

**LAB MANUAL**  
**for**  
**Computer Graphics ( PCS 702)**  
**B.Tech CSE**



**GRAPHIC ERA HILL UNIVERSITY**  
**(Society Area, Clement Town, Dehradun)**  
**[www.gehu.ac.in](http://www.gehu.ac.in)**  
**Department of Computer Science & Engineering**

## OBJECTIVES OF COMPUTER GRAPHICS LAB

The main objective of this lab is to introduce to the students, the notions and aspects of developing graphics for computers using both primitive and derived. A student will be acquainted with the basic algorithms and their implementations in a programming language to create basic primitives like line triangles, line segments, polygons etc. They will be exposed to different kinds of transformations in graphics. This lab course offers a comprehensive coverage of Computer Graphics which is in sync with the industry practices and consumer demand regarding graphical software products.

In this lab students are going to learn about

- Creating various **computer graphics primitives** like lines, polygons, etc.
- Applying various **transformations** in computer graphics like translation, rotation, etc.
- Creating various **2d and 3d graphics** in opengl library.
- Creating **bezier curves** of various degrees with animating line segment.
- Creating and using various **opengl functions** to rotate 3d models.
- Creating **color buffer** and enabling **Z depth test** in opengl.
- Creating models in Computer graphics which can be transformed using **keys** and mouse

### PREREQUISITES FOR THIS LAB:

Students must have a good knowledge about JavaScript and C++ programming languages and how to use a build system, IDE and UNIX operating system. She must be familiar with the concepts of application development and should be able to write code for simple logic based problems and handle operational aspects of the underlying operating system .

Tools Required :

1. Unix Os
2. Opengl library 3.3+
3. GCC compiler tools for c++
4. IDE for c++.

## List of Programs

| S.No. | Programs Name  |
|-------|--|
| 1.    | Write a Code in opengl to draw a triangle.   |
| 2.    | Write a Code in opengl to create a polygon .   |
| 3.    | Write a Code in opengl to create a line loop .   |
| 4.    | Write a Code in opengl to create a line strip .  |
| 5.    | Write a Code in opengl to create a triangle fan .  |
| 6.    | Write a Code in opengl to create a triangle strip .  |
| 7.    | Write a Code in opengl to create a self rotating triangle .                                      |
| 8.    | Write a Code in opengl to create a triangle with a given color and rotate it with keys.          |
| 9.    | Write a Code in opengl to create a self rotating polygons .                                      |
| 10.   | Write a Code in opengl to create a self rotating cube with its each face having different color. |
| 11.   | Write a code in javascript to create a bezier curve which runs in any browser.                   |

**Program 1:** Write a Code in opengl to draw a triangle.

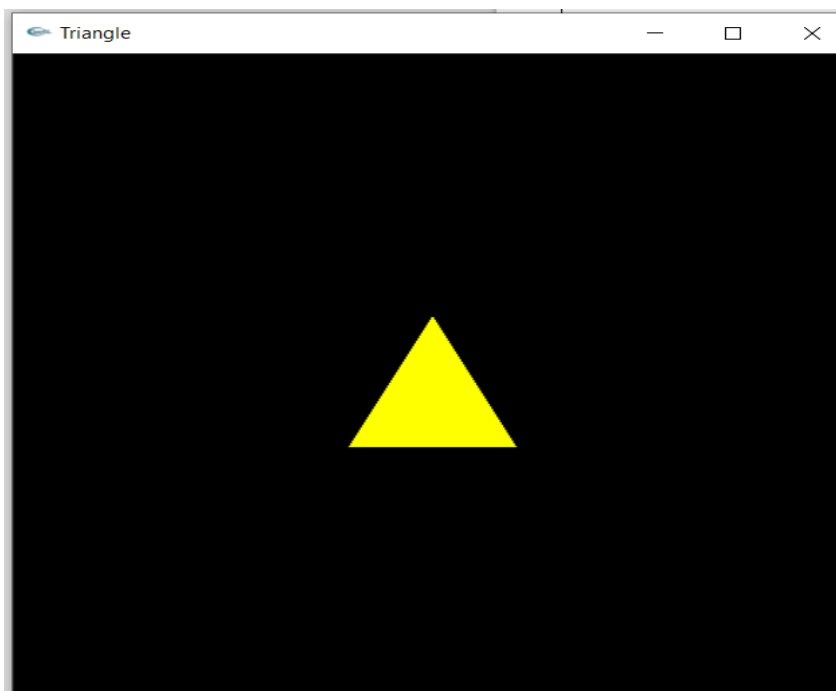
**Code:-**

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

void display()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glClearColor(1.0, 1.0, 1.0, 1.0);
    glColor3f(1.0, 1.0, 0.0);
    glBegin(GL_TRIANGLES);
    glVertex3f(-0.2, -0.2, 0.0);
    glVertex3f(0.2, -0.2, 0.0);
    glVertex3f(0.0, 0.2, 0.0);
    glEnd();
    glFlush();
}

