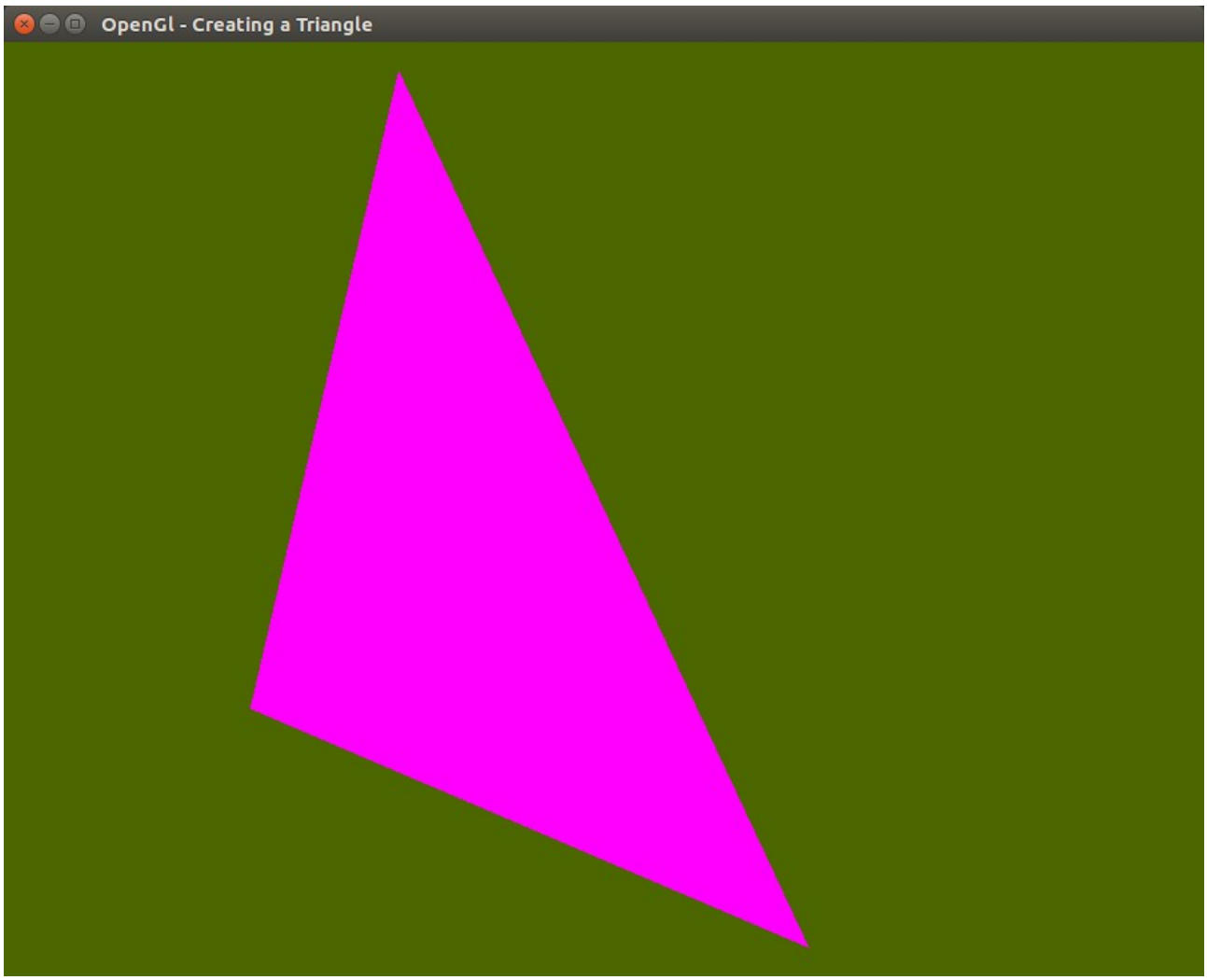
CG PRACTICALS
1. code to draw a triangle

