VISVESVARAYA TECHNOLOGICAL UNIVERSITY

BELGAVI, KARNATAKA -590 018

A Minor Project Report on

"IMPLEMENTATION OF 3D WINDMILL"

Submitted in partial fulfillment for the Computer Graphics and Visualization Laboratory with mini-project course of Sixth Semester of Bachelor of Engineering in Computer Science & Engineering during the academic year 2021-22.

By

Aleen Ulla	**4MN19CS006**
Arun Kumar T K	**4MN20CS401**

|| Under the Guidance of ||

Prof. Bharath Bharadwaj B S

Assistant Professor

Dept. of CS&E

MIT Thandavapura

2021-22

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

MAHARAJA INSTITUTE OF TECHNOLOGY THANDAVAPURA

NH 766, Nanjangud Taluk, Mysuru – 571302

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

MAHARAJA INSTITUTE OF TECHNOLOGY THANDAVAPURA

~~ CERTIFICATE ~~

*Certified that the minor project work entitled "**Implementation of 3D windmill**" is a bonafide work carried out by **Aleen Ulla** (4MN19CS006) & **Arun Kumar T K** (4MN19CS006) for the course **Computer Graphics and Visualization with Mini-Project** with course code **18CSL67** of Sixth Semester in Computer Science & Engineering under Visvesvaraya Technological University, Belagavi during academic year **2021-22**.*

It is certified that all corrections/suggestions indicated for Internal Assignment have been incorporated in the report. The report has been approved as it satisfies the course requirements.

_____	_____
Signature of Lab Staff In-Charge	Signature of the HoD
Prof. Bharath Bharadwaj B S	**Dr. Ranjit K N**
Assistant Professor	Associate Professor & HoD
Dept. of CS&E	Dept. of CS&E
MIT Thandavapura	MIT Thandavapura

External viva

Name of the Examiners	**Signature with date**

1)…….

2)…….

~~~~ ACKNOWLEDGEMENT ~~~~

It is the time to acknowledge all those who have extended their guidance, inspiration and their whole hearted co-operation all along our project work.

We are grateful to **Dr. Y T Krishne Gowda**, Principal, MIT Thandavapura, **Dr. H K Chethan,** Professor and Mentor, CS&E, MIT Thandavapura and also **Dr. Ranjit K N**, Associate Professor and Head, CS&E, MIT Thandavapura for having provided us academic environment which nurtured our practical skills contributing to the success of our project.

We wish to place a deep sense of gratitude to all Teaching and Non-Teaching staffs of Computer Science and Engineering Department for whole-hearted guidance and constant support without which this endeavour would not have been possible.

Our gratitude will not be complete without thanking our parents and also our friends, who have been a constant source of support and aspirations.

NAME	SIGNATURE
Aleen Ulla	
Arun Kumar T K	

~~~ ABSTRACT ~~~

In this project, The Windmill output will appear on the screen wherein the user can run the fan of the Windmill. He can also choose the color of the fan blades with or without the texturing type for the fan blades. The color of the fan is available to the users when they click the right mouse button. The different colors which the user can choose from are organized into various menu selections. Similarly, texture for the fan blades can also be selected through two submenu selections : ENABLE and DISABLE . The start and stop function of the Windmill's fan can be controlled using either the menus displayed under the right mouse button function or by using the keyboard function. Also, an aircraft has been implemented such that it travels at a constant rate from the extreme left corner to the extreme right corner. However, the user can choose to use the keyboard function to fast-forward the aircraft or rewind it back into place. This has been made available for the user through the glutKeyboardFunc. Hence, the program is made interactive using both mouse functionalities and keyboard functionalities.

~~~~~ CONTENTS ~~~~~

SL. No.		Index	Page No.
1		**INTRODUCTION**	**1-4**
	1.1	Aim of the Project	3
	1.2	Overview of the Project	4
	1.3	Outcome of the Project	4
2		**DESIGN AND IMPLEMENTATION**	**5-9**
	2.1	Algorithm	5
	2.2	Flowchart	6
	2.3	OpenGL API's Used with Description	7
3		**RESULT ANALYSIS**	**10-13**
	3.1	Snapshots	10
	3.2	Discussion	13
4		**CONCLUSION AND FUTURE WORK**	**14**
	4.1	Conclusion	14
	4.2	Future Enhancement	14
		REFERENCES	
		APPENDIX A – Source Code	**A1-A18**

~~~~~ LIST OF FIGURES ~~~~~

SL. No.	Index	Page No.
1.1	Library organization of OpenGL	2
2.2	Flowchart for 3D Windmill	6
3.1	Initial preview to the viewer when he starts the execution	10
3.2	Displays the function window lighting	10
3.3	This displays the function option	11
3.4	This displays the texture option	11
3.5	Sets fan color to blue	12
3.6	Sets fan color to brown	12

CHAPTER – 1

INTRODUCTION

Graphics provides one of the most natural means of communicating with a computer,since our highly developed 2D or 3D pattern-recognition abilities allow us to perceive and process pictorial data rapidly. Computers have become a powerful medium for the rapid and economical production of pictures. Graphics provide a so natural means of communicating with the computer that they have become widespread. Interactive graphics is the most important means of producing pictures since the invention of photography and television. We can make pictures of not only the real world objects but also of abstract objects such as mathematical surfaces on 4D and of data that have no inherent geometry. A computer graphics system is a computer system with all the components of the general purpose computer system. There are five major elements in system: input devices, processor, memory, frame buffer, output devices.

A broad classification of major subfields in Computer Graphics might be:

1. **Geometry:** Studies ways to represent and process surfaces.
2. **Animation**: Studies ways to represent and manipulate motion.
3. **Rendering:** Studies algorithms to reproduce light transport.
4. **Imaging:** Studies image acquisition or image editing.

➢ **Computer graphics**: Computer graphics is concerned with all aspects of producing pictures and images using a computer.

➢ **Translation**: Translation is an operation that displaces points by a fixed size in a given direction.

➢ **Rotation**: Three features of transformation extended to other rotations

AREAS OF APPLICATION OF COMPUTER GRAPHICS

➢ User interface and Process control.

➢ Plotting of graphs and charts.

➢ Simulation and Animation.

➢ Computer aided Drafting and designs.

➢ Cartography.

➢ Office automation and Desktop publishing

OpenGL Technology

OpenGL is the premier environment for developing portable, interactive 2D and 3D graphics applications. Since its introduction in 1992, OpenGL has become the industry's most widely used and supported 2D and 3D graphics application programming interface (API), bringing thousands of applications to a wide variety of computer platforms.

OpenGL fosters innovation and speeds application development by incorporating a broad set of rendering, texture mapping, special effects, and other powerful visualization functions. Developers can leverage the power of OpenGL across all popular desktop and workstation platforms, ensuring wide application deployment.

OpenGL Available Everywhere: Supported on all UNIX® workstations, and shipped standard with every Windows 95/98/2000/NT and MacOS PC, no other graphics API operates on a wider range of hardware platforms and software environments.

OpenGL runs on every major operating system including Mac OS, OS/2, UNIX, Windows, Windows NT, Linux, Open Step, and BeOS; it also works with every major windowing system, including Win32, MacOS, Presentation Manager, and X-Window System.

The OpenGL interface

Our application will be designed to access OpenGL directly through functions in three libraries namely: gl, glu, glut.

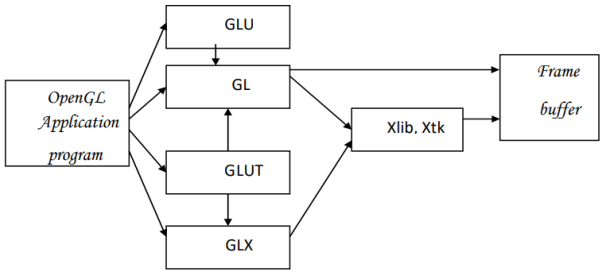

Fig 1.1: Library organization of OpenGL

1.1 Aim

Computer graphics are graphics created using computers and the representation of image data by a computer specifically with help from specialized graphic hardware and software.