int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(50, 50);
    glutCreateWindow("Triangle");
    glutDisplayFunc(display);
    glutMainLoop();
}
```

**OUTPUT-**



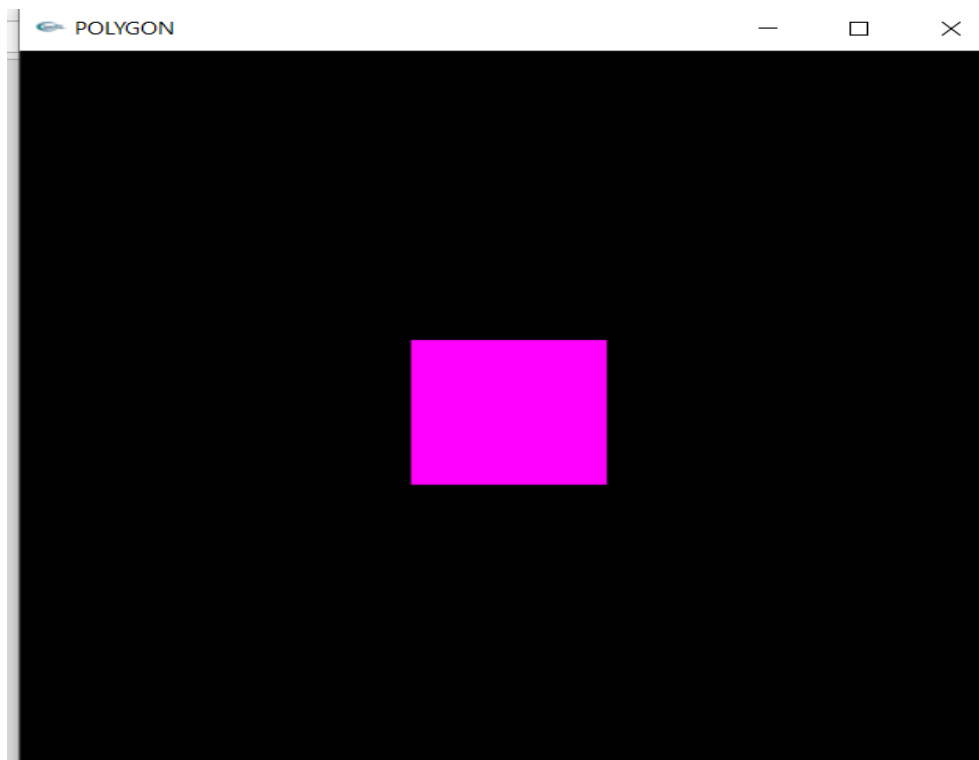
**Program 2:** Write a Code in opengl to create a polygon.  
**Code:-**

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

void display()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glClearColor(1.0, 1.0, 1.0, 1.0);
    glColor3f(1.0, 0.0, 1.0);
    glBegin(GL_POLYGON);
    glVertex3f(-0.2, 0.2, 0.0);
    glVertex3f(0.2, 0.2, 0.0);
    glVertex3f(0.2, -0.2, 0.0);
    glVertex3f(-0.2, -0.2, 0.0);
    glEnd();
    glFlush();
}

int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(100, 100);
    glutCreateWindow("POLYGON");
    glutDisplayFunc(display);
    glutMainLoop();
}
```