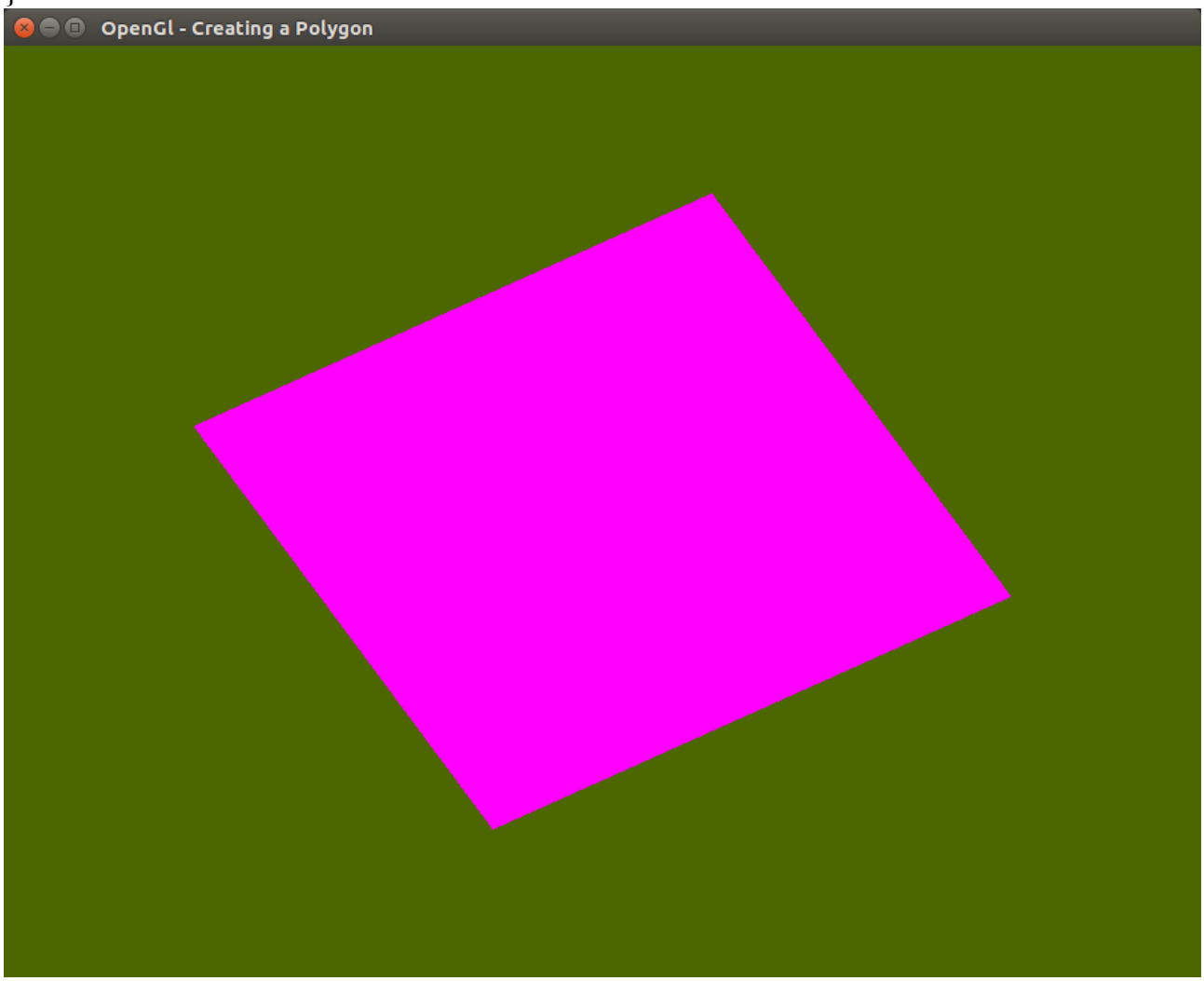
```
#include "GL/freeglut.h"
#include "GL/gl.h"
void drawTriangle(){
glClearColor(0.3,0.4,0.0,0.0);
glClear(GL_COLOR_BUFFER_BIT);
glColor3f(1.0,0.0,1.0);
glOrtho(-1.0,1.0,-1.0,1.0,-1.0,1.0);
 glBegin(GL_TRIANGLES);
   glVertex3f(0,1.0,0);
   glVertex3f(0,-1,0);
   glVertex3f(0.7,0.2,0);
   glEnd();
  glFlush();
}


int main(int argc, char **argv){
 glutInit(&argc,argv);
 glutInitDisplayMode(GLUT_SINGLE);
 glutInitWindowSize(900,700);
 glutInitWindowPosition(100,100);
 glutCreateWindow("OpenGl - Creating a Triangle");
 glutDisplayFunc(drawTriangle);
 glutMainLoop();
 return 0;
}
```