The interaction and understanding of computers and interpretation of data has been made easier because of computer graphics. Computer graphic development has had a significant impact on many types of media and have revolutionized animation, movies and the video game industry.

Computer graphics is widespread today. Computer imagery is found on television, in newspapers, for example in weather reports, or for example 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.

Computer generated imagery can be categorized into several different types: two dimensional (2D), three dimensional (3D), and animated graphics. As technology has improved, 3D computer graphics have become more common, but 2D computer graphics are still widely used. 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 (time) component".

OpenGL Utility Toolkit (GLUT) was developed by Mark Kilgard, it Hides the complexities of differing window system APIs, Default user interface for class projects, Glut routines have prefix glut, E.g. - glutCreateWindow().

1.2 Overview

In this program, we are controlling the functions of the fan using the menu, which can be done by clicking the RIGHT MOUSE BUTTON. If we click on the 1st option FAN COLOR, it opens the sub-menu in which user can select the color of his liking. The user can choose the color of his choice from the menu. In the menu, there is the option of default color(White), Brown color, Red color and Blue color of which selection can be made. Similarly, if we click on the 2nd option FAN TEXTURE, it opens the sub-menu in which user can disable or enable the texture to the wings of the fan.

1.3 Outcome

The 3rd option OPERATIONS opens up the sub-menu for fan functions such as start, stop,speed1, speed2, speed3, speed4, and speed5. The 4th option is to QUIT and come out of the executing program. We can control the fan functions also by using the keys on the keyboard. The keys which are used to create the speed differential are:-

- “s”: - this key is used to start the rotation of the fan initially.
- “q”: - this key is used to stop the fan rotation.
- “1”: - this key is used to rotate the fan at speed1.
- “2”: - this key is used to rotate the fan at speed2.
- “3”: - this key is used to rotate the fan at speed3.
- “4”: - this key is used to rotate the fan at speed4.
- “5”: - this key is used to rotate the fan at speed5.

CHAPTER – 2

DESIGN AND IMPLEMENTATION

2.1 Algorithm

Step1:- Start

Step2:- Create scenery

Step3:- if key='s' , then windmill movement starts

Step4:- if key='q' , then windmill movement stops

Step5:- if key='1' , then windmill speed is set to slow

Step6:- if key='2' , then windmill speed is set to normal

Step7:- if key='3' , then windmill speed is set to medium

Step8:- if key='4' , then windmill speed is set to fast

Step9:- if key='5' , then windmill speed is set to high speed

Step10:- Stop

2.2 Flowchart

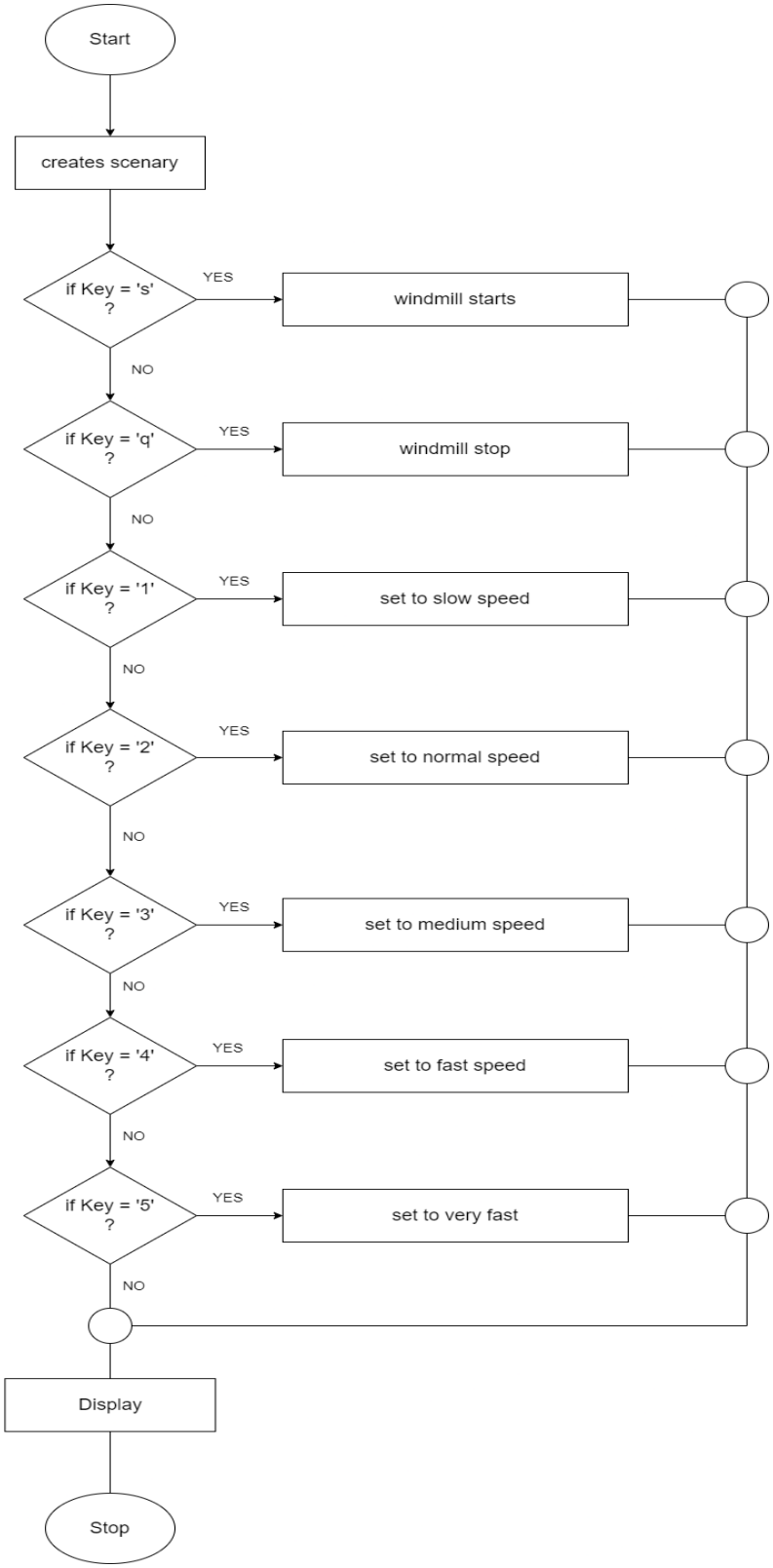

Fig 2.2 Flowchart for 3D Windmill

2.3 OpenGL API's Used with Description

OpenGL has become a widely accepted standard for developing graphics application. Most of our applications will be designed to access OpenGL directly through functions in three libraries. Functions in main GL library have names that begin with the letters gl and are stored in a library usually referred to as GL.