**OUTPUT-**



**Program 3:** Write a Code in opengl to create a line loop.

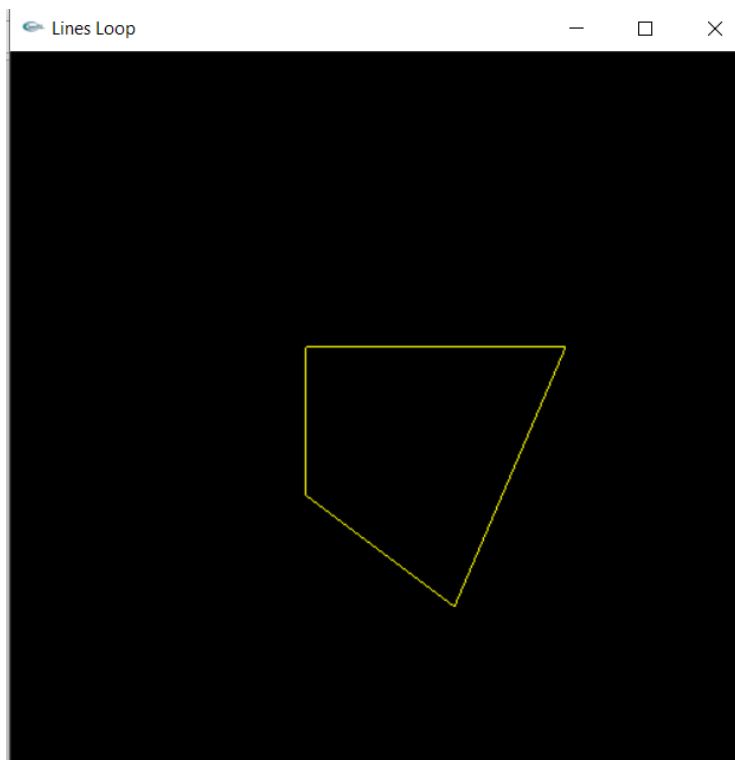
**Code:-**

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

void display()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glClearColor(1.0, 1.0, 1.0, 1.0);
    glColor3f(1.0, 1.0, 0.0);
    glBegin(GL_LINE_LOOP);
    glVertex3f(-0.2, 0.2, 0.0);
    glVertex3f(0.5, 0.2, 0.0);
    glVertex3f(0.2, -0.5, 0.0);
    glVertex3f(-0.2, -0.2, 0.0);
    glVertex3f(-0.2, 0.2, 0.0);
    glEnd();
    glFlush();
}

int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(100, 100);
    glutCreateWindow("Lines Loop");
    glutDisplayFunc(display);
    glutMainLoop();
}
```

**OUTPUT-**



**Program 4:** Write a Code in opengl to create a line strip .

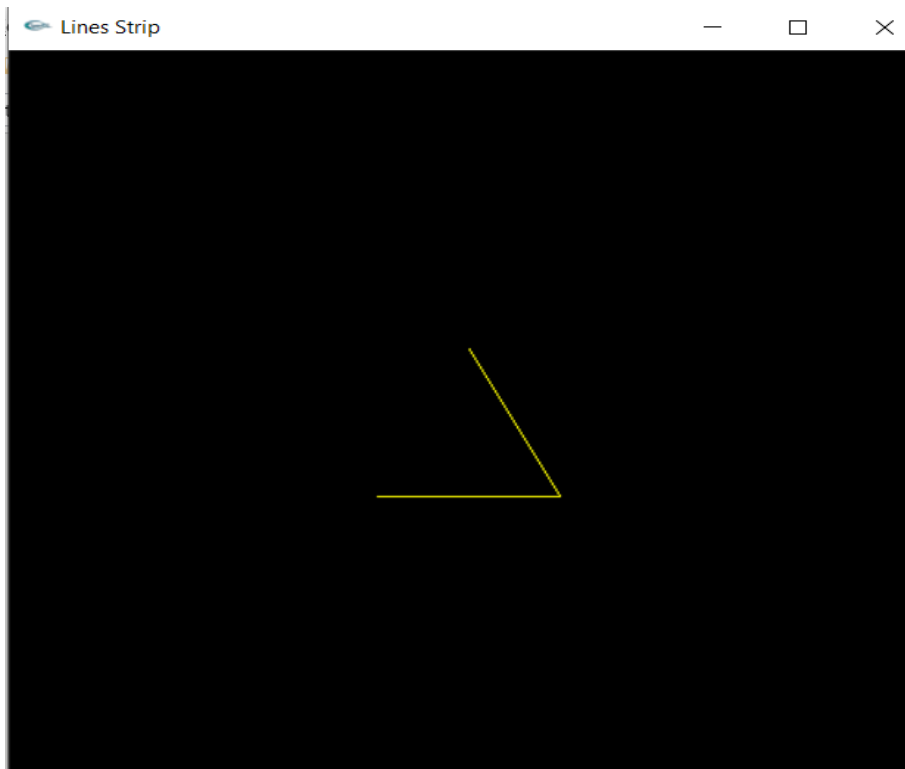
**Code:-**

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

void display()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glClearColor(1.0, 1.0, 1.0, 1.0);
    glColor3f(1.0, 1.0, 0.0);
    glBegin(GL_LINE_STRIP);
    glVertex3f(-0.2, -0.2, 0.0);
    glVertex3f(0.2, -0.2, 0.0);
    glVertex3f(0.0, 0.2, 0.0);
    glEnd();
    glFlush();
}

int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(50, 50);
    glutCreateWindow("Lines Strip");
    glutDisplayFunc(display);
    glutMainLoop();
}
```

