

**INDIRA COLLEGE OF COMMERCE AND SCIENCE**

**Project Title**

**ROTATING SQUARE AND SPINNIG CUBE**

**T.Y.B.Sc (Computer Science)**

**Year :-2017-2018**

**TEAM MEMBERS :-**

**1.Ganesh Ramesh Gholap**

**2.Aniket Kantilal Gole**

INDIRA COLLEGE OF

COMMERCE AND SCIENCE

TYBSc[Comp. Sci.]

|  |
| --- |
| CERTIFICATE |

**TO WHOMSOEVER IT MAY CONCERN**

This is to certify that the project work entitled . \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ have been successfully completed by :

1. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
2. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

In partial fulfillment in the award of the degree in **TYBSc[Comp.Sci.]** which is laid down by the **SPPU** for the Academic Year **2017-2018**

**Date:**

**Project Guide Principal,**

**Indira College of Commerce and Science**

**Internal Examiner External Examiner**

**ACKNOWLEDGEMENT**

We are very greatful to present the project report on “**Rotating square and spinning cube**”. We would like to express our explicit thank and acknowledgement to our project guides for their kindly and timely guidance.

We are very greatful to all the faculty members from the Computer Science Department especially **Mrs.Kavita Dhakad** for giving us valuable advice, time and suggestions as far as project is concerned.

A special thanks to **Dr. Janardan Pawar (Vice Principal,Computer Science Department)** for this help and providing us IT Services.

**INDEX**

|  |  |  |
| --- | --- | --- |
| **Sr No** | **Title** | **Page no** |
| 1 | Certificate | 2 |
| 2 | Acknowledgement | 3 |
| 3. | Computer Graphics concepts used | 5 |
| 4. | Application code | 7 |
| 5. | Screen Shots | 9 |
| 6. | OpenGL Library and Methods Used | 10 |
| 7. | Bibliography | 14 |

**CONCEPTS USED**

3D Rotation:  
  
In 3D rotation, we have to specify the angle of rotation along with the axis of rotation. We can perform 3D rotation about X, Y, and Z axes.  
Z-Axis Rotation  
  
x' = x\*cos q - y\*sin q  
y' = x\*sin q + y\*cos q   
z' = z  
                | cos q      sin q      0     0|  
Rz (q) =   |-sin     q  cos q      0     0|  
                |0            0        1      0|  
                | 0            0        0      1|  
   
  
X-Axis Rotation  
  
So we do the same replacement in the equations:  
y' = y\*cos q - z\*sin q  
z' = y\*sin q + z\*cos q  
x' = x  
                | 1    0     0     0|  
Rz (q) =   | 0     cos q      sin q     0|  
                | 0            -sin q        cos q      0|  
                | 0            0        0      1|  
   
  
Y-Axis Rotation  
  
So we do the same replacement in equations:  
z' = z\*cos q - x\*sin q  
x' = z\*sin q + x\*cos q  
y' = y  
                | cos q    0     -sin q     0|  
Rz (q) =   | 0     1      0     0|  
                | sin q      0       cos q      0|  
                | 0            0        0      1|

    
Rotation about an Arbitrary Axis  
  
First define the axis of Rotation by 2 points - P1, P2 then do the following:  
1. Translate so that rotation axis passes through origin.  
2. Rotate so that the rotation axis is aligned with one of the principle coordinate axes.  
3. Perform rotation of object about coordinate axis.  
4. Perform inverse rotation of 2.  
5. Perform inverse translation of 1.  
  