2.creating a polygon

```
#include "GL/freeglut.h"
#include "GL/gl.h"
void drawShape(){
glClearColor(0.3,0.4,0.0,0.0);
glClear(GL_COLOR_BUFFER_BIT);
glColor3f(1.0,0.0,1.0);
glOrtho(-1.0,1.0,-1.0,1.0,-1.0,1.0);
 glBegin(GL_POLYGON);
   glVertex3f(-0.5,-0.5,0);
   glVertex3f(0.5,-0.5,0);
   glVertex3f(0.5,0.5,0);
   glVertex3f(-0.5,0.5,0);
   glEnd();
 glFlush();
}
```

```
int main(int argc, char **argv){
 glutInit(&argc,argv);
 glutInitDisplayMode(GLUT_SINGLE);
 glutInitWindowSize(900,700);
 glutInitWindowPosition(100,100);
 glutCreateWindow("OpenGl - Creating a Polygon");
 glutDisplayFunc(drawShape);
 glutMainLoop();
 return 0;
}
```



3. creating line loop

```
#include "GL/freeglut.h"
#include "GL/gl.h"
float _angle=0.0f;
void drawShape(){
glClearColor(0.3,0.4,0.0,0.0);
glClear(GL_COLOR_BUFFER_BIT);
glRotatef(15,0.0,0.0,1.0);
glColor3f(1.0,0.0,1.0);
glOrtho(-1.0,1.0,-1.0,1.0,-1.0,1.0);
   glBegin(GL_LINE_LOOP);
```

```
   glVertex3f(0,0,0);
   glVertex3f(0,1,0);
   glVertex3f(1,1,0);
   glVertex3f(0.5,0,0);