**OUTPUT-**



**Program 5:** Write a Code in opengl to create a triangle fan.

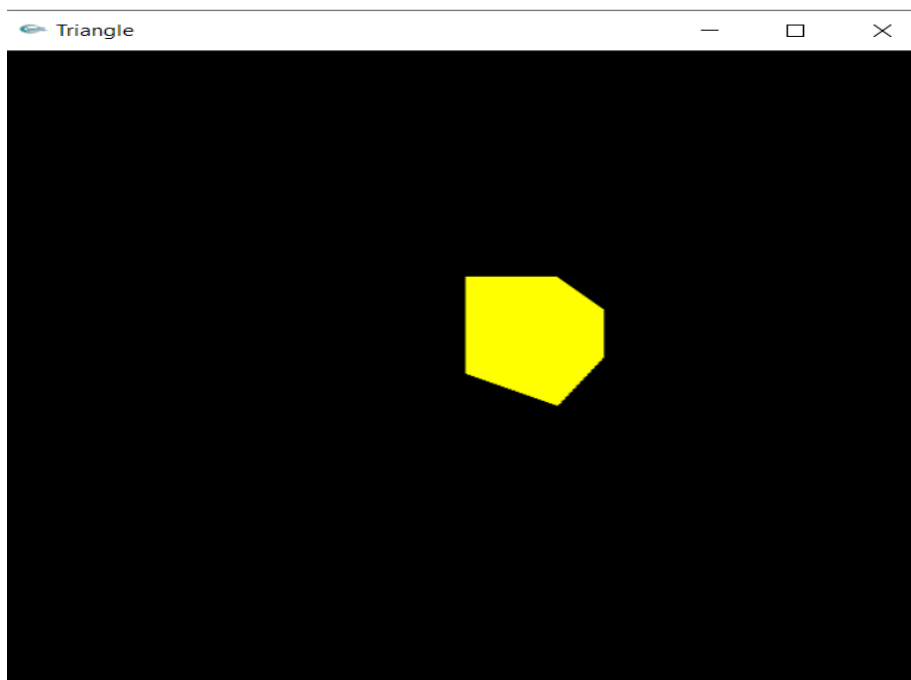
**Code:-**

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

void display()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glClearColor(1.0, 1.0, 1.0, 1.0);
    glColor3f(1.0, 1.0, 0.0);
    glBegin(GL_TRIANGLE_FAN);
    glVertex2f(0.0, 0.0);
    glVertex2f(0.0, 0.3);
    glVertex2f(0.2, 0.3);
    glVertex2f(0.3, 0.2);
    glVertex2f(0.3, 0.05);
    glVertex2f(0.2, -0.1);
    glEnd();
    glFlush();
}