[R] =  |n12 + (1 - n12)cosq             n1n2(1 - cosq) + n3sinq        n1n3(1 - cosq) - n2sinq             0|  
          |n1n2(1 - cosq) - n3sinq      n22 + (1 - n22)cosq             n2n3(1 - cosq) + n1sinq            0|  
          |n1n3(1 - cosq) + n2sinq     n2n3(1 - cosq) - n1sinq      n32 + (1 - n32)cosq            0|  
          |0                    0                 0                                 1|

**APPLICATION CODE**

#include<stdlib.h>

#include<GL/glut.h>

GLfloat vertices[]={-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[]=  {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};

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};

static GLfloat theta[]={0.0,0.0,0.0};

static GLint axis=2;

void display(void)

{

  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);

  glDrawElements(GL\_QUADS,24,GL\_UNSIGNED\_BYTE, cubeIndices);

  glFlush();

  glutSwapBuffers();

}

void spinCube()

{

    theta[axis]+=2.0;

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

    glutPostRedisplay();

}

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;

}

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);

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

    glutInitWindowSize(500,500);

    glutCreateWindow("Spin a color cube");

    glutReshapeFunc(myReshape);

    glutDisplayFunc(display);

    glutIdleFunc(spinCube);

    glutMouseFunc(mouse);

    glEnable(GL\_DEPTH\_TEST);

    glEnableClientState(GL\_COLOR\_ARRAY);

    glEnableClientState(GL\_VERTEX\_ARRAY);

    glVertexPointer(3,GL\_FLOAT, 0, vertices);

    glColorPointer(3, GL\_FLOAT, 0, colors);

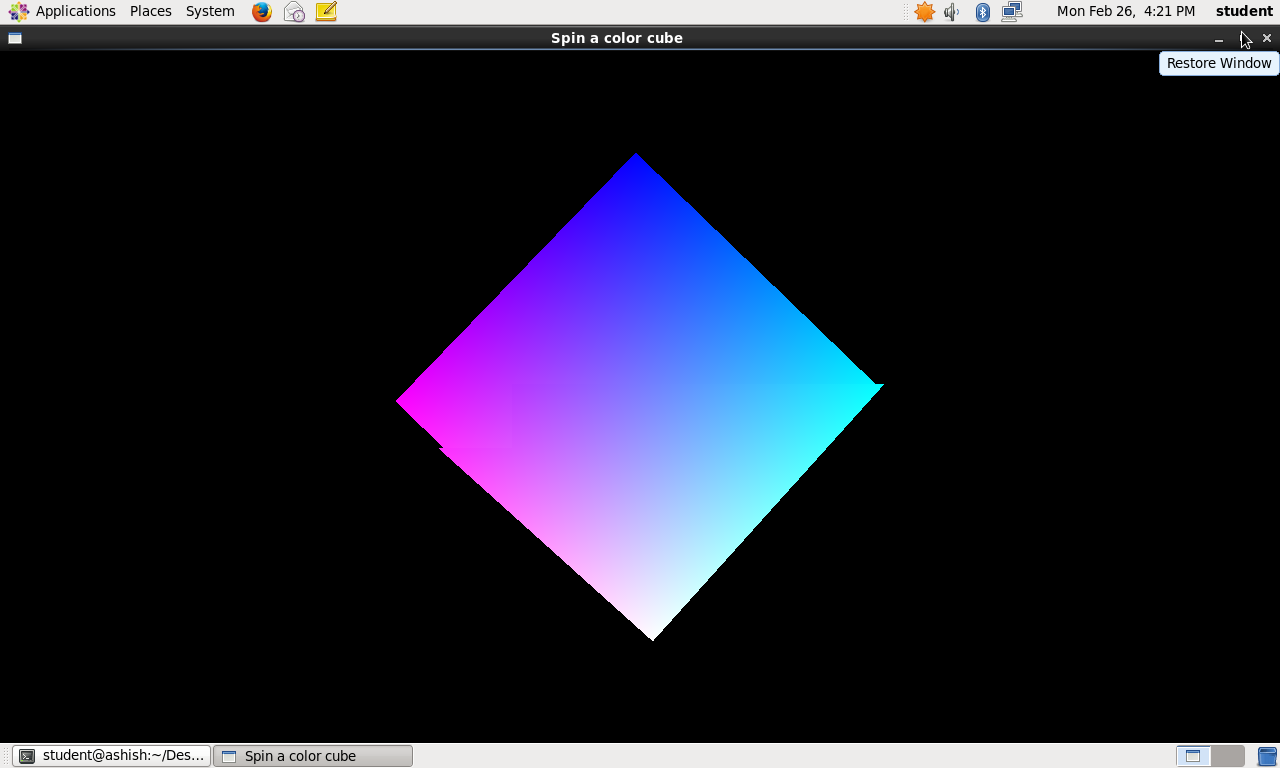
  glColor3f(1.0,1.0,1.0);