Function Name	Description
glutInitDisplayMode	sets the initial display mode. **Declaration:** void glutInitDisplayMode (unsigned int mode); **Remarks:** 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.
glutInitWindowposition	set the initial window position. **Declaration:** void glutInitWindowPosition(int x, int y); x: Window X location in pixels. y: Window Y location in pixels.
glutInitWindowSize	set the initial window size. **Declaration:** void glutInitWindowSize(int width,int height); width: Width in pixels height: Height in pixels.
glutCreateWindow	set the title to graphics window. **Declaration:** Int glutCreateWindow(char *title);

	Remarks: This function creates a window on the display. The string title can be used to label the window.The integer value returned can be used to set the current window when multiple windows are created.
glutDisplayFunc	**Declaration**: void glutDisplayFunc(void(*func)void)); **Remarks:** This function registers the display function that isexecuted when the window needs to be redrawn
glClear	**Declaration:** void glClear(); **Remarks**: This function clears the particular buffer.
glClearColor	**Declaration:** void glClearColor(GLfloat red, GLfloat green, Glfloat blue, Glfloat alpha); **Remarks:** This function sets the color value that is used when clearing the color buffer.
glEnd	**Declaration:** void glEnd(); **Remarks:** This function is used in conjunction with glBegin to delimit the vertices of an opengl primitive.
glMatrixMode	**Declaration:** void glMatrixMode(GLenum mode); **Remarks:** This function specifies which matrix will be affected by subsequent transformations mode can be GL_MODELVIEW,GL_PROJECTION or GL_TEXTURE..

glOrtho	**Declaration:** void glOrtho(GLdouble left, GLdouble right, GLdouble bottom,GLdouble top,GLdouble near,GLdouble far); **Remarks:** It defines an orthographic viewing volume with all parameters measured from the center of the projection plane.
glIint	**Declaration:** Void glutInit(int *argc, char **argv). **Remarks:** To start through graphics system, we must first call glutInit (),glutInit will initialize the GLUT library and negotiate a session with the window system. During this process, glutInit may cause the termination of the GLUT program with an error message to the user if GLUT cannot be properly initialized.
glutAddMenuEntry	**Declaration:** Void glutAddMenuEntry(char *name,int value); **Remarks:** Adds a menu entry to the bottom of the current menu.The String name will be displayed for the newly added menu entry.An identifier is used to represent each option.
glutCreateMenu	**Declaration:** int glutCreateMenu(void (*func)(int value)); **Remarks:** It creates a new pop-up menu and returns a uniquesmall integer identifier.The range of the allocated identifiers starts at 1.

CHAPTER – 3

RESULT ANALYSIS

3.1 Snapshots

Fig 3.1: Initial preview to the viewer when he starts the execution

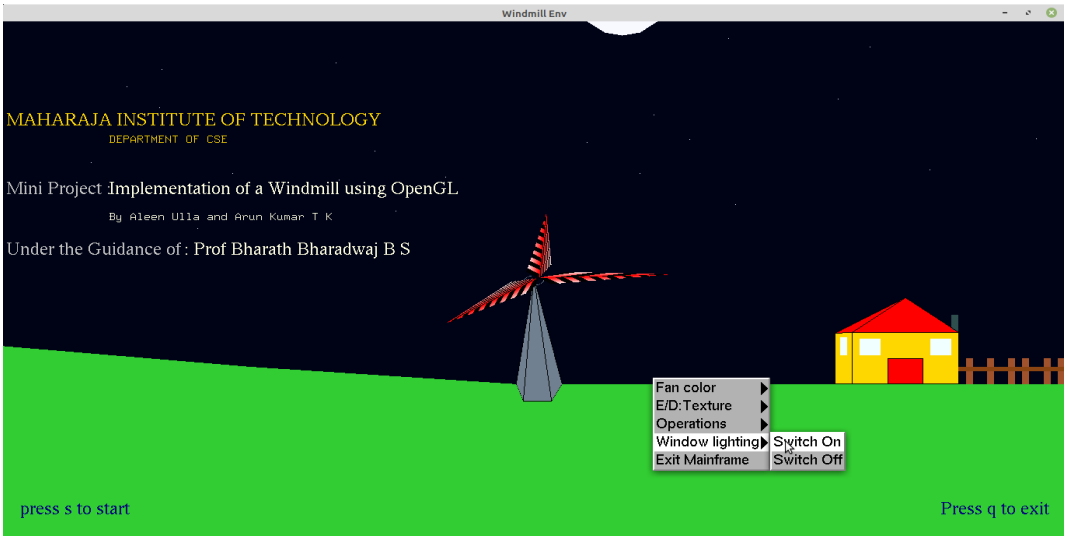

Fig 3.2: Displays the function window lighting

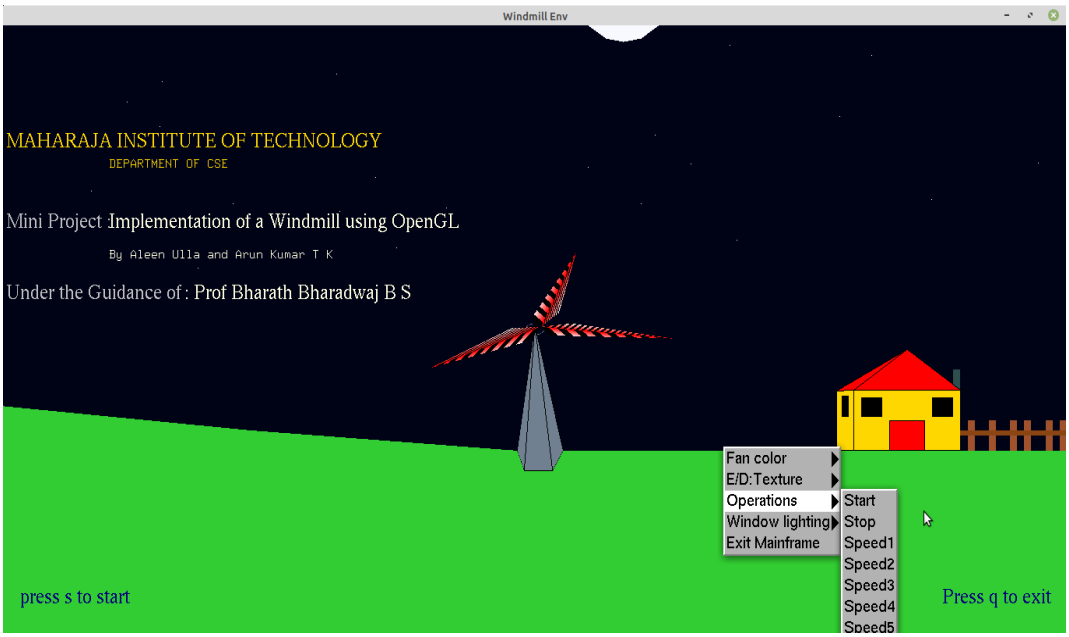

Fig 3.3: This displays the function option.

Fig 3.4: This displays the texture option.