int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(50, 50);
    glutCreateWindow("Triangle");
    glutDisplayFunc(display);
    glutMainLoop();
}
```

**OUTPUT-**





**Program 6:** Write a Code in opengl to create a triangle strip.

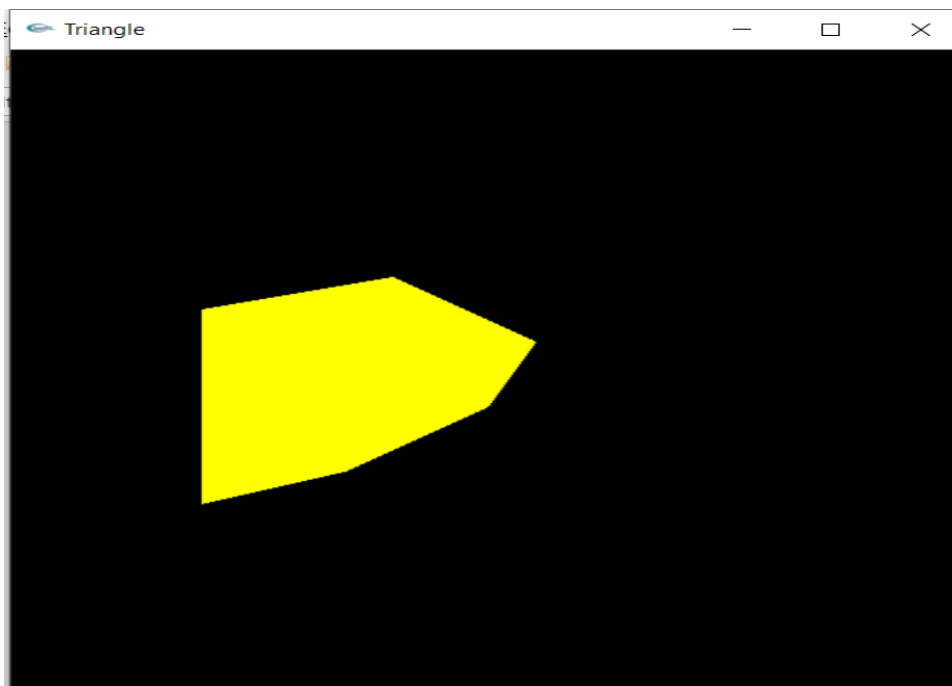
**Code:-**

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

void display()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glClearColor(1.0, 1.0, 1.0, 1.0);
    glColor3f(1.0, 1.0, 0.0);
    glBegin(GL_TRIANGLE_STRIP);
    glVertex2f(-0.6,-0.4);
    glVertex2f(-0.6,0.2);
    glVertex2f(-0.3,-0.3);
    glVertex2f(-0.2,0.3);
    glVertex2f(0.0,-0.1);
    glVertex2f(0.1,0.1);
    glEnd();
    glFlush();
}

int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(50, 50);
    glutCreateWindow("Triangle");
    glutDisplayFunc(display);
    glutMainLoop();
}
```

**OUTPUT-**



**Program 7:** Write a Code in opengl to create a rotating triangle.

**Code:-**

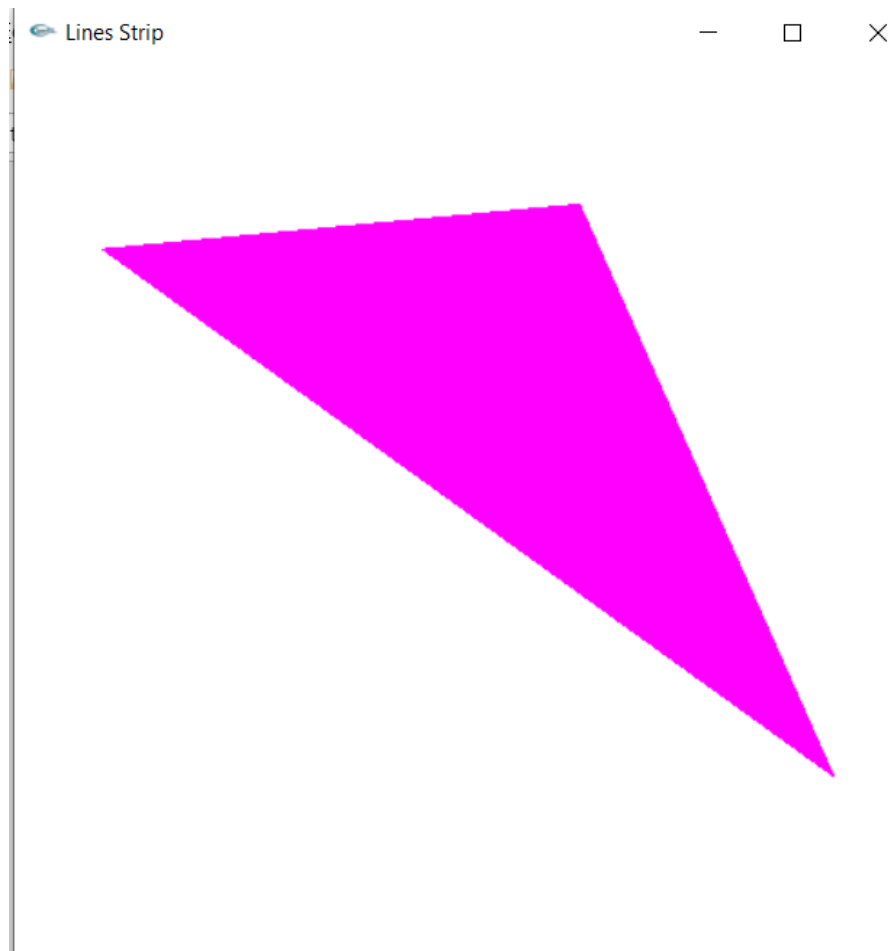
```
#include<GL/glut.h>
#include<GL/gl.h>
float angle = 0.0f;
void display()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glClearColor(1.0, 1.0, 1.0, 1.0);
    glRotatef(angle,0.0,0.0,1.0);
    glColor3f(1.0, 0.0, 1.0);
    glBegin(GL_TRIANGLES);
    glVertex3f(0,1.0,0);
    glVertex3f(0,-1,0);
    glVertex3f(0.7,0.2,0);
    glEnd();
    glFlush();
}

void update(int x)
{
    angle+=0.2f;
    if(angle >= 360)
        angle-= 360;

    glutPostRedisplay();
    glutTimerFunc(50, update, 0);
}

int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(50, 50);
    glutCreateWindow("Lines Strip");
    glutDisplayFunc(display);
    glutTimerFunc(500, update, 0);
    glutMainLoop();
}
```

