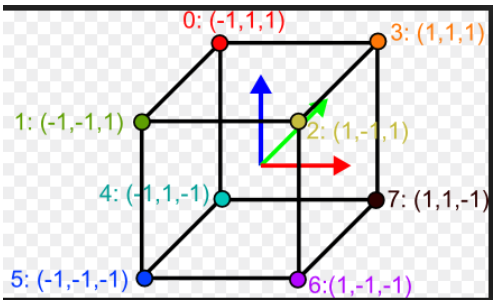
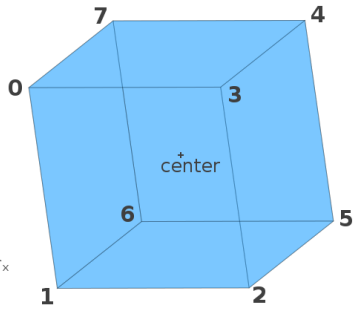
3. Program to draw a color cube and spin it using OpenGL transformation matrices.

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

GLfloat vertices[] = { -1, -1, -1,
                        1, -1, -1,
                        1, 1, -1,
                       -1, 1, -1,
                       -1, -1, 1,
                        1, -1, 1,
                        1, 1, 1,
                       -1, 1, 1
                     };
```




```
GLfloat colors[] =   {  0, 0, 0,   // white color
                        1, 0, 0,   // red color .. so on for eight faces of cube
                        1, 1, 0,
                        0, 1, 0,
                        0, 0, 1,
                        1, 0, 1,
                        1, 1, 1,
                        0, 1, 1
                     };
```



Anti-clockwise -> visible face

Clockwise -> Non visible face

```
GLubyte cubeIndices[] = {0, 3, 2, 1,
                         2, 3, 7, 6,
                         0, 4, 7, 3,
                         1, 2, 6, 5,
                         4, 5, 6, 7,
                         0, 1, 5, 4
                        };
```

glDrawElements() draws a sequence of primitives by hopping around vertex arrays with the associated array indices. It reduces both the number of function calls and the number of vertices to transfer. glDrawElements() requires 4 parameters. The first one is the type of primitive, the second is the number of indices of index array, the third is data type of index array and the last parameter is the address of index array.

```
static GLfloat theta[]= {0, 0, 0};   // initial angles
static GLint axis=2;      // let us assume the right mouse button has been clicked initially

void display(void)
{
   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   glLoadIdentity();

   glRotatef (theta[0], 1, 0, 0);          // first angle rotation via x axis
   glRotatef (theta[1], 0, 1, 0);          // second angle rotation via y axis
   glRotatef (theta[2], 0, 0, 1);          // third angle rotation via z axis

   glDrawElements(GL_QUADS,24,GL_UNSIGNED_BYTE,cubeIndices); // draw the cube
   glutSwapBuffers();                      // show the output
}
```

```
void spinCube()
{
    theta[axis] += 2;              // rotate every 2 degrees

    if (theta[axis] > 360)      // it the rotation angle crosses 360 degrees, make it 0 degree
        theta[axis] -= 360;

    glutPostRedisplay();           // call display again
}

void mouse(int btn, int state, int x, int y)
{
    if (btn==GLUT_LEFT_BUTTON && state==GLUT_DOWN)
        axis=0;                    // x axis rotation

    if (btn==GLUT_MIDDLE_BUTTON && state==GLUT_DOWN)
        axis=1;                    // y axis rotation

    if (btn==GLUT_RIGHT_BUTTON && state==GLUT_DOWN)
        axis=2;                    // z axis rotation
}

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

    if(w<=h)
        glOrtho (-2, 2, -2*(GLfloat)h/(GLfloat)w, 2*(GLfloat)h / (GLfloat)w, -10, 10);
    else
        glOrtho (-2*(GLfloat)w/(GLfloat)h, 2*(GLfloat)w / (GLfloat)h,  -2, 2, -10, 10);

    glMatrixMode(GL_MODELVIEW);
}

int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB|GLUT_DEPTH);
    glutInitWindowSize(500, 500);
    glutCreateWindow("Spin a color cube");

    glutReshapeFunc(myReshape);  // calls myReshape whenever we change the window size

    glutDisplayFunc(display);      // call display function

    glutIdleFunc(spinCube);        // whenever we are idle, calls spinCube function
```

Maintaining the ASPECT RATIO, i.e., whenever we change the window size, our output should remain same, not distorted

