**Chapter 4**

**IMPLEMENTATION**

**4.1 Source Code**

#include <GL/glut.h>

#include<stdio.h>

#include<conio.h>

#include<process.h>

#define M\_SIDE1 20

#define M\_SIDE2 21

#define M\_BACK 22

#define M\_FRONT 23

#define M\_CUSTOM 24

#define SIZE 500

float x\_angle = 0.0;

float y\_angle = 0.0;

float z\_angle = 0.0;

float camera\_angle=0.0;

float c=1.0;

GLfloat pos[] = { 0.0, 0.0, -10.0, 1.0 };

GLfloat white[] = { 2.5, 2.5, 6.0, 6.0 };

GLfloat red[] = { 0.7, .4, 0.0, 1.0 };

GLfloat deep\_blue[] = { 0.3, 0.3, 0.9, 1.0 };

GLfloat shiny[] = { 50.0 };

GLfloat dull[] = { 0.0 };

GLUquadricObj \*Cylinder;

enum { X, Y, Z } axis = X;

void change\_view (int sel)

{

switch (sel)

{

case M\_CUSTOM:{printf("ENTER VIEWANGLE:");

scanf("%f",&camera\_angle);};

break;

case M\_SIDE1: {camera\_angle=90;}

break;

case M\_SIDE2: camera\_angle=-90;

break;

case M\_BACK: camera\_angle=180;

break;

case M\_FRONT:camera\_angle=0;

default: break;

}

}

void initialize\_menu (void)

{

glutCreateMenu(change\_view );

glutAddMenuEntry("SIDE VIEW 1", 20);

glutAddMenuEntry("SIDE VIEW 2", 21);

glutAddMenuEntry("BACK VIEW", 22);

glutAddMenuEntry("FRONT VIEW", 23);

glutAddMenuEntry("CUSTOM VIEW", 24);

glutAttachMenu(GLUT\_MIDDLE\_BUTTON);

}

void mouse\_button (int button, int state, int x, int y)

{

if (button == GLUT\_LEFT\_BUTTON )

{

axis=Z; c=c+0.4;

printf("WIND SPEED INCREASE\t SPEED=%fKm/Hr\n\n",c\*1.5);

glutPostRedisplay();

}

else if (button == GLUT\_RIGHT\_BUTTON )

{

axis = Z; c=c-0.4;

printf("WIND SPEED DECREASE\t SPEED=%fKm/Hr\n\n",c\*1.5);

glutPostRedisplay();

}

}

void spin(void)

{

switch (axis)

{

case X: x\_angle += 1.0;

break;

case Y: y\_angle += 1.0;

break;

case Z: z\_angle += c ;

break;

default: break;

}

glutPostRedisplay();

}

void display (void)

{

Cylinder = gluNewQuadric();

gluQuadricDrawStyle( Cylinder, GLU\_FILL);

gluQuadricNormals( Cylinder, GLU\_SMOOTH);

gluQuadricOrientation( Cylinder, GLU\_OUTSIDE );

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glEnable(GL\_TEXTURE\_2D);

//Bottom

glTexParameteri(GL\_TEXTURE\_2D,GL\_TEXTURE\_MIN\_FILTER,

GL\_NEAREST);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER,

GL\_NEAREST);

glMaterialfv(GL\_FRONT, GL\_AMBIENT, white);

glBegin(GL\_QUADS);

glNormal3f(0.0, 1.0f, 0.0f);

glTexCoord2f(0.0f, 0.0f);

glVertex3f(-25.0,-25.0,-44);

glTexCoord2f(0.0f, 1.0f);

glVertex3f(-25.0,25.0,-44);

glTexCoord2f(1.0f, 1.0f);

glVertex3f(25.0,25.0,-44);

glTexCoord2f(1.0f, 0.0f);

glVertex3f(25.0,-25.0,-44);

glEnd();

glDisable(GL\_TEXTURE\_2D);

glRotatef(camera\_angle,0.0,1.0,0.0);

gluCylinder(Cylinder,.4,.4,4,16,20);