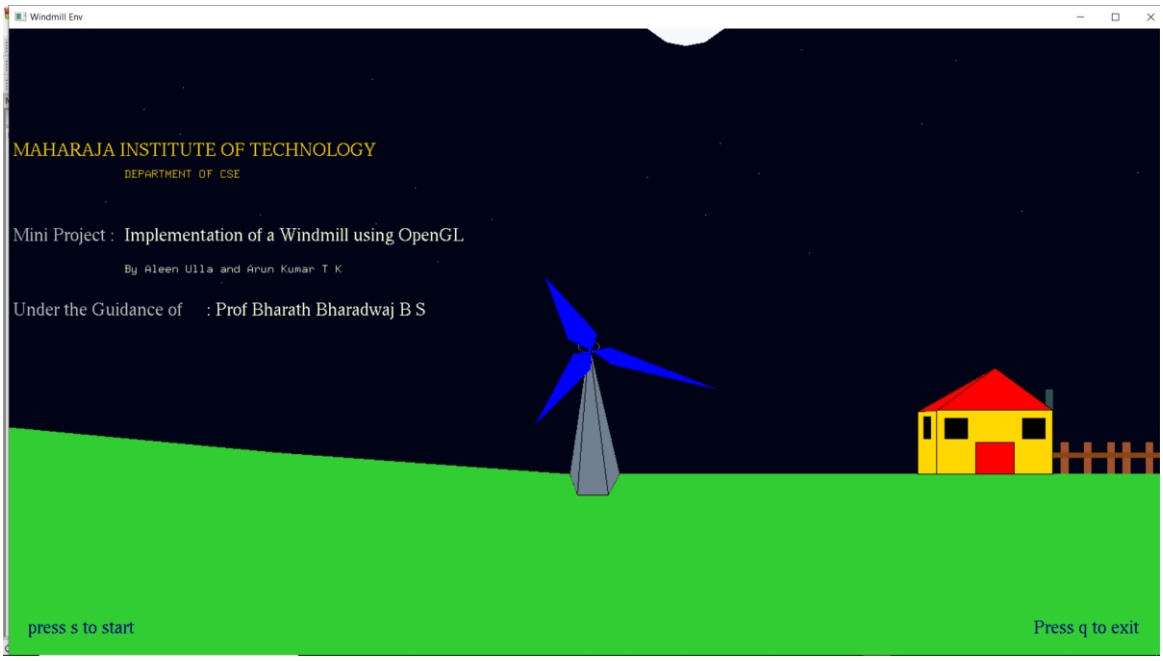

Fig 3.5: Sets fan color to blue

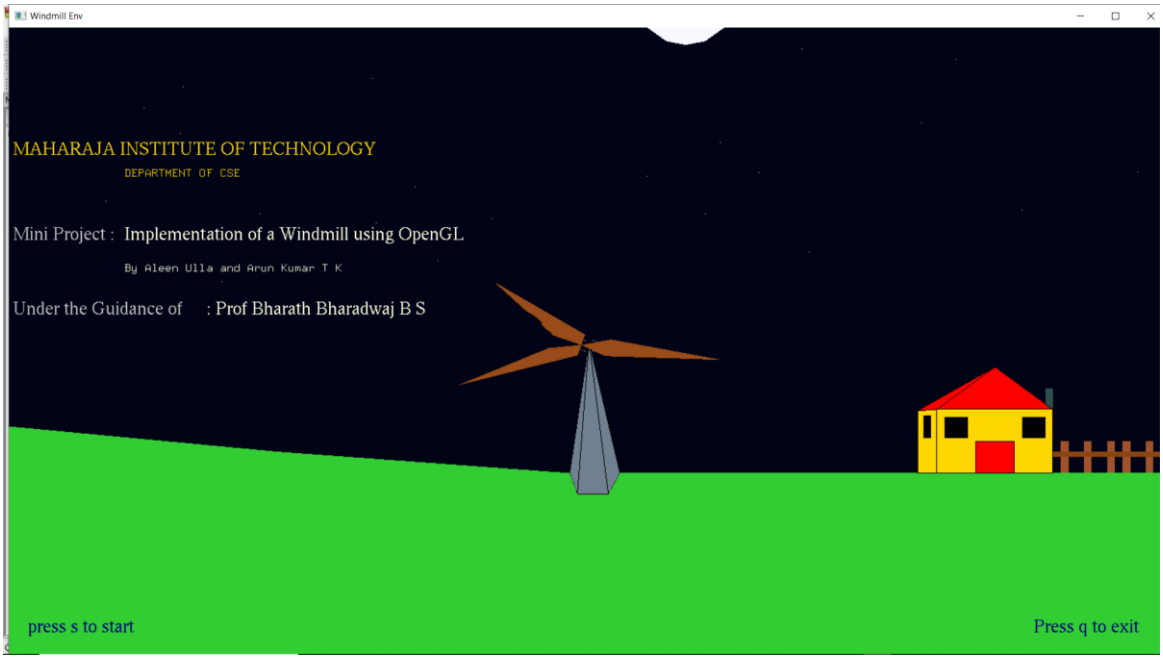

Fig 3.6: Sets fan color to brown

3.2 Discussion

The main aim of the project is to show the reflection of cartoons using polygons. The project is simulated using the software called OpenGL. Here we are displaying the movements of cartoons using various polygons that is sphere, cone, cube etc. The lighting effects and rotation operations are carried out. This project has been a successful learning experience to practically use the functions defined in OpenGL and to understand a variety of features and options present in it. This project clearly demonstrated the richness of graphics library and its ability to make complex objects in a very simple way. We intended to make maximum use of the OpenGL functions which we have done to our satisfaction.

CHAPTER – 4

CONCLUSION AND FUTURE WORK

4.1 Conclusion

The Windmill Graphics package has been developed Using OpenGL. The illustration of graphical principles and OpenGL features are included and application program is efficiently developed.

The aim in developing this program was to design a simple program using Open GL application software by applying the skills we learnt in class, and in doing so, to understand the algorithms and the techniques underlying interactive graphics better.

The designed program will incorporate all the basic properties that a simple program must possess. The program is user friendly as the only skill required in executing this program is the knowledge of graphics.

The main idea of the program is to implement the Windmill by creating the wings and applying rotation to them and giving different speeds of rotation. Thus , executing the program successfully.

4.2 Future Enhancement

1. We can use the program to implement the blueprint for modelling similar structures.
2. We can try to improve the quality of the objects present.
3. We can use texture mapping to give a more realistic effect.
4. Other models can be derived from this model.
5. More objects can be used to show the 3d model of an entire city.

REFERENCES

1. The Red Book –OpenGL Programming Guide,6th edition.
2. Rost , Randi J. : OpenGL Shading Language, Addison-Wesley
3. Interactive Computer Graphics-A Top Down Approach Using OpenGL, Edward Angel, Pearson-5th edition.
4. www.stackoverflow.com
5. www.opengl.org
6. www.khronos.org
7. www.wikipedia.org

APPENDIX

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

int global = 0,tex1=0;
float pos,y=0;
int flag=0;

char s[30]="press s to start";
char s1[100]="MAHARAJA INSTITUTE OF TECHNOLOGY";
char s2[100]="DEPARTMENT OF CSE";
char s3[100]="Mini Project : ";
char s4[100]="Implementation of a Windmill using OpenGL";
char s5[100]="By Aleen Ulla and Arun Kumar T K";
char s6[100]="Under the Guidance of ";
char s7[50]="Press q to exit";
char s8[75]=" : Prof Bharath Bharadwaj B S";

void myinit()
{
glClearColor(0.0, 0.012, 0.086,1.0);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluOrtho2D(-150,150,-150,150);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
}

static GLfloat theta[]={0.0,0.0,1.0};
static GLint axis=2;

GLfloat *make_texture(int maxs, int maxt)
{
int s, t;
static GLfloat *texture;

texture = (GLfloat *)malloc(maxs * maxt * sizeof(GLfloat));
for(t = 0; t < maxt; t++) {
for(s = 0; s < maxs; s++) {
texture[s + maxs * t] = ((s >> 4) & 0x1) ^ ((t >> 4) & 0x1);
}
}
return texture;
}

void fan()
{
glColor3f(0.439,0.502,0.565); /*circle holding the fan*/
glutSolidSphere(2.8,500,100);

glBegin(GL_POLYGON);
glColor3f(1.0,1.0,1.0);
glVertex2f(0.0,2.0);
glVertex2f(3.0,10.0);
glVertex2f(25.0,30.0);
glVertex2f(5.0,2.0);

glVertex2f(0.0,2.0);
glVertex2f(-3.0,10.0);
glVertex2f(-25.0,30.0);
glVertex2f(-5.0,2.0);
```