    glutMainLoop();

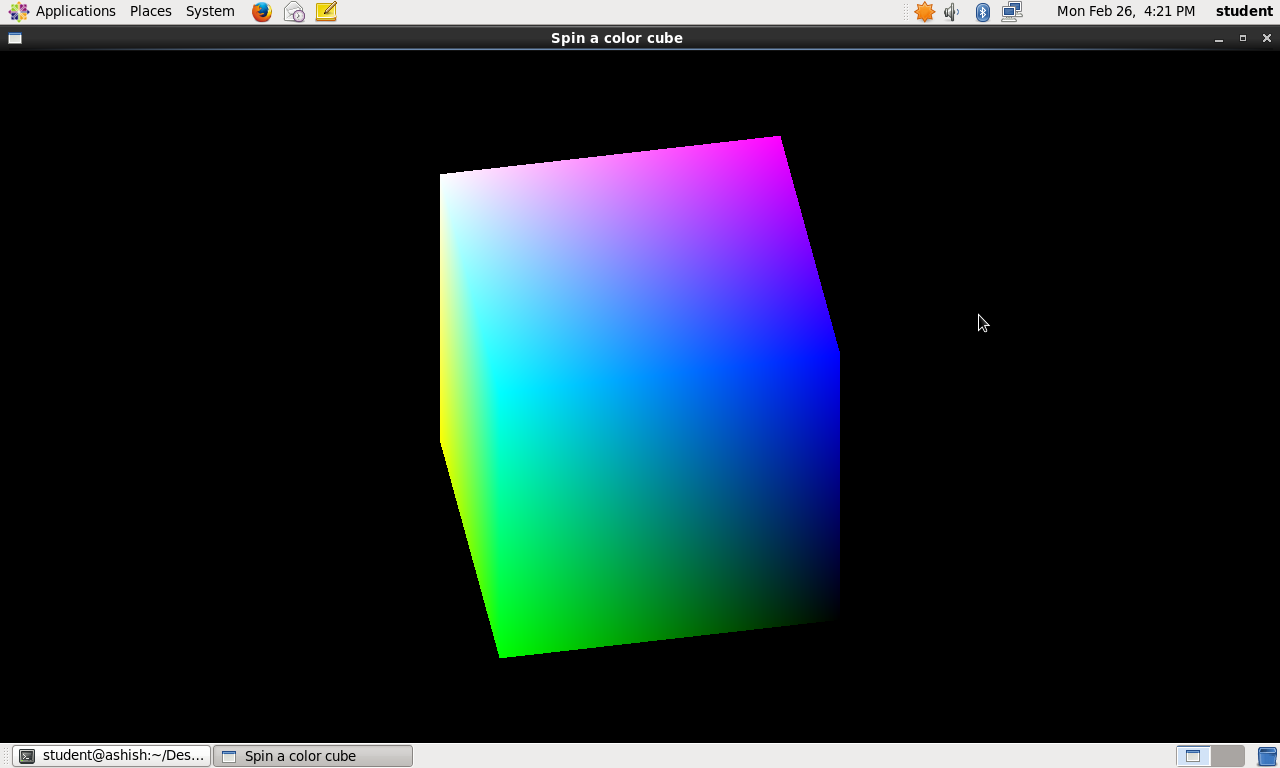
}

**SCREEN SHOTS**

**1)Rotating Square**



**2)Spinning Cube**



**OPENGL LIBRARY AND API USED**

1. **Library:**

## [OpenGL](http://en.wikipedia.org/wiki/OpenGL)

OpenGL (Open Graphics Library) is cross-platform API for rendering 2D and 3D graphics. The API mainly tries to focus on using the GPU to achieve [hardware-accelerated](http://en.wikipedia.org/wiki/Hardware_acceleration) rendering.