   glVertex3f(0,0,0);
   glEnd();
  glFlush();
}


int main(int argc, char **argv){
 glutInit(&argc,argv);
 glutInitDisplayMode(GLUT_SINGLE);
 glutInitWindowSize(900,700);
 glutInitWindowPosition(100,100);
 glutCreateWindow("OpenGl - Creating a LineLoop");
 glutDisplayFunc(drawShape);
 glutMainLoop();
 return 0;
}
```

4. creating a linestrip

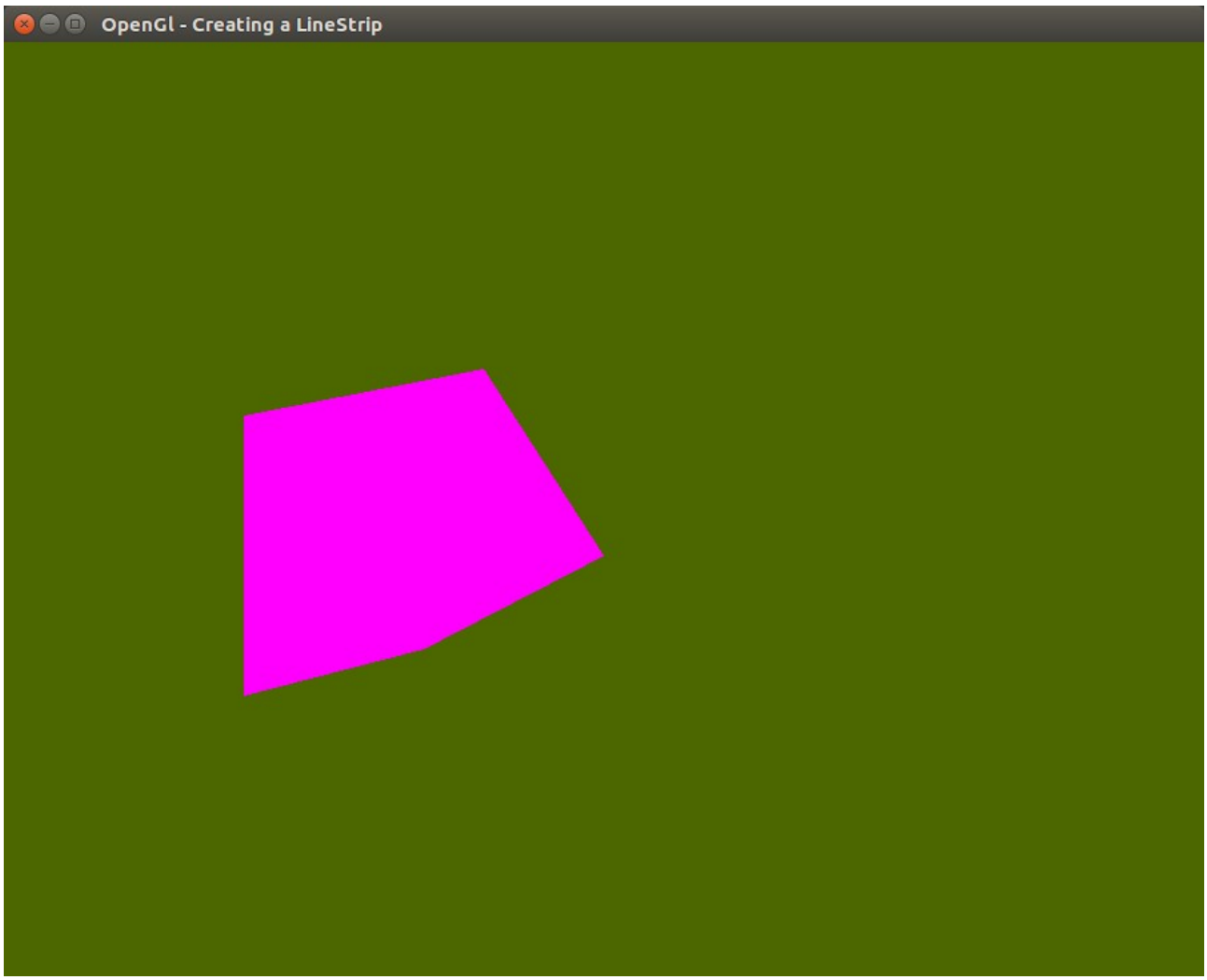
```
#include "GL/freeglut.h"
#include "GL/gl.h"
float _angle=0.0f;
void drawShape(){
glClearColor(0.3,0.4,0.0,0.0);
glClear(GL_COLOR_BUFFER_BIT);
glRotatef(15,0.0,0.0,1.0);
glColor3f(1.0,0.0,1.0);
glOrtho(-1.0,1.0,-1.0,1.0,-1.0,1.0);
  glBegin(GL_LINE_STRIP);
   glVertex3f(0,1.0,0);
   glVertex3f(0.7,0.7,0);
   glVertex3f(0,-1,0);
   glVertex3f(-0.7,-0.7,0);
   glEnd();
  glFlush();
}


int main(int argc, char **argv){
 glutInit(&argc,argv);
 glutInitDisplayMode(GLUT_SINGLE);
 glutInitWindowSize(900,700);
 glutInitWindowPosition(100,100);
 glutCreateWindow("OpenGl - Creating a LineStrip");
 glutDisplayFunc(drawShape);
 glutMainLoop();
 return 0;
}
```