```
glVertex2f(0.0,2.0);
glVertex2f(-4.0,-5.0);
glVertex2f(0.0,-35.0);
glVertex2f(4.0,-5.0);
glEnd();

if(global==1 && tex1==1) //Default-White color texture
{
glEnable(GL_TEXTURE_2D);
glBegin(GL_POLYGON);
glColor3f(1.0,1.0,1.0);
glTexCoord2i(0,0);
glVertex2f(0.0,2.0);
glColor3f(0.0,0.0,0.0);
glTexCoord2i(0,1);
glVertex2f(3.0,10.0);
glColor3f(1.0,1.0,1.0);
glTexCoord2i(0,1);
glVertex2f(25.0,30.0);
glColor3f(0.0,0.0,0.0);
glTexCoord2i(0,0);
glVertex2f(5.0,2.0);

glColor3f(1.0,1.0,1.0);
glTexCoord2i(0,0);
glVertex2f(0.0,2.0);
glColor3f(0.0,0.0,0.0);
glTexCoord2i(0,1);
glVertex2f(-3.0,10.0);
glColor3f(1.0,1.0,1.0);
glTexCoord2i(0,1);
glVertex2f(-25.0,30.0);
glColor3f(0.0,0.0,0.0);
glTexCoord2i(0,0);
glVertex2f(-5.0,2.0);

glColor3f(1.0,1.0,1.0);
glTexCoord2i(0,0);
glVertex2f(0.0,2.0);
glColor3f(0.0,0.0,0.0);
glTexCoord2i(0,1);
glVertex2f(-4.0,-5.0);
glColor3f(1.0,1.0,1.0);
glTexCoord2i(0,1);
glVertex2f(0.0,-35.0);
glColor3f(0.0,0.0,0.0);
glTexCoord2i(0,0);
glVertex2f(4.0,-5.0);
glEnd();
glDisable(GL_TEXTURE_2D);
}

if(global==2 && tex1==1) //Brown texture color
{
glEnable(GL_TEXTURE_2D);
glBegin(GL_POLYGON);
glColor3f(0.6,0.3,0.1);
glTexCoord2i(0,0);
glVertex2f(0.0,2.0);
glColor3f(1.0,1.0,1.0);
glTexCoord2i(0,1);
glVertex2f(3.0,10.0);
glColor3f(0.6,0.3,0.1);
```

```
glTexCoord2i(0,1);
glVertex2f(25.0,30.0);
glColor3f(1.0,1.0,1.0);
glTexCoord2i(0,0);
glVertex2f(5.0,2.0);

glColor3f(0.6,0.3,0.1);
glTexCoord2i(0,0);
glVertex2f(0.0,2.0);
glColor3f(1.0,1.0,1.0);
glTexCoord2i(0,1);
glVertex2f(-3.0,10.0);
glColor3f(0.6,0.3,0.1);
glTexCoord2i(0,1);
glVertex2f(-25.0,30.0);
glColor3f(1.0,1.0,1.0);
glTexCoord2i(0,0);
glVertex2f(-5.0,2.0);

glColor3f(0.6,0.3,0.1);
glTexCoord2i(0,0);
glVertex2f(0.0,2.0);
glColor3f(1.0,1.0,1.0);
glTexCoord2i(0,1);
glVertex2f(-4.0,-5.0);
glColor3f(0.6,0.3,0.1);
glTexCoord2i(0,1);
glVertex2f(0.0,-35.0);
glColor3f(1.0,1.0,1.0);
glTexCoord2i(0,0);
glVertex2f(4.0,-5.0);
glEnd();
glDisable(GL_TEXTURE_2D);
}

if(global==3 && tex1==1) //Red color texture
{
glEnable(GL_TEXTURE_2D);
glBegin(GL_POLYGON);
glColor3f(1.0,0.0,0.0);
glTexCoord2i(0,0);
glVertex2f(0.0,2.0);
glColor3f(1.0,1.0,1.0);
glTexCoord2i(0,1);
glVertex2f(3.0,10.0);
glColor3f(1.0,0.0,0.0);
glTexCoord2i(0,1);
glVertex2f(25.0,30.0);
glColor3f(1.0,1.0,1.0);
glTexCoord2i(0,0);
glVertex2f(5.0,2.0);

glColor3f(1.0,0.0,0.0);
glTexCoord2i(0,0);
glVertex2f(0.0,2.0);
glColor3f(1.0,1.0,1.0);
glTexCoord2i(0,1);
glVertex2f(-3.0,10.0);
glColor3f(1.0,0.0,0.0);
glTexCoord2i(0,1);
glVertex2f(-25.0,30.0);
glColor3f(1.0,1.0,1.0);
glTexCoord2i(0,0);
glVertex2f(-5.0,2.0);
```

```
glColor3f(1.0,0.0,0.0);
glTexCoord2i(0,0);
glVertex2f(0.0,2.0);
glColor3f(1.0,1.0,1.0);
glTexCoord2i(0,1);
glVertex2f(-4.0,-5.0);
glColor3f(1.0,0.0,0.0);
glTexCoord2i(0,1);
glVertex2f(0.0,-35.0);
glColor3f(1.0,1.0,1.0);
glTexCoord2i(0,0);
glVertex2f(4.0,-5.0);
glEnd();
glDisable(GL_TEXTURE_2D);
}

if(global==4 && tex1==1) //Blue color texture
{
glEnable(GL_TEXTURE_2D);
glBegin(GL_POLYGON);
glColor3f(0.0,0.0,1.0);
glTexCoord2i(0,0);
glVertex2f(0.0,2.0);
glColor3f(1.0,1.0,1.0);
glTexCoord2i(0,1);
glVertex2f(3.0,10.0);
glColor3f(0.0,0.0,1.0);
glTexCoord2i(0,1);
glVertex2f(25.0,30.0);
glColor3f(1.0,1.0,1.0);
glTexCoord2i(0,0);
glVertex2f(5.0,2.0);

glColor3f(0.0,0.0,1.0);
glTexCoord2i(0,0);
glVertex2f(0.0,2.0);
glColor3f(1.0,1.0,1.0);
glTexCoord2i(0,1);
glVertex2f(-3.0,10.0);
glColor3f(0.0,0.0,1.0);
glTexCoord2i(0,1);
glVertex2f(-25.0,30.0);
glColor3f(1.0,1.0,1.0);
glTexCoord2i(0,0);
glVertex2f(-5.0,2.0);

glColor3f(0.0,0.0,1.0);
glTexCoord2i(0,0);
glVertex2f(0.0,2.0);
glColor3f(1.0,1.0,1.0);
glTexCoord2i(0,1);
glVertex2f(-4.0,-5.0);
glColor3f(0.0,0.0,1.0);
glTexCoord2i(0,1);
glVertex2f(0.0,-35.0);
glColor3f(1.0,1.0,1.0);
glTexCoord2i(0,0);
glVertex2f(4.0,-5.0);
glEnd();
glDisable(GL_TEXTURE_2D);
}

if(global==1 && tex1==0) //Default-White color without texture
```

```
{
glBegin(GL_POLYGON);
glColor3f(1.0,1.0,1.0);
glVertex2f(0.0,2.0);
glVertex2f(3.0,10.0);
glVertex2f(25.0,30.0);
glVertex2f(5.0,2.0);

glVertex2f(0.0,2.0);
glVertex2f(-3.0,10.0);
glVertex2f(-25.0,30.0);
glVertex2f(-5.0,2.0);

glVertex2f(0.0,2.0);
glVertex2f(-4.0,-5.0);
glVertex2f(0.0,-35.0);
glVertex2f(4.0,-5.0);
glEnd();
}

if(global==2 && tex1==0) //Brown color without texture
{
glBegin(GL_POLYGON);
glColor3f(0.6,0.3,0.1);
glVertex2f(0.0,2.0);
glVertex2f(3.0,10.0);
glVertex2f(25.0,30.0);
glVertex2f(5.0,2.0);

glVertex2f(0.0,2.0);
glVertex2f(-3.0,10.0);
glVertex2f(-25.0,30.0);
glVertex2f(-5.0,2.0);

glVertex2f(0.0,2.0);
glVertex2f(-4.0,-5.0);
glVertex2f(0.0,-35.0);
glVertex2f(4.0,-5.0);
glEnd();
}

if(global==3 && tex1==0) //Red color without texture
{
glBegin(GL_POLYGON);
glColor3f(1.0,0.0,0.0);
glVertex2f(0.0,2.0);
glVertex2f(3.0,10.0);
glVertex2f(25.0,30.0);
glVertex2f(5.0,2.0);

