**5.Develop a program to demonstrate 3D transformation on basic objects**

#include <stdlib.h>

#include <GL/glut.h>

GLfloat vertices[][3] = {{-1.0,-1.0,-1.0},{1.0,-1.0,-1.0},{1.0,1.0,-1.0}, {-1.0,1.0,1.0}, {-1.0,-1.0,1.0} {1.0,-1.0,1.0}, {1.0,1.0,1.0}, {-1.0,1.0,1.0}};

GLfloat normals[][3] = {{-1.0,-1.0,-1.0},{1.0,-1.0,-1.0}, {1.0,1.0,-1.0}, {-1.0,1.0,-1.0},

{-1.0,-1.0,1.0} {1.0,-1.0,1.0}, {1.0,1.0,1.0}, {-1.0,1.0,1.0}};

GLfloat colors[][3] = {{0.0,0.0,0.0},{1.0,0.0,0.0}, {1.0,1.0,0.0}, {0.0,1.0,0.0}, {0.0,0.0,1.0},{1.0,0.0,1.0}, {1.0,1.0,1.0}, {0.0,1.0,1.0}};

void polygon(int a, int b, int c , int d)

{/\* draw a polygon via list of vertices \*/

glBegin(GL\_POLYGON);

glColor3fv(colors[a]);

glNormal3fv(normals[a]);

glVertex3fv(vertices[a]);

glColor3fv(colors[b]);

glNormal3fv(normals[b]);

glVertex3fv(vertices[b]);

glColor3fv(colors[c]);

glNormal3fv(normals[c]);

glVertex3fv(vertices[c]);

glColor3fv(colors[d]);

glNormal3fv(normals[d]);

glVertex3fv(vertices[d]);

glEnd();

}

void display(void)

{/\* display callback, clear frame buffer and z buffer,

rotate cube and draw, swap buffers \*/

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

glLoadIdentity();

glRotatef(theta[0], 1.0, 0.0, 0.0);

glRotatef(theta[1], 0.0, 1.0, 0.0);

glRotatef(theta[2], 0.0, 0.0, 1.0);

colorcube();

glFlush();

glutSwapBuffers();

void spinCube()

{/\* Idle callback, spin cube 2 degrees about selected axis\*/

theta[axis] += 0.1;

if( theta[axis] > 360.0 ) theta[axis] -= 360.0;

/\* display(); \*/

glutPostRedisplay();

}

void mouse(int btn, int state, int x, int y)

{/\* mouse callback, selects an axis about which to rotate \*/

if(btn==GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN) axis = 0; if(btn==GLUT\_MIDDLE\_BUTTON && state == GLUT\_DOWN) axis = 1;

if(btn==GLUT\_RIGHT\_BUTTON && state == GLUT\_DOWN) axis = 2;

}

void myReshape(int w, int h)

{ glViewport(0, 0, w, h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

if (w <= h)

glOrtho(-2.0, 2.0, -2.0 \* (GLfloat) h / (GLfloat) w,

2.0 \* (GLfloat) h / (GLfloat) w, -10.0, 10.0);

else

glOrtho(-2.0 \* (GLfloat) w / (GLfloat) h,

2.0 \* (GLfloat) w / (GLfloat) h, -2.0, 2.0, -10.0, 10.0);

glMatrixMode(GL\_MODELVIEW);

}

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

{

glutInit(&argc, argv);

/\* need both double buffering and z buffer \*/

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH);

glutInitWindowSize(500, 500);

glutCreateWindow("Rotating a Color Cube");

glutReshapeFunc(myReshape);

glutDisplayFunc(display);

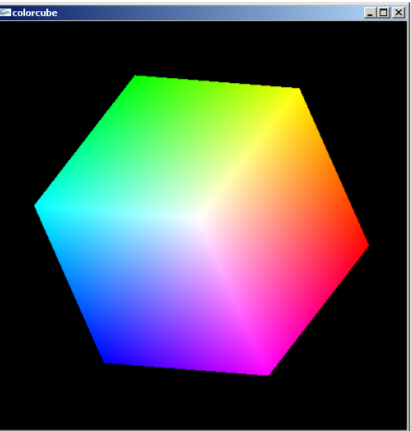
glutIdleFunc(spinCube);

glutMouseFunc(mouse);

glEnable(GL\_DEPTH\_TEST); /\* Enable hidden surface--removal \*/

glutMainLoop();

}



**6. Develop a program to demonstrate Animation effects on simple objects.**

#include <GL/glut.h>

#include <math.h>

#include <stdlib.h>

const double TWO\_PI = 6.2831853;

GLsizei winWidth = 500, winHeight = 500; // Initial display window size.

GLuint regHex; // Define name for display list.

static GLfloat rotTheta = 0.0;

struct scrPt

{

GLint x, y;

};

static void init (void)

{

struct scrPt hexVertex;

GLdouble hexTheta;

GLint k;

glClearColor (1.0, 1.0, 1.0, 0.0);

/\* Set up a display list for a red regular hexagon.

\* Vertices for the hexagon are six equally spaced

\* points around the circumference of a circle.

\*/

regHex = glGenLists (1);

glNewList (regHex, GL\_COMPILE);

glColor3f (1.0, 0.0, 0.0);

glBegin (GL\_POLYGON);

for (k = 0; k < 6; k++) {

hexTheta = TWO\_PI \* k / 6;

hexVertex.x = 150 + 100 \* cos (hexTheta);

hexVertex.y = 150 + 100 \* sin (hexTheta);

glVertex2i (hexVertex.x, hexVertex.y);

}

glEnd ( );

glEndList ( );

}

void displayHex (void)

{

glClear (GL\_COLOR\_BUFFER\_BIT);

glPushMatrix ( );

glRotatef (rotTheta, 0.0, 0.0, 1.0);

glCallList (regHex);

glPopMatrix ( );

glutSwapBuffers ( );

glFlush ( );

}

void rotateHex (void)

{

rotTheta += 3.0;

if (rotTheta > 360.0)

rotTheta -= 360.0;

glutPostRedisplay ( );

}

void winReshapeFcn (GLint newWidth, GLint newHeight)

{

glViewport (0, 0, (GLsizei) newWidth, (GLsizei) newHeight);

glMatrixMode (GL\_PROJECTION);

glLoadIdentity ( );

gluOrtho2D (-320.0, 320.0, -320.0, 320.0);

glMatrixMode (GL\_MODELVIEW);

glLoadIdentity ( );

glClear (GL\_COLOR\_BUFFER\_BIT);

}

void mouseFcn (GLint button, GLint action, GLint x, GLint y)

{

switch (button) {

case GLUT\_MIDDLE\_BUTTON: // Start the rotation.

if (action == GLUT\_DOWN)

glutIdleFunc (rotateHex);

break;

case GLUT\_RIGHT\_BUTTON: // Stop the rotation.

if (action == GLUT\_DOWN)

glutIdleFunc (NULL);

break;

default:

break;

}

}

int main (int argc, char\*\* argv)

{

glutInit (&argc, argv);

glutInitDisplayMode (GLUT\_DOUBLE | GLUT\_RGB);

glutInitWindowPosition (150, 150);

glutInitWindowSize (winWidth, winHeight);

glutCreateWindow ("Animation Example");

init ( );

glutDisplayFunc (displayHex);

glutReshapeFunc (winReshapeFcn);

glutMouseFunc (mouseFcn);

glutMainLoop ( );

return(0);

}