5. #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);
glColor3f(1.0,0.0,1.0);
glOrtho(-1.0,1.0,-1.0,1.0,-1.0,1.0);
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(0.35,0.05);
    glVertex2f(0.3,-0.1);
    glVertex2f(0.2,-0.2);
    glVertex2f(0.0,-0.25);
    glVertex2f(-0.2,-0.2);
    glVertex2f(-0.3,-0.1);
   glEnd();
  glFlush();

```
}


int main(int argc, char **argv){
 glutInit(&argc,argv);
 glutInitDisplayMode(GLUT_SINGLE);
 glutInitWindowSize(900,700);
 glutInitWindowPosition(100,100);
 glutCreateWindow("OpenGl - Creating a Triangle Fan");
 glutDisplayFunc(drawTriangle);
 glutMainLoop();
 return 0;
}
```

6. creating a triangle strip

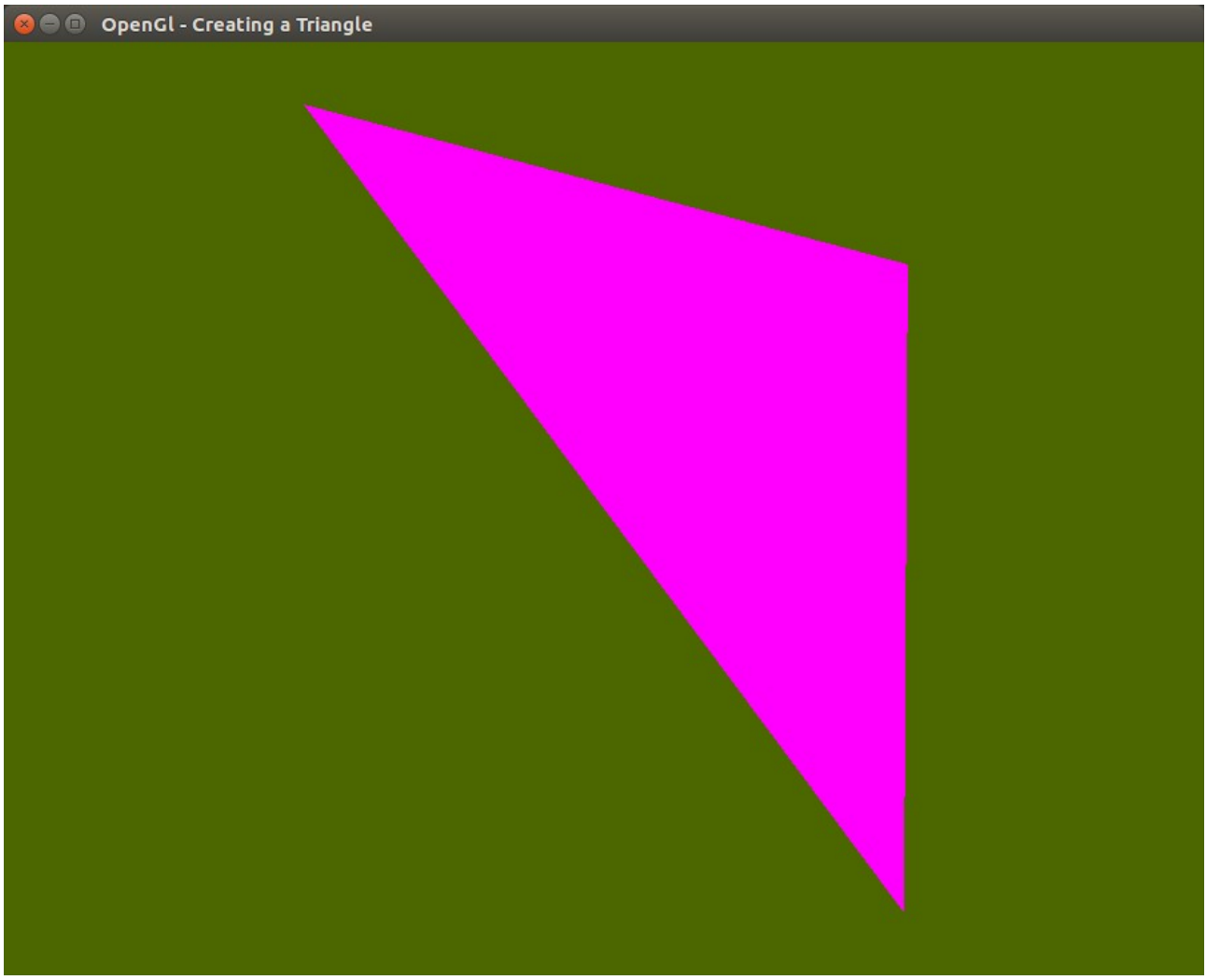
```
#include "GL/freeglut.h"
#include "GL/gl.h"
float _angle=0.0f;
void drawTriangle(){
glClearColor(0.3,0.4,0.0,0.0);
glClear(GL_COLOR_BUFFER_BIT);
glColor3f(1.0,0.0,1.0);
glOrtho(-1.0,1.0,-1.0,1.0,-1.0,1.0);
   glBegin(GL_TRIANGLE_STRIP);
   glVertex2f(-0.6,-0.4);
   glVertex2f(-0.6,0.2);
   glVertex2f(-0.3,-0.3);
   glVertex2f(-0.2,0.3);
   glVertex2f(0.0,-0.1);
   glEnd();
  glFlush();
}


int main(int argc, char **argv){
 glutInit(&argc,argv);
 glutInitDisplayMode(GLUT_SINGLE);
 glutInitWindowSize(900,700);
 glutInitWindowPosition(100,100);
 glutCreateWindow("OpenGl - Creating a LineStrip");
 glutDisplayFunc(drawTriangle);
 glutMainLoop();
 return 0;
}
```