## OUTPUT-



**Program 8:** Write a Code in opengl to create a triangle with a given color and rotate it with keys.

**Code:-**

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

float angle = 0.0f;
void display()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glClearColor(1.0, 1.0, 1.0, 1.0);
    glRotatef(angle, 0.0,0.0, 1.0);
    glColor3f(1.0, 1.0, 0.0);
    glBegin(GL_TRIANGLES);
    glVertex3f(0.0, 0.0, 0.0);
    glVertex3f(-0.5, -0.5, 0.0);
    glVertex3f(0.5, -0.5, 0.0);
    glEnd();
    glFlush();
}

void keyPress(int key, int x, int y)
{
    if(key == 27)
        exit(0);
    if(key == GLUT_KEY_RIGHT)
        angle+=5;
    if(key == GLUT_KEY_LEFT)
        angle-=5;
    glutPostRedisplay();
}

int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(50, 50);
    glutCreateWindow("Triangle");
    glutDisplayFunc(display);
    glutSpecialFunc(keyPress);
    glutMainLoop();
}
```

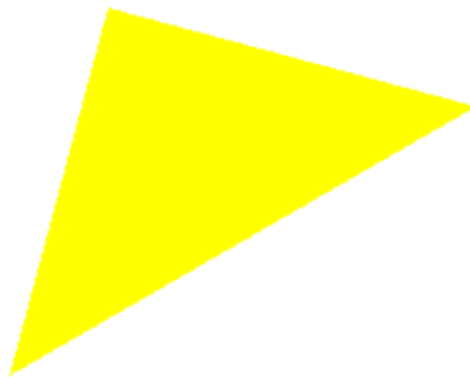
## OUTPUT-

 Triangle

—

□

×



**Program 9:** Write a Code in opengl to create create a self rotating polygons.  
**Code:-**

```
#include<GL/glut.h>
#include<GL/gl.h>
float angle = 0.0f;
void display()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glClearColor(1.0, 1.0, 1.0, 1.0);
    glColor3f(1.0, 0.0, 1.0);
    glRotatef(angle, 0.0, 0.0, 1.0);
    glBegin(GL_POLYGON);
    glVertex3f(-0.2, 0.2, 0.0);
    glVertex3f(0.2, 0.2, 0.0);
    glVertex3f(0.2, -0.2, 0.0);
    glVertex3f(-0.2, -0.2, 0.0);
    glEnd();
    glFlush();
}


void update(int x)
{
    angle+=2.0f;

    if(angle >= 360)
        angle-=360;

    glutPostRedisplay();
    glutTimerFunc(150, update, 0);
}

int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(100, 100);
    glutCreateWindow("POLYGON");
    glutDisplayFunc(display);
    glutTimerFunc(150, update, 0);
    glutMainLoop();
}
```

## OUTPUT-

 POLYGON

—

□

×



**Program 10:** Write a Code in opengl to create a self rotating cube with its each face having different color.

**Code:-**

