**学号：E51814014 姓名：吴振龙**

**实验2 基本图元的绘制及图形变换**

1. **实验目的**
2. 掌握二维图形点、线、多边形的绘制方法
3. 熟悉OpenGL中多边形的绘制及填充方法。用不同的方式绘制多边形。

3. 学习掌握OpenGL中图形的变换的方式和相关函数。

4. 学会查文档，自学。

1. **实验原理**

点函数glBegin(GLenum mode)标志描述一个几何图元的顶点列表的开始，其参数mode表示几何图元的描述类型。函数glEnd()标志顶点列表的结束。所有类型及说明见表2-1所示。

1. **表2-1 几何图元类型和说明**

|  |  |
| --- | --- |
| **类型** | **说明** |
| GL\_POINTS | 单个顶点集 |
| GL\_LINES | 多组双顶点线段 |
| GL\_POLYGON | 单个简单填充凸多边形 |
| GL\_TRAINGLES | 多组独立填充三角形 |
| GL\_QUADS | 多组独立填充四边形 |
| GL\_LINE\_STRIP | 不闭合折线 |
| GL\_LINE\_LOOP | 闭合折线 |
| GL\_TRAINGLE\_STRIP | 线型连续填充三角形串 |
| GL\_TRAINGLE\_FAN | 扇形连续填充三角形串 |
| GL\_QUAD\_STRIP | 连续填充四边形串 |
|  | |

glMatrixMode(GL\_MODELVIEW);设置矩阵模式为模型变换模式，表示在世界坐标系下；

glMatrixMode(GL\_PROJECTION);设置矩阵模式为投影变换模式；

加载单位阵：

glLoadIdentity()

右乘变换矩阵:

glRotatef(theta, vx, vy, vz)； //沿逆时针方向绕过原点和点(vx, vy, vz)的轴旋转物体theta度，原点为旋转不动点；

glTranslatef(dx, dy, dz)； //平移对象，位移向量是(x, y, z)

glScalef( sx, sy, sz)；//依据比例因子sx,sy,sz对物体进行缩放或反射，原点为不动点；

每个函数的参数还可以是d(double)类型

1. **实验内容**

1. 发挥想象，编程实现在窗口中绘制若干图形及设置它们的属性，并回答问题：

（1）绘制若干个点，分别设置每个点的颜色和大小。

代码：

*def* points\_display():

    # glViewport(100, 100, 50, 50)

    glClear(GL\_COLOR\_BUFFER\_BIT)

    glPointSize(10)

    glBegin(GL\_POINTS)

    glColor3f(255, 165, 0)

    glVertex3f(-0.8, -0.7, 1.0)

    glEnd()

    glPointSize(20)

    glBegin(GL\_POINTS)

    glColor3f(0, 255, 0)

    glVertex3f(-0.0, -0.7, 1.0)

    glEnd()

    glPointSize(30)

    glBegin(GL\_POINTS)

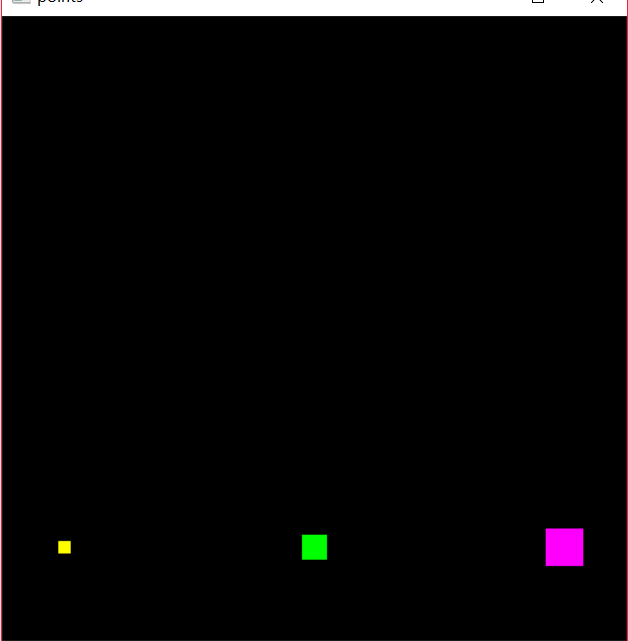
    glColor3f(255, 0, 255)

    glVertex3f(0.8, -0.7, 1.0)

    glEnd()

    glFlush()

实现效果：



思考题1：glPointSize()可以放在glBegin()和glEnd()函数对之间吗？

不能

（2）绘制若干条直线段。设置线段的颜色，线型和线宽。

思考题1：以下命令会显示什么样的线段?

glEnable(GL\_LINE\_STIPPLE)；

glColor3f(0.0, 1.0, 0.0);

glLineWidth(2.0);

glLineStipple(3, 0xcccc);

glDisable(GL\_LINE\_STIPPLE)；

思考题2：以下命令会显示什么样的线段?

glShadeModel(GL\_SMOOTH);

glBegin(GL\_LINES);

glColor3f(1.0, 0.0, 0.0);

glVertex2f(0.0, 0.5);