7.
self rotating triangle

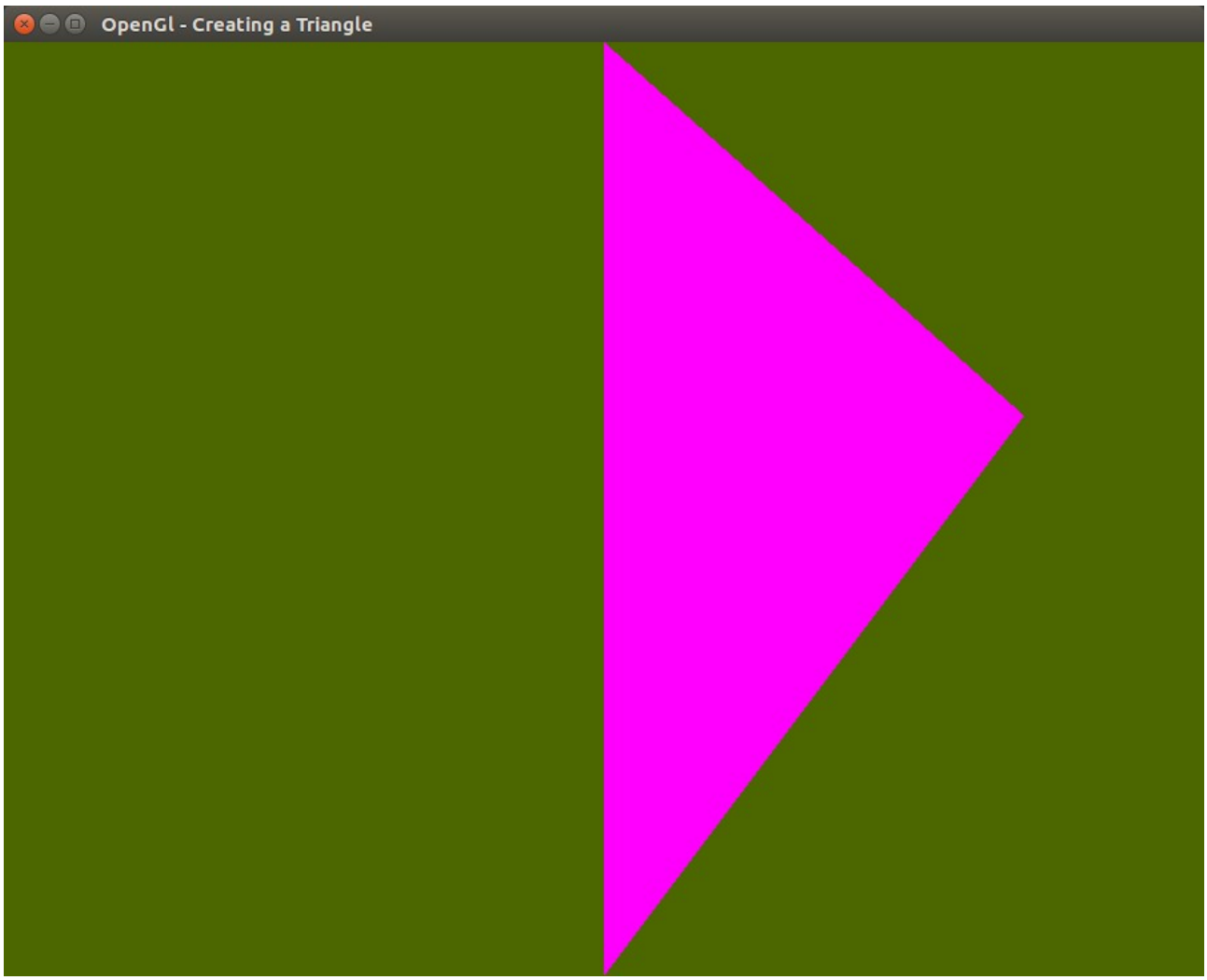
```
#include "GL/freeglut.h"
#include "GL/gl.h"
float _angle=0.0f;
void drawShape(){
glClearColor(0.3,0.4,0.0,0.0);
glClear(GL_COLOR_BUFFER_BIT);
glRotatef(_angle,0.0,0.0,1.0);//constant value for self rotation
glColor3f(1.0,0.0,1.0);
glOrtho(-1.0,1.0,-1.0,1.0,-1.0,1.0);
 glBegin(GL_TRIANGLES);
   glVertex3f(0,1.0,0);
   glVertex3f(0,-1,0);
   glVertex3f(0.7,0.2,0);
   glEnd();
 glFlush();
}

void update(int 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(150, update, 0);
}
int main(int argc, char **argv){
 glutInit(&argc,argv);
 glutInitDisplayMode(GLUT_SINGLE);
 glutInitWindowSize(900,700);
 glutInitWindowPosition(100,100);
 glutCreateWindow("OpenGl - Creating a Triangle");
 glutDisplayFunc(drawShape);
 glutTimerFunc(50, update, 0);//self rotation
 glutMainLoop();
 return 0;
}
```