```
#include "GL/freeglut.h"
#include "GL/gl.h"
float _angle=0.0f;
void drawTriangle()
{
glClearColor(0.3,0.4,0.0,0.0);
glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
glRotatef(15,0.1,0.0,1.0);
//glOrtho(-1.0,1.0,-1.0,1.0,-1.0,1.0);
glBegin(GL_POLYGON);

glColor3f( 1.0, 0.0, 0.0 );    glVertex3f( 0.5, -0.5, -0.5 );    // P1 is red
glColor3f( 0.0, 1.0, 0.0 );    glVertex3f( 0.5, 0.5, -0.5 );    // P2 is
green
glColor3f( 0.0, 0.0, 1.0 );    glVertex3f( -0.5, 0.5, -0.5 );    // P3 is blue
glColor3f( 1.0, 0.0, 1.0 );    glVertex3f( -0.5, -0.5, -0.5 );    // P4 is
purple glEnd();

// White side - BACK glBegin(GL_POLYGON); glColor3f( 1.0, 1.0, 1.0 );
glVertex3f( 0.5, -0.5, 0.5 );
glVertex3f( 0.5, 0.5, 0.5 );
glVertex3f( -0.5, 0.5, 0.5 );
glVertex3f( -0.5, -0.5, 0.5 ); glEnd();

// Purple side - RIGHT glBegin(GL_POLYGON);
glColor3f( 1.0, 0.0, 1.0 );
glVertex3f( 0.5, -0.5, -0.5 );
glVertex3f( 0.5, 0.5, -0.5 );
glVertex3f( 0.5, 0.5, 0.5 );
glVertex3f( 0.5, -0.5, 0.5 ); glEnd();

// Green side - LEFT glBegin(GL_POLYGON); glColor3f( 0.0, 1.0, 0.0 );
glVertex3f( -0.5, -0.5, 0.5 );
glVertex3f( -0.5, 0.5, 0.5 );
glVertex3f( -0.5, 0.5, -0.5 );
glVertex3f( -0.5, -0.5, -0.5 );
glEnd();

// Blue side - TOP glBegin(GL_POLYGON); glColor3f( 0.0, 0.0, 1.0 );
glVertex3f( 0.5, 0.5, 0.5 );
glVertex3f( 0.5, 0.5, -0.5 );
glVertex3f( -0.5, 0.5, -0.5 );
glVertex3f( -0.5, 0.5, 0.5 ); glEnd();

// Red side - BOTTOM glBegin(GL_POLYGON); glColor3f( 1.0, 0.0, 0.0 );
glVertex3f( 0.5, -0.5, -0.5 );glVertex3f( 0.5, -0.5, 0.5 );
glVertex3f( -0.5, -0.5, 0.5 );
glVertex3f( -0.5, -0.5, -0.5 );
glEnd(); glFlush(); glutSwapBuffers();
}
```



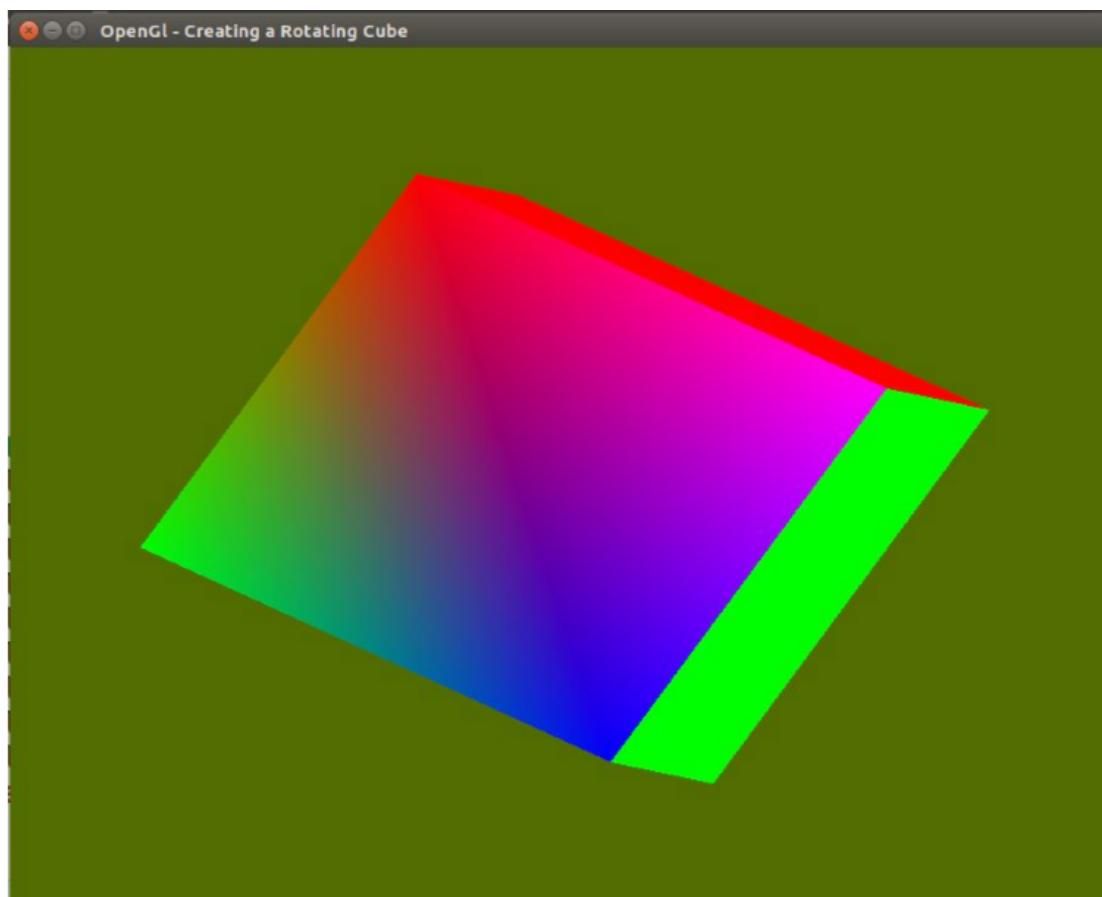
```