```
    glutMouseFunc(mouse);          // calls mouse function whenever we interact with mouse

    glEnable(GL_DEPTH_TEST);   // enables depth – for 3D

    glEnableClientState(GL_COLOR_ARRAY);          // enables colour and vertex properties
    glEnableClientState(GL_VERTEX_ARRAY);

    glVertexPointer(3, GL_FLOAT, 0, vertices); // glVertexPointer(size,type,stride,pointer)
    glColorPointer(3, GL_FLOAT, 0, colors);      // glColorPointer(size,type,stride,pointer)

    glColor3f(1, 1, 1);

    glutMainLoop();
}
```

void glEnableClientState (GLenum cap);

cap

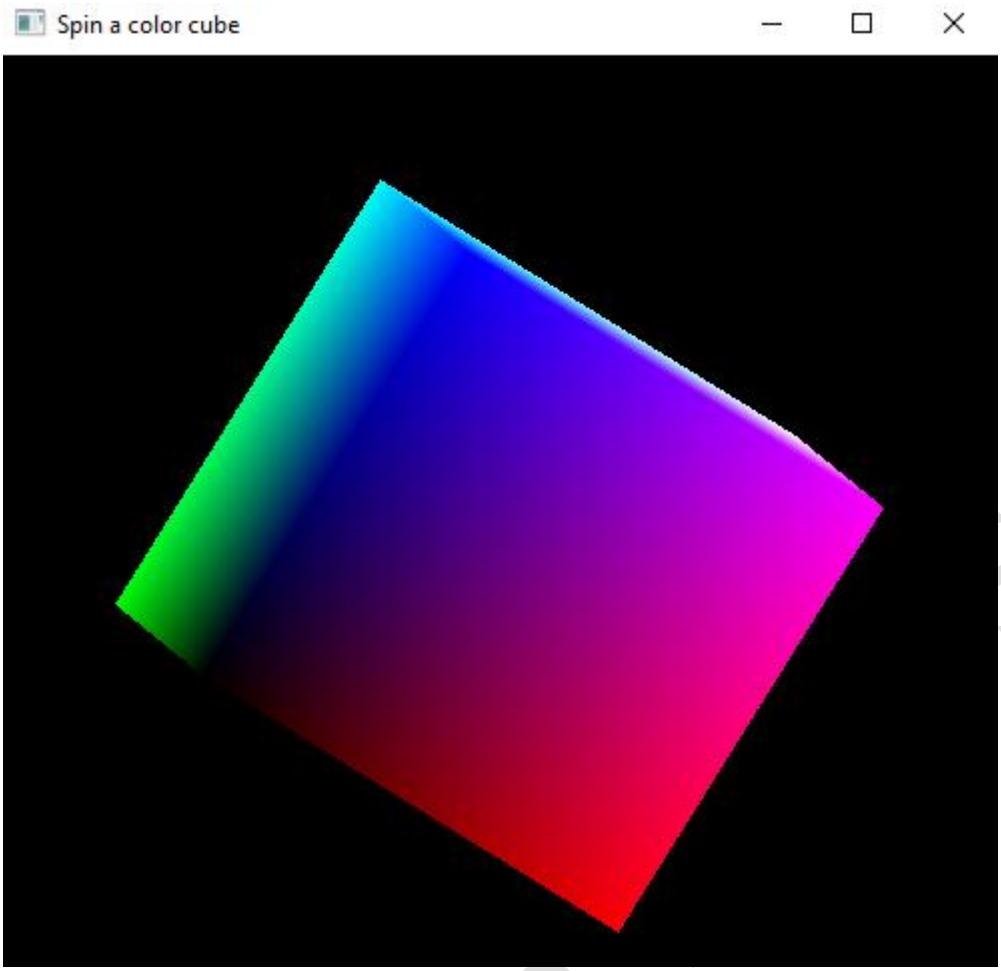
Specifies the capability to enable. Symbolic constants GL_COLOR_ARRAY, GL_EDGE_FLAG_ARRAY, GL_FOG_COORD_ARRAY, GL_INDEX_ARRAY, GL_NORMAL_ARRAY, GL_SECONDARY_COLOR_ARRAY, GL_TEXTURE_COORD_ARRAY, and GL_VERTEX_ARRAY are accepted.

glVertexPointer specifies the location and data format of an array of vertex coordinates to use when rendering. **size** specifies the number of coordinates per vertex, and must be 2, 3, or 4. **type** specifies the data type of each coordinate, and **stride** specifies the byte stride from one vertex to the next, allowing vertices and attributes to be packed into a single array or stored in separate arrays. **pointer** specifies a pointer to the first coordinate of the first vertex in the array. The initial value is 0.

glColorPointer specifies the location and data format of an array of color components to use when rendering. **size** specifies the number of components per color, and must be 3 or 4. **type** specifies the data type of each color component, and **stride** specifies the byte stride from one color to the next, allowing vertices and attributes to be packed into a single array or stored in separate arrays. **pointer** specifies a pointer to the first color of the first vertex in the array. The initial value is 0.

****

OUTPUT



Press left mouse button, middle and right ones and observe the change in the rotation