8 . key press rotation

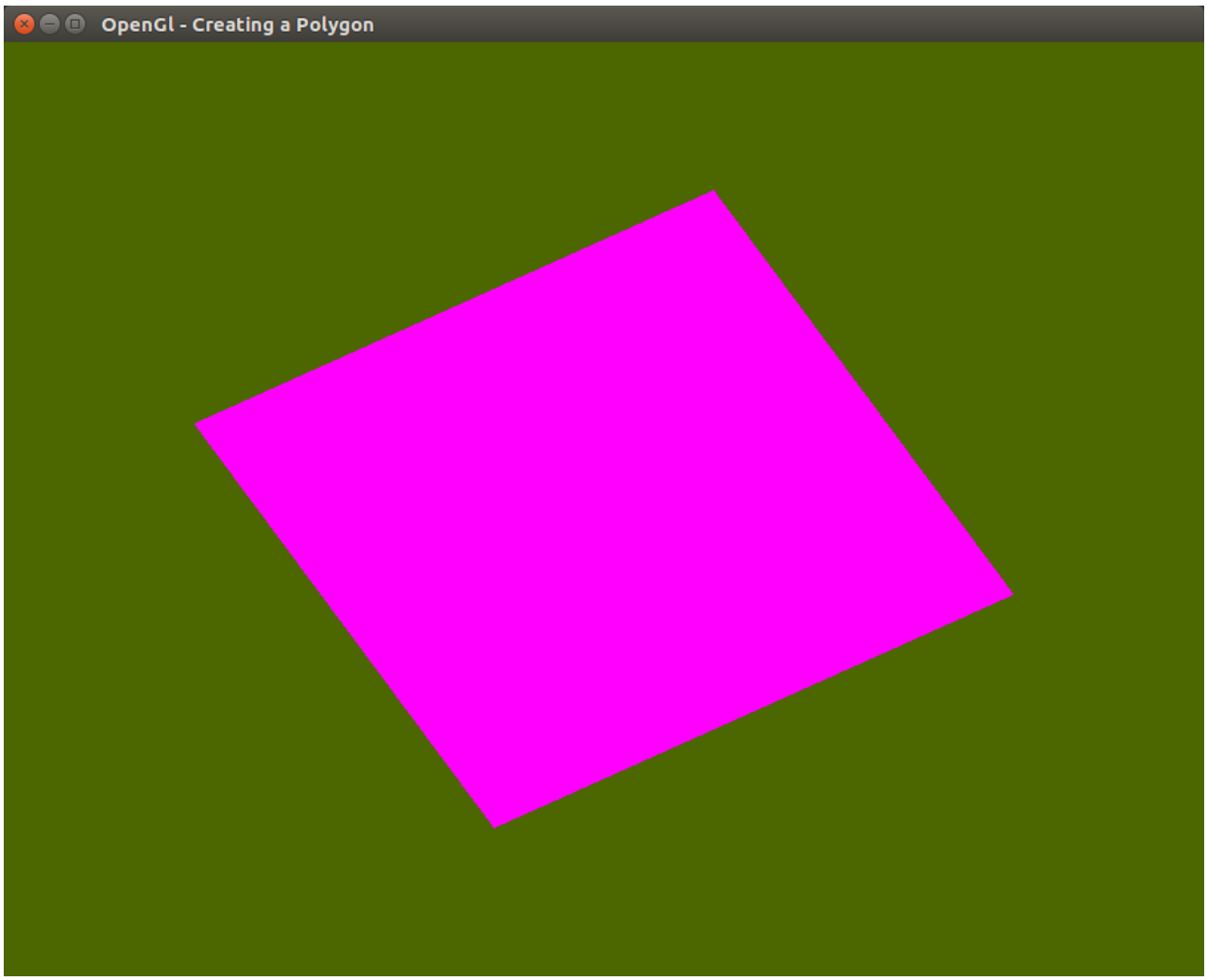
```
#include "GL/freeglut.h"
#include "GL/gl.h"
float _angle=0.0f;
void drawTriangle(){
glClearColor(0.3,0.4,0.0,0.0);
glClear(GL_COLOR_BUFFER_BIT);
glRotatef(_angle,0.0,0.0,1.0);//constant value for self rotation
glColor3f(1.0,0.0,1.0);
glOrtho(-1.0,1.0,-1.0,1.0,-1.0,1.0);
 glBegin(GL_TRIANGLES);
   glVertex3f(0,1.0,0);
   glVertex3f(0,-1,0);
   glVertex3f(0.7,0.2,0);
   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(900,700);
 glutInitWindowPosition(100,100);
 glutCreateWindow("OpenGl - Creating a Triangle");
 glutDisplayFunc(drawTriangle);
 glutSpecialFunc(keyPress);
 glutMainLoop();
 return 0;
}
```



9. self rotating polygon

```
#include "GL/freeglut.h"
#include "GL/gl.h"
float _angle=0.0f;
void drawShape(){
glClearColor(0.3,0.4,0.0,0.0);
glClear(GL_COLOR_BUFFER_BIT);
glRotatef(_angle,0.0,0.0,1.0);//constant value for self rotation
glColor3f(1.0,0.0,1.0);
glOrtho(-1.0,1.0,-1.0,1.0,-1.0,1.0);
 glBegin(GL_POLYGON);
   glVertex3f(-0.5,-0.5,0);
   glVertex3f(0.5,-0.5,0);
   glVertex3f(0.5,0.5,0);
   glVertex3f(-0.5,0.5,0);
   glEnd();
 glFlush();
}
```

```
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(150, update, 0);
}
int main(int argc, char **argv){
 glutInit(&argc,argv);
 glutInitDisplayMode(GLUT_SINGLE);
 glutInitWindowSize(900,700);
 glutInitWindowPosition(100,100);
 glutCreateWindow("OpenGl - Creating a Triangle");
 glutDisplayFunc(drawShape);
 glutTimerFunc(50, update, 0);//self rotation
 glutMainLoop();
 return 0;
}
```