glVertex2f(0.0,2.0);
glVertex2f(-3.0,10.0);
glVertex2f(-25.0,30.0);
glVertex2f(-5.0,2.0);

glVertex2f(0.0,2.0);
glVertex2f(-4.0,-5.0);
glVertex2f(0.0,-35.0);
glVertex2f(4.0,-5.0);
glEnd();
}

if(global==4 && tex1==0) //Blue color without texture
{
```

```
glBegin(GL_POLYGON);
glColor3f(0.0,0.0,1.0);
glVertex2f(0.0,2.0);
glVertex2f(3.0,10.0);
glVertex2f(25.0,30.0);
glVertex2f(5.0,2.0);

glVertex2f(0.0,2.0);
glVertex2f(-3.0,10.0);
glVertex2f(-25.0,30.0);
glVertex2f(-5.0,2.0);

glVertex2f(0.0,2.0);
glVertex2f(-4.0,-5.0);
glVertex2f(0.0,-35.0);
glVertex2f(4.0,-5.0);
glEnd();
}
}

void grass1()
{
glBegin(GL_POLYGON);
glColor3f(0.196, 0.804, 0.196);
glVertex2f(-3.0,-70.0);
glVertex2f(-150.0,-70.0);
glVertex2f(-150.0,-150.0);
glVertex2f(-3.0,-150);
glEnd();

glBegin(GL_POLYGON);
glVertex2f(-3.0,-70.0);
glVertex2f(150.0,-70);
glVertex2f(150.0,-150.0);
glVertex2f(-3.0,-150.0);
glEnd();

glBegin(GL_POLYGON);
glVertex2f(8.0,-60.0);
glVertex2f(150.0,-60.0);
glVertex2f(150.0,-150.0);
glVertex2f(5.0,-150.0);
glEnd();

glBegin(GL_POLYGON);
glVertex2f(8.0,-60.0);
glVertex2f(8.0,-70.0);
glVertex2f(5.0,-70.0);
glEnd();

glBegin(GL_POLYGON);
glVertex2f(-5.0,-60.0);
glVertex2f(-150.0,-60.0);
glVertex2f(-150.0,-150.0);
glVertex2f(-5.0,-150.0);
glEnd();

glBegin(GL_POLYGON);
glVertex2f(-5.0,-60.0);
glVertex2f(-5.0,-70.0);
glVertex2f(-3.0,-70.0);
glEnd();
}
```

```
void grasselev()
{
glColor3f(0.196, 0.804, 0.196);
glBegin(GL_POLYGON);
glVertex2f(-5.0,-60.0);
glVertex2f(-80.0,-50.0);
glVertex2f(-80.0,-60.0);
glEnd();

glBegin(GL_POLYGON);
glVertex2f(-80.0,-50.0);
glVertex2f(-150.0,-38.0);
glVertex2f(-150.0,-60.0);
glVertex2f(-80.0,-60.0);
glEnd();
}

void drawhouse()
{

glBegin(GL_POLYGON); //square base
glColor3f(1.000, 0.843, 0.000);
glVertex2f(90.0,-60.0);
glVertex2f(90.0,-30.0);
glVertex2f(120.0,-30.0);
glVertex2f(120.0,-60.0);
glEnd();

glBegin(GL_POLYGON); //triangular top
glColor3f(1.0,0.0,0.0);
glVertex2f(90.0,-30.0);
glVertex2f(105.0,-10.0);
glVertex2f(120.0,-30.0);
glEnd();

glBegin(GL_POLYGON); //door
glColor3f(2.0,0.0,0.0);
glVertex2f(100.0,-60.0);
glVertex2f(100.0,-45.0);
glVertex2f(110.0,-45.0);
glVertex2f(110.0,-60.0);
glEnd();

glBegin(GL_POLYGON); //chimney
glColor3f(0.184, 0.310, 0.310);
glVertex2f(120.0,-30.0);
glVertex2f(120.0,-20.0);
glVertex2f(118.0,-20.0);
glVertex2f(118.0,-28.0);
glEnd();

/*House side */
glBegin(GL_POLYGON); //side wall
glColor3f(1.000, 0.843, 0.000);
glVertex2f(85.0,-30.6);
glVertex2f(90.0,-30.0);
glVertex2f(90.0,-60.0);
glVertex2f(85.0,-60.0);
glEnd();

glBegin(GL_POLYGON); //roof
glColor3f(1.0,0.0,0.0);
glVertex2f(85.0,-30.6);
glVertex2f(105.0,-10.0);
```

```
glVertex2f(90.0,-30.0);
glEnd();
/* House side */


}

int clr;
void hwindows()
{
/* Windows */
glColor3f(1.0,1.0,1.0);
glBegin(GL_POLYGON);
glVertex2f(92.0,-43.5);
glVertex2f(92.0,-33.5);
glVertex2f(98.0,-33.5);
glVertex2f(98.0,-43.5);
glEnd();

glBegin(GL_POLYGON);
glVertex2f(112.0,-43.5);
glVertex2f(112.0,-33.5);
glVertex2f(118.0,-33.5);
glVertex2f(118.0,-43.5);
glEnd();

glBegin(GL_POLYGON);
glVertex2f(86.5,-43.5);
glVertex2f(86.5,-33.0);
glVertex2f(88.5,-33.0);
glVertex2f(88.5,-43.5);
glEnd();

/* Windows */
if(clr==1)
{
/* Windows */
glColor3f(0.941, 1.000, 1.000);
glBegin(GL_POLYGON);
glVertex2f(92.0,-43.5);
glVertex2f(92.0,-33.5);
glVertex2f(98.0,-33.5);
glVertex2f(98.0,-43.5);
glEnd();

glBegin(GL_POLYGON);
glVertex2f(112.0,-43.5);
glVertex2f(112.0,-33.5);
glVertex2f(118.0,-33.5);
glVertex2f(118.0,-43.5);
glEnd();

glBegin(GL_POLYGON);
glVertex2f(86.5,-43.5);
glVertex2f(86.5,-33.0);
glVertex2f(88.5,-33.0);
glVertex2f(88.5,-43.5);
glEnd();

/* Windows */
}
if(clr==0)
{
/* Windows */
```

```
glColor3f(0.0,0.0,0.0);
glBegin(GL_POLYGON);
glVertex2f(92.0,-43.5);
glVertex2f(92.0,-33.5);
glVertex2f(98.0,-33.5);
glVertex2f(98.0,-43.5);
glEnd();

glBegin(GL_POLYGON);
glVertex2f(112.0,-43.5);
glVertex2f(112.0,-33.5);
glVertex2f(118.0,-33.5);
glVertex2f(118.0,-43.5);
glEnd();

glBegin(GL_POLYGON);
glVertex2f(86.5,-43.5);
glVertex2f(86.5,-33.0);
glVertex2f(88.5,-33.0);
glVertex2f(88.5,-43.5);
glEnd();
/* Windows */
}
}

void l_loop()
{
/*outline house side*/
glBegin(GL_LINE_LOOP); //outline 1
glLineWidth(0.1);
glColor3f(0.0,0.0,0.0);
glVertex2f(85.0,-30.6);
glVertex2f(90.0,-30.0);
glVertex2f(90.0,-60.0);
glVertex2f(85.0,-60.0);
glEnd();

glBegin(GL_LINE_LOOP); //outline 2
glLineWidth(0.1);
glColor3f(0.0,0.0,0.0);
glVertex2f(85.0,-30.6);
glVertex2f(105.0,-10.0);
glVertex2f(90.0,-30.0);
glEnd();
/*outline house side */