## [GLU](http://en.wikipedia.org/wiki/OpenGL_Utility_Library)

GLU (OpenGL Utility Library) is a graphics library for OpenGL, consisting of utility functions which can be used with OpenGL. The functions mainly focus on primitive rendering and mapping between screen- and world-coordinates, etc.

## [GLUT](http://en.wikipedia.org/wiki/OpenGL_Utility_Toolkit)

GLUT (OpenGL Utility Toolkit) is a library of utilities for OpenGL, which primarily focuses on window definition, window control and monitoring of keyboard and mouse input.

**B API:**

* **void glutMainLoop(void);**

glutMainLoop enters the GLUT event processing loop. This routine should be called at most once in a GLUT program. Once called, this routine will never return. It will call as necessary any callbacks that have been registered.

* **void glutDisplayFunc(void (\*func)(void));**

glutDisplayFunc sets the display callback for the current window. When GLUT determines that the normal plane for the window needs to be redisplayed, the display callback for the window is called. Before the callback, the current window is set to the window needing to be redisplayed and the layer in use is set to the normal plane.

* **void glutKeyboardFunc(void (\*func)(unsigned char key, int x, int y));**

glutKeyboardFunc sets the keyboard callback for the current window. When a user types into the window, each key press generating an ASCII character will generate a keyboard callback. The key callback parameter is the generated ASCII character.

* **void glutReshapeFunc(void (\*func)(int width, int height));**

glutReshapeFunc sets the reshape callback for the current window. The reshape callback is triggered when a window is reshaped. A reshape callback is also triggered immediately before a window's first display callback after a window is created or whenever an overlay for the window is established.

* **int glutCreateWindow(char \*name);**

glutCreateWindow creates a top-level window. The name will be provided to the window system as the window's name. The intent is that the window system will label the window with the name.

* **void glutInit(int \*argcp, char \*\*argv);**

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.

* **void glutInitDisplayMode(unsigned int mode);**

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.

* **void glutInitWindowSize(int width, int height);**

Windows created by glutCreateWindow will be requested to be created with the current initial window position and size. The initial value of the initial window size GLUT state is 300 by 300. The initial window size components must be greater than zero. The intent of the initial window position and size values is to provide a suggestion to the window system for a window's initial size and position.

* **void glutPostRedisplay(void);**

Mark the normal plane of current window as needing to be redisplayed. The next iteration through glutMainLoop, the window's display callback will be called to redisplay the window's normal plane. Multiple calls to glutPostRedisplay before the next display callback opportunity generates only a single redisplay callback. glutPostRedisplay may be called within a window's display or overlay display callback to re-mark that window for redisplay.

* **void glColor3f(GLfloat pink, GLfloat yellow, GLfloat blue);**

glColor has two major variants: glColor3f and glColor4f. glColor3f variants specify new pink, yellow, and blue values explicitly and set the current alpha value to 1.0 (full intensity) implicitly.

* **void glVertex3f(GLfloat x, GLfloat y, GLfloat z);**

This is the default trigonometry system coordinates. The default view will make Z axis points to you, X axis at right and Y at top.

**Bibliography**

1. **Computer Graphics course book (Tech-Max) publication**
2. **Online resources**
3. **Website Referred**

[**www.opengl.org**](http://www.opengl.org/)

[**www.stackoverflow.com**](http://www.stackoverflow.com/)

[**www.codeproject.com**](http://www.codeproject.com/)