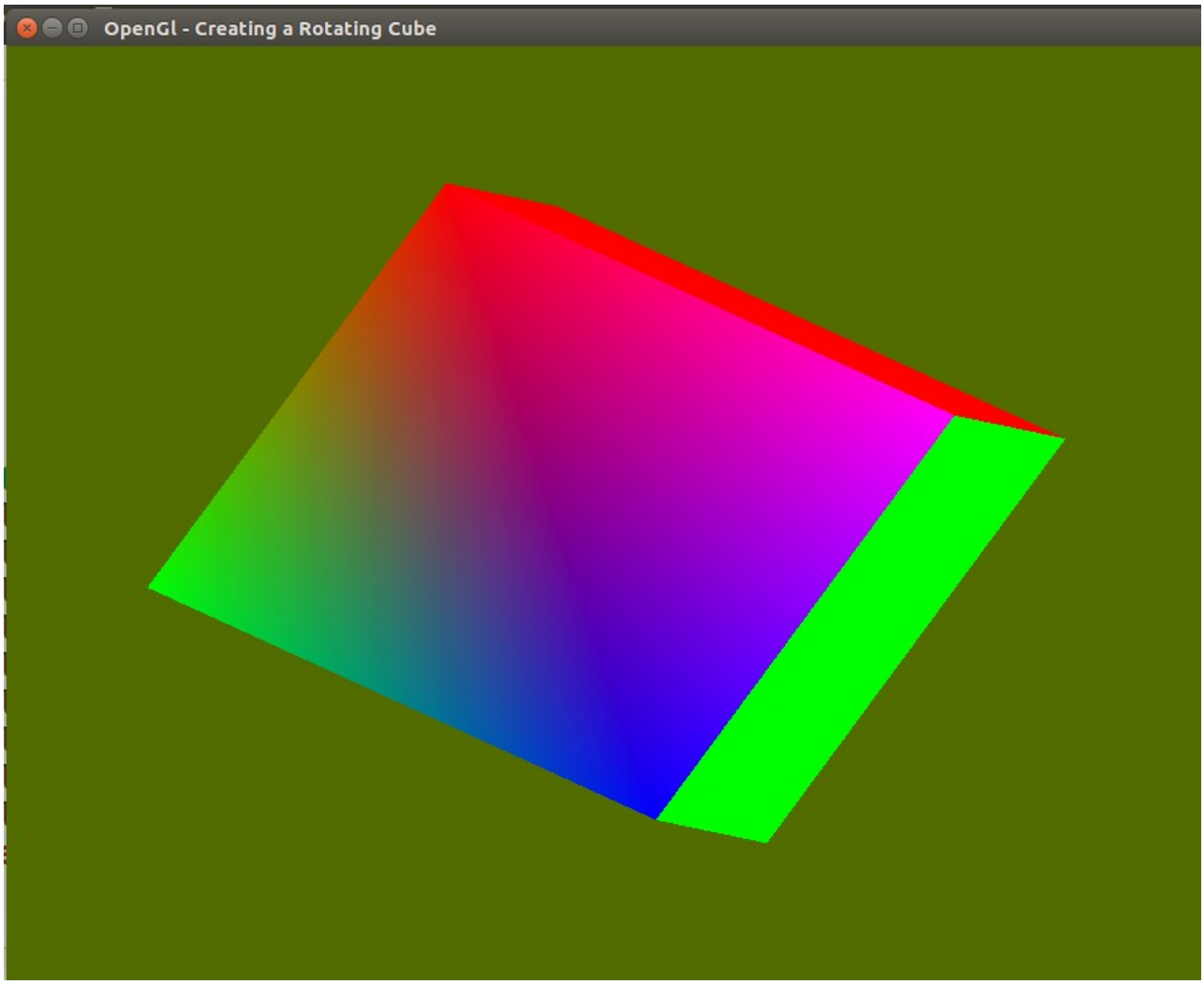


10. self rotating cube

```
#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;
}
```



11.
Beizer curve

```
<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>
```