @online OpenGL Overview , `https://https://www.khronos.org/opengl/`