/*outline house triangle top,square base,door*/

glBegin(GL_LINE_LOOP); //outline square base
glLineWidth(0.1);
glColor3f(0.0,0.0,0.0);
glVertex2f(90.0,-60.0);
glVertex2f(90.0,-30.0);
glVertex2f(120.0,-30.0);
glVertex2f(120.0,-60.0);
glEnd();

glBegin(GL_LINE_LOOP); //outline triangle top
glLineWidth(0.1);
glColor3f(0.0,0.0,0.0);
glVertex2f(90.0,-30.0);
glVertex2f(105.0,-10.0);
glVertex2f(120.0,-30.0);
glEnd();
```

```
glBegin(GL_LINE_LOOP); //outline door
glLineWidth(0.1);
glColor3f(0.0,0.0,0.0);
glVertex2f(100.0,-60.0);
glVertex2f(100.0,-45.0);
glVertex2f(110.0,-45.0);
glVertex2f(110.0,-60.0);
glEnd();

/*outline house triangle top,square base,door*/

}

void fence()
{
glBegin(GL_POLYGON);
glColor3f(0.627, 0.322, 0.176);
glVertex2f(122.0,-60.0);
glVertex2f(124.0,-60.0);
glVertex2f(124.0,-45.0);
glVertex2f(122.0,-45.0);
glEnd();

glBegin(GL_POLYGON);
glColor3f(0.627, 0.322, 0.176);
glVertex2f(128.0,-60.0);
glVertex2f(130.0,-60.0);
glVertex2f(130.0,-45.0);
glVertex2f(128.0,-45.0);
glEnd();

glBegin(GL_POLYGON);
glColor3f(0.627, 0.322, 0.176);
glVertex2f(134.0,-60.0);
glVertex2f(136.0,-60.0);
glVertex2f(136.0,-45.0);
glVertex2f(134.0,-45.0);
glEnd();

glBegin(GL_POLYGON);
glColor3f(0.627, 0.322, 0.176);
glVertex2f(138.0,-60.0);
glVertex2f(140.0,-60.0);
glVertex2f(140.0,-45.0);
glVertex2f(138.0,-45.0);
glEnd();

glBegin(GL_POLYGON);
glColor3f(0.627, 0.322, 0.176);
glVertex2f(144.0,-60.0);
glVertex2f(146.0,-60.0);
glVertex2f(146.0,-45.0);
glVertex2f(144.0,-45.0);
glEnd();

glBegin(GL_POLYGON);
glColor3f(0.627, 0.322, 0.176);
glVertex2f(148.0,-60.0);
glVertex2f(150.0,-60.0);
glVertex2f(150.0,-45.0);
glVertex2f(148.0,-45.0);
glEnd();
```

```
glBegin(GL_POLYGON);
glColor3f(0.545, 0.271, 0.075);
glVertex2f(120.0,-52.5);
glVertex2f(120.0,-50.0);
glVertex2f(150.0,-50.0);
glVertex2f(150.0,-52.5);
glEnd();
}

void celestials()
{ glColor3f(0.973, 0.973, 1.000);
glBegin(GL_POINTS);
glPointSize(25.85);
glVertex2f(15.0,80.0);
glVertex2f(-25.0,60.0);
glVertex2f(34.0,96.0);
glVertex2f(-125.0,68.0);
glVertex2f(142.0,82.0);
glVertex2f(-45.0,75.0);
glVertex2f(112.0,64.0);
glVertex2f(-55.0,58.0);
glVertex2f(55.0,140.0);
glVertex2f(-105.0,122.0);
glVertex2f(95.0,135.0);
glVertex2f(-115.0,112.0);
glVertex2f(86.0,105.0);
glVertex2f(-56.0,126.0);
glVertex2f(-39.0,40.0);
glVertex2f(57.0,44.0);
glVertex2f(-105.7,100.0);
glVertex2f(-95.0,30.0);
glVertex2f(-85.0,62.0);
glVertex2f(30.0,62.0);
glVertex2f(44.0,81.5);
glEnd();

/*moon*/
glBegin(GL_POLYGON);
glColor3f(0.973, 0.973, 1.000);
glVertex2f(15.0,150.0);
glVertex2f(20.0,143.5);
glVertex2f(25.0,141.8);
glVertex2f(30.0,143.5);
glVertex2f(35.0,150.0);
glEnd();

}

void aircraft(int x,int y)
{
glBegin(GL_POLYGON); //NOSE
glColor3f(0.863,0.078,0.235);
glVertex2f(-105.0+x,135.0+y);
glVertex2f(-115.0+x,140.0+y);
glVertex2f(-115.0+x,130.0+y);
glEnd();

glBegin(GL_POLYGON); //body
glColor3f( 0.753,0.753,0.753 );
glVertex2f(-115.0+x,140.0+y);
glVertex2f(-136.0+x,142.5+y);
glVertex2f(-145.0+x,132.0+y);
glVertex2f(-140.0+x,127.5+y);
glVertex2f(-115.0+x,130.0+y);
```

```
glEnd();

glBegin(GL_POLYGON); //tail
glColor3f(0.863,0.078,0.235);
glVertex2f(-136.0+x,142.5+y);
glVertex2f(-140.0+x,145.0+y);
glVertex2f(-145.0+x,132.0+y);
glEnd();

glBegin(GL_POLYGON); //exhaust
glColor3f(1.000,0.271,0.000);
glVertex2f(-145.0+x,132.0+y);
glVertex2f(-149.8+x,130.5+y);
glVertex2f(-140.0+x,127.5+y);
glEnd();

glBegin(GL_POLYGON); //cockpit
glColor3f(1.000, 1.000, 0.000);
glVertex2f(-132.0+x,141.5+y);
glVertex2f(-125.5+x,144.0+y);
glVertex2f(-115.0+x,140.0+y);
glEnd();

/* outline aircraft nose and cockpit */

glBegin(GL_LINE_LOOP); //outline nose
glLineWidth(0.1);
glColor3f(0.0,0.0,0.0);
glVertex2f(-105.0+x,135.0+y);
glVertex2f(-115.0+x,140.0+y);
glVertex2f(-115.0+x,130.0+y);
glEnd();

glBegin(GL_LINE_LOOP); //outline cockpit
glLineWidth(0.1);
glColor3f(0.0,0.0,0.0);
glVertex2f(-132.0+x,141.5+y);
glVertex2f(-125.5+x,144.0+y);
glVertex2f(-115.0+x,140.0+y);
glEnd();


/* outline aircraft nose and cockpit */

}

void display1(void)
{
pos+=0.1;
aircraft(pos,y);
glutPostRedisplay();
//glFlush();
//glutSwapBuffers();
}

void aircraftmov1(void)
{
pos+=1.4; //fast-forward
aircraft(pos,y);
glutPostRedisplay();
//glutSwapBuffers();
}

void aircraftmov2(void)
```

```
{
pos-=0.8; //Rewind
aircraft(pos,y);
glutPostRedisplay();
//glutSwapBuffers();
}

void pillar()
{
/* Pillar */
glBegin(GL_POLYGON);
glColor3f(0.439,0.502,0.565);
glVertex2i(0.0,2.0);
glVertex2i(-5.0,-60.0);
glVertex2i(-3.0,-70.0);
glVertex2i(5.0,-70.0);
glVertex2i(8.0,-60.0);
glEnd();
/* Pillar */

