```
/* Rotating cube with viewer movement.*/
/* Cube definition and display similar to rotating--cube program */

/* We use the Lookat function in the display callback to point
the viewer, whose position can be altered by the x,X,y,Y,z, and Z keys.
The perspective view is set in the reshape callback */

#include <stdio.h>
#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 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}};

static GLfloat theta[] = {0.0,0.0,0.0};
static GLint axis = 2;
static GLdouble viewer[]= {0.0, 0.0, 5.0}; /* initial viewer location */


void polygon(int a, int b, int c , int d)
{
   glBegin(GL_POLYGON);
      glColor3fv(colors[a]);
      glVertex3fv(vertices[a]);
      glColor3fv(colors[b]);
      glVertex3fv(vertices[b]);
      glColor3fv(colors[c]);
      glVertex3fv(vertices[c]);
      glColor3fv(colors[d]);
      glVertex3fv(vertices[d]);
   glEnd();
}

void colorcube()
{
   polygon(0,3,2,1);
   polygon(2,3,7,6);
   polygon(0,4,7,3);
   polygon(1,2,6,5);
   polygon(4,5,6,7);
   polygon(0,1,5,4);
}


void display(void)
{

   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

/* Update viewer position in modelview matrix */

   glLoadIdentity();
   gluLookAt(viewer[0],viewer[1],viewer[2],
             0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

/* rotate cube */

   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();

   glutSwapBuffers();
}


void mouse(int btn, int state, int x, int y)
{
   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;
   theta[axis] += 2.0;
   if( theta[axis] > 360.0 ) theta[axis] -= 360.0;
   display();
}


void keys(unsigned char key, int x, int y)
{

/* Use x, X, y, Y, z, and Z keys to move viewer */

   if(key == 'x') viewer[0]-= 1.0;
   if(key == 'X') viewer[0]+= 1.0;
   if(key == 'y') viewer[1]-= 1.0;
   if(key == 'Y') viewer[1]+= 1.0;
   if(key == 'z') viewer[2]-= 1.0;
   if(key == 'Z') viewer[2]+= 1.0;

   printf("v[x]: %f, v[y]: %f, v[z]: %f.\n",
          viewer[0], viewer[1], viewer[2]);

   display();
}


void myReshape(int w, int h)
{
 glViewport(0, 0, w, h);

/* Use a perspective view */

   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   if(w<=h)
      glFrustum(-2.0, 2.0, -2.0 * (GLfloat) h/ (GLfloat) w,
                2.0* (GLfloat) h / (GLfloat) w, 2.0, 20.0);
   else
      glFrustum(-2.0 * (GLfloat) w / (GLfloat) h,
                2.0 * (GLfloat) w / (GLfloat) h,
```

```
119:                 -2.0, 2.0, 2.0, 20.0);
120:
121: /* Or we can use gluPerspective */
122:
123:   /*gluPerspective(45.0, w/h, 1.0, 10.0); */
124:
125:    glMatrixMode(GL_MODELVIEW);
126: }
127:
128:
129: void main(int argc, char **argv)
130: {
131:    glutInit(&argc, argv);
132:    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
133:    glutInitWindowSize(500, 500);
134:    glutCreateWindow("colorcube");
135:    glutReshapeFunc(myReshape);
136:    glutDisplayFunc(display);
137:    glutMouseFunc(mouse);
138:    glutKeyboardFunc(keys);
139:    glEnable(GL_DEPTH_TEST);
140:    glutMainLoop();
141: }
```