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_AMBIENT\_AND\_DIFFUSE, red);

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_SPECULAR, red);

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_SHININESS, shiny);

glPushMatrix();

glutSolidTorus (1.4, 1.4, 6, 6);

glutSolidCube(2.5);

glPushMatrix();

glTranslatef(0.0,-2.0,0.0);

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_DIFFUSE, red);

//material property for the base of the windmill

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_SPECULAR, red);

glPushMatrix();

glRotatef(90.0,1.0,0.0,0.0);

glTranslatef(0.0,0.0,-2.0);

gluCylinder(Cylinder,1.0,1.5,27,50,50);

glPopMatrix();

glPopMatrix();

glRotatef(z\_angle, 0.0, 0.0, 1.0);

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_AMBIENT\_AND\_DIFFUSE, red);

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_SPECULAR, white);

glPushMatrix();

glTranslatef(0.0,0.0,1.5);

glutSolidCone(1.5,2.5,50,50);

glPopMatrix();

glPushMatrix(); //blade 1

glTranslatef(0.0,0.0,2.2);

glRotatef(90.0,1.0,0.0,0.0);

glPushMatrix();

glRotatef(120,0.0,1.0,0.0);

glutSolidCone(0.9, 16.0, 15, 15);

glPopMatrix();

glPopMatrix();

glPushMatrix(); //blade 2

glTranslatef(0.0,0.0,2.2);

glRotatef(90.0,1.0,0.0,0.0);

glPushMatrix();

glRotatef(-120,0.0,1.0,0.0);

glutSolidCone(0.9, 16.0, 15, 15);

glPopMatrix();

glPopMatrix();

glPushMatrix(); //blade 3

glTranslatef(0.0,0.0,2.2);

glRotatef(90.0,1.0,0.0,0.0);

glutSolidCone(0.9, 16.0, 15, 15);

glPopMatrix();

glLightfv(GL\_LIGHT1, GL\_POSITION, pos);

glutSwapBuffers();

}

void init (void)

{

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(-25.0, 25.0, -25.0, 25.0, -250.0, 250.0);

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

glEnable(GL\_LIGHT1);

glEnable(GL\_NORMALIZE);

}

void special (int key, int x, int y)

{

switch (key)

{

case GLUT\_KEY\_LEFT: {axis = X; camera\_angle--;

printf("\n\nVIEW ANGLE=%f", camera\_angle);};

glutPostRedisplay();

break;

case GLUT\_KEY\_RIGHT: {axis = Y; camera\_angle++;

printf("\n\nVIEW ANGLE=%f", camera\_angle);};

glutPostRedisplay();

break;

case GLUT\_KEY\_UP: c=c+0.4;

glutPostRedisplay();

break;

case GLUT\_KEY\_END: exit(0);

case GLUT\_KEY\_DOWN:{c=c-0.4;};

glutPostRedisplay();

break;

default: break;

}

}

void reshape (int width, int height)

{

GLfloat w, h;

glViewport(0, 0, width, height);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

if (width > height)

{

w = (25.0 \* width) / height;

h = 25.0;

}

else

{

w = 25.0;

h = (25.0 \* height) / width;

}

glOrtho(-w, w, -h, h, -250.0, 250.0);

glutPostRedisplay();

}

void main (int argc, char \*argv[ ])

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGBA | GLUT\_DOUBLE);

glutInitWindowSize(SIZE, SIZE);

glutInitWindowPosition(100, 50);

glutCreateWindow("SIMULATION OF WINDMILL");

glutIdleFunc(spin);

glutDisplayFunc(display);

glutSpecialFunc(special);

glutMouseFunc(mouse\_button);

initialize\_menu();

glutReshapeFunc(reshape);

init();

glEnable(GL\_DEPTH\_TEST);

glutMainLoop();

}