/*outline for pillar */
glBegin(GL_LINE_LOOP); //outline p1
glLineWidth(0.1);
glColor3f(0.0,0.0,0.0);
glVertex2i(0.0,2.0);
glVertex2i(-5.0,-60.0);
glVertex2i(-3.0,-70.0);
glVertex2i(5.0,-70.0);
glVertex2i(8.0,-60.0);
glEnd();

glBegin(GL_LINE_LOOP); //outline p2
glLineWidth(0.1);
glColor3f(0.0,0.0,0.0);
glVertex2i(0.0,2.0);
glVertex2i(-5.0,-60.0);
glVertex2i(-3.0,-70.0);
glEnd();

glBegin(GL_LINE_LOOP); //outline p3
glLineWidth(0.1);
glColor3f(0.0,0.0,0.0);
glVertex2i(0.0,2.0);
glVertex2i(5.0,-70.0);
glVertex2i(8.0,-60.0);
glEnd();
}

void chrtr()
{
int i;
glColor3f(0.000, 0.000, 0.502);
glRasterPos2f(-145.0,-135.0);
for(i=0;s[i]!='\0';i++)
glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,s[i]); // char s

glColor3f(0.000, 0.000, 0.502);
glRasterPos2f(115.0,-135.0);
for(i=0;s7[i]!='\0';i++)
glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,s7[i]); // char s7

glColor3f(1.000, 0.843, 0.000);
glRasterPos2f(-149.0,90.0);
for(i=0;s1[i]!='\0';i++)
```

```
glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,s1[i]); // char s1

glColor3f(1.000, 0.843, 0.000);
glRasterPos2f(-120.0,80.0);
for(i=0;s2[i]!='\0';i++)
glutBitmapCharacter(GLUT_BITMAP_9_BY_15,s2[i]); // char s2

glColor3f(0.753, 0.753, 0.753);
glRasterPos2f(-149.0,50.0);
for(i=0;s3[i]!='\0';i++)
glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,s3[i]); // char s3

glColor3f(1.000, 1.000, 0.878);
glRasterPos2f(-120.0,50.0);
for(i=0;s4[i]!='\0';i++)
glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,s4[i]); // char s4

glColor3f(1.000, 1.000, 0.878);
glRasterPos2f(-120.0,35.0);
for(i=0;s5[i]!='\0';i++)
glutBitmapCharacter(GLUT_BITMAP_9_BY_15,s5[i]); // char s5

glColor3f(0.753, 0.753, 0.753);
glRasterPos2f(-149.0,15.0);
for(i=0;s6[i]!='\0';i++)
glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,s6[i]); // char s6

glColor3f(1.000, 1.000, 0.878);
glRasterPos2f(-100.0,15.0);
for(i=0;s8[i]!='\0';i++)
glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,s8[i]); // char s8


}

void display(void)
{

glClear(GL_DEPTH_BUFFER_BIT|GL_COLOR_BUFFER_BIT);
glLoadIdentity();

pillar();
celestials(); // stars & moon clipping
display1(); //aircraft
grass1(); //grass layout
grasselev(); //grassland elevation

drawhouse(); //house function
l_loop(); //outline
hwindows(); // house window func
fence();

chrtr(); // S print

glRotatef(theta[0],-1.0,0.0,0.0); /*rotation of the fan*/
glRotatef(theta[1],0.0,-1.0,0.0); /*rotation of the fan*/
glRotatef(theta[2],0.0,0.0,-1.0); /*rotation of the fan*/

fan();

glutSwapBuffers();
glFlush();
}
```

```
void idle() /*speed at start*/
{
theta[axis]+=2.0;
if(theta[axis]>360.0)
theta[axis]-=360.0;
glutPostRedisplay();
}

void idle1() /*speed at 1st */
{theta[axis]+=4;
if(theta[axis]>360.0)
theta[axis]-=360.0;
glutPostRedisplay();
}

void idle2() /*speed at 2nd */
{theta[axis]+=6;
if(theta[axis]>360.0)
theta[axis]-=360.0;
glutPostRedisplay();
}

void idle3() /*speed at 3rd */
{theta[axis]+=8;
if(theta[axis]>360.0)
theta[axis]-=360.0;
glutPostRedisplay();
}
void idle4() /*speed at 4th */
{theta[axis]+=10;
if(theta[axis]>360.0)
theta[axis]-=360.0;
glutPostRedisplay();
}
void idle5() /*speed at 5th */
{theta[axis]+=12;
if(theta[axis]>360.0)
theta[axis]-=360.0;
glutPostRedisplay();
}

void keys(unsigned char key,int x,int y) /*keyboard function to
implement the fan rotation*/
{
if(key=='s') //start speed
{
glutIdleFunc(idle);
}

if(key=='1') //speed-1
{
glutIdleFunc(idle1);
}

if(key=='2') //speed-2
{
glutIdleFunc(idle2);
}

if(key=='3') //speed-3
{
glutIdleFunc(idle3);
}
```

```
if(key=='4') //speed-4
{
glutIdleFunc(idle4);
}

if(key=='5') //speed-5
{
glutIdleFunc(idle5);
}

if(key=='q') //stop fan
{
glutIdleFunc(NULL);
//exit(0);
}

/* FOR AIRCRAFT */
if(key=='f')
{
aircraftmov1();
}

if(key=='b')
{
aircraftmov2();
}
}

void demo_menu(int id) //menu list demo
{
switch(id)
{
case 3:exit(0);
break;
}
glutPostRedisplay();
}

void color1(int id)
{
switch(id)
{
case 1:global=1;
break;
case 2:global=2;
break;
case 3:global=3;
break;
case 4:global=4;
break;
}
glutPostRedisplay();
}

void lit(int id)
{
switch(id)
{
case 1: clr=1;
break;
case 2: clr=0;
break;
}
}
```

```
void func1(int id)
{
switch(id)
{
case 1:glutIdleFunc(idle);
break;
case 2:glutIdleFunc(NULL);
break;
case 3:glutIdleFunc(idle1);
break;
case 4:glutIdleFunc(idle2);
break;
case 5:glutIdleFunc(idle3);
break;
case 6:glutIdleFunc(idle4);
break;
case 7:glutIdleFunc(idle5);
break;
}
glutPostRedisplay();
}

void tex2(int id)
{
switch(id)
{
case 1:tex1=1;
break;
case 2:tex1=0;
break;
}
glutPostRedisplay();
}

const int TEXDIM = 256;

int main(int argc,char **argv)
{
printf("A Project by Aleen & Arun");
int color,op,text1,wcolor;
GLfloat *tex;

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

glutInitWindowPosition(0,0);
glutInitWindowSize(2400,2600);
glutCreateWindow("Windmill Env");
//glutReshapeFunc(myreshape);
myinit();

color = glutCreateMenu(color1);
glutAddMenuEntry("Default",1);
glutAddMenuEntry("Brown",2);
glutAddMenuEntry("Red",3);
glutAddMenuEntry("Blue",4);


op = glutCreateMenu(func1);
glutAddMenuEntry("Start",1);
glutAddMenuEntry("Stop",2);
glutAddMenuEntry("Speed1",3);
glutAddMenuEntry("Speed2",4);
```

```
glutAddMenuEntry("Speed3",5);
glutAddMenuEntry("Speed4",6);
glutAddMenuEntry("Speed5",7);


text1 = glutCreateMenu(tex2);
glutAddMenuEntry("Enable",1);
glutAddMenuEntry("Disable",2);

wcolor = glutCreateMenu(lit);
glutAddMenuEntry("Switch On",1);
glutAddMenuEntry("Switch Off",2);

glutCreateMenu(demo_menu);
glutAddSubMenu("Fan color",color);
glutAddSubMenu("E/D:Texture",text1);
glutAddSubMenu("Operations",op);
glutAddSubMenu("Window lighting",wcolor);
glutAddMenuEntry("Exit Mainframe",3);
glutAttachMenu(GLUT_RIGHT_BUTTON);

glutDisplayFunc(display);
glutKeyboardFunc(keys);


glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
tex = make_texture(TEXDIM, TEXDIM);
glTexImage2D(GL_TEXTURE_2D, 0, 1, TEXDIM, TEXDIM, 0, GL_RED, GL_FLOAT,
tex);
free(tex);

glutMainLoop();
return 0;
}
```