void update(int value) {
    _angle += 2.0f;
    /* if (_angle > 360) {
        _angle -= 360;
    }*/
    glutPostRedisplay(); //Tell GLUT that the display has changed
    //Tell GLUT to call update again in 150 milliseconds
    glutTimerFunc(300, update, 0);
}

int main(int argc, char **argv)
{
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB|GLUT_DEPTH);
    glutInitWindowSize(900,700);
    glutInitWindowPosition(100,100);
    glutCreateWindow("OpenGL - Creating a Triangle");
    glEnable(GL_DEPTH_TEST);
    glutDisplayFunc(drawTriangle);
    glutTimerFunc(50, update, 0);
    glutMainLoop();
    return 0;
}

```

## OUTPUT-



**Program 11:** Write a Code in javascript to create a bezier curve which runs in any browser

**Code:-**

```
<html>
<body>
<canvas id="canvas" width="450" height="300" style="border:1px solid #a3a3a3"></
canvas>
<script>
var canvas=document.getElementById("canvas") var c=canvas.getContext("2d");
var tt=0;
var xc=260,yc=90;
function draw()
{
c.clearRect(0,0,450,300);
c.fillStyle="#d9d9d9";
c.fillRect(0,0,450,300);
//drawing a beizer curve of degree two
var x0=120,y0=200,x1=xc,y1=yc,x2=360,y2=200;
var x=x0,y=y0;
//function to draw animated line function drawAnimatedLine()
{ c.strokeStyle="yellow";
c.beginPath();
x01=(x1-x0)*tt+x0; x11=(x2-x1)*tt+x1;
y01=(y1-y0)*tt+y0;
y11=(y2-y1)*tt+y1;
tt=tt+0.005;
if(tt>=1)
tt=0; c.moveTo(x01,y01);
c.lineTo(x11,y11);
c.stroke(); c.closePath();
c.fillStyle="red";
}
function dragPoints(ev)
{ xc=ev.clientX;
yc=ev.clientY;
}
canvas.onmousemove=(evt)=>{dragPoints(evt)};
//drawing convex hull for curve c.strokeStyle="green";
c.beginPath();
c.moveTo(x0,y0);
c.lineTo(x1,y1);
c.lineTo(x2,y2);
c.stroke();

//convex hull ends here c.fillStyle="red";
c.beginPath();

for(var t=0.0;t<=1.0;t=t+0.005
{
c.moveTo(x,y);
c.arc(x-1,y,1,0,2*Math.PI);
x= (t*t)*x0 + 2*(t*(1-t)) *x1 +(1-t)*(1-t)*x2;
y= (t*t)*y0 + 2*(t*(1-t)) *y1 +(1-t)*(1-t)*y2;
}
```

```
c.moveTo(x,y);  
c.arc(x-1,y,1,0,2*Math.PI);  
c.fill();  
drawAnimatedLine();  
requestAnimationFrame(draw);  
}
```

```
requestAnimationFrame(draw);  
</script>  
</body>  
</html>
```

## OUTPUT-