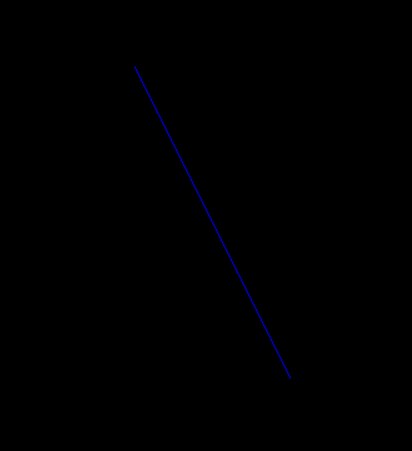
glColor3f(0.0, 0.0, 1.0);

glVertex2f(0.5, -0.5);

glEnd();



将glShadeModel(GL\_SMOOTH)改为glShadeModel(GL\_FLAT)，其余命令不变，结果又会如何？



（3）绘制若干个多边形。设置多边形的颜色，显示模式，隐藏某些边。

代码：

*def* polygon\_display():

    glClear(GL\_COLOR\_BUFFER\_BIT)

    glPolygonMode(GL\_FRONT, GL\_FILL)

    glBegin(GL\_POLYGON)

    glColor3f(0, 191, 255)

    glVertex3f(-0.8, -0.8, 0.0)

    glColor3f(255, 0, 255)

    glVertex3f(-0.2, -0.8, 0.0)

    glColor3f(255, 255, 0)

    glVertex3f(-0.5, -0.2, 0.0)

    glEnd()

    glPolygonMode(GL\_FRONT, GL\_FILL)

    glBegin(GL\_LINE\_STRIP)

    glColor3f(0, 191, 255)

    glVertex3f(0.8, 0.8, 0.0)

    glColor3f(255, 0, 255)

    glVertex3f(0.2, 0.8, 0.0)

    glColor3f(255, 255, 0)

    glVertex3f(0.5, 0.2, 0.0)

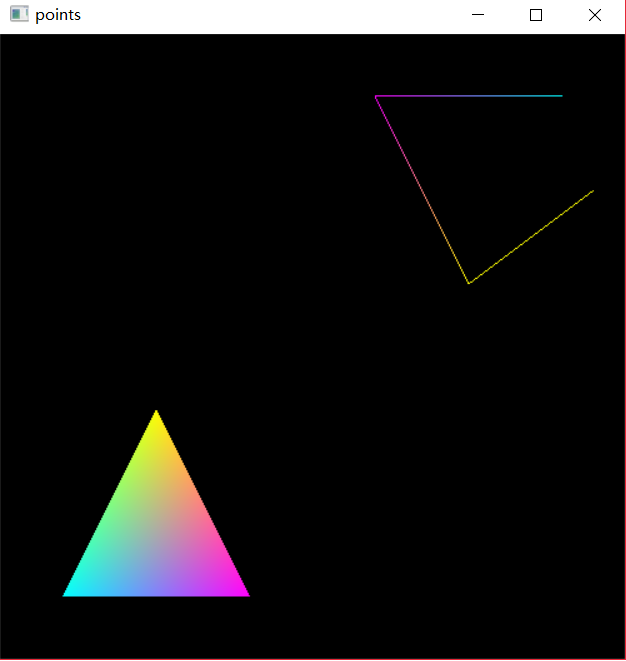
    glColor3f(255, 69, 0)

    glVertex3f(0.9, 0.5, 0.0)

    glEnd()

glFlush()

实现效果：



思考题1：以下命令会显示什么样的多边形?

glShadeModel(GL\_SMOOTH);

glBegin(GL\_POLYGON);

glColor3f(1.0, 0.0, 0.0);

glVertex2f(-0.5, -0.5);

glColor3f(0.0, 1.0, 0.0);

glVertex2f(-0.5, 0.5);

glColor3f(0.0, 0.0, 1.0);

glVertex2f(0.5, 0.5);

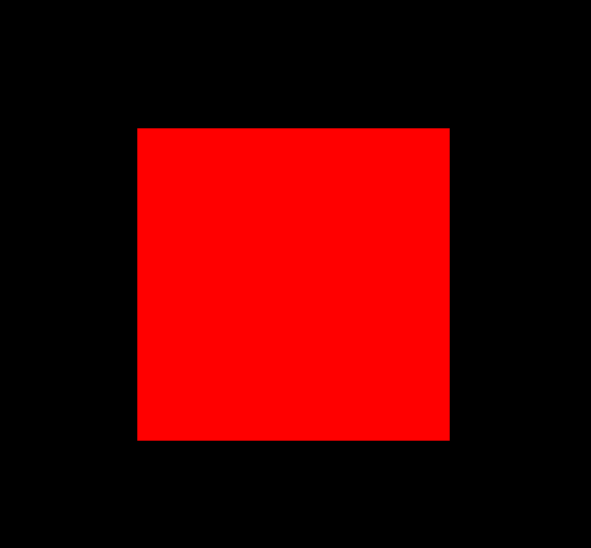
glColor3f(1.0, 1.0, 0.0);

glVertex2f(0.5, -0.5);

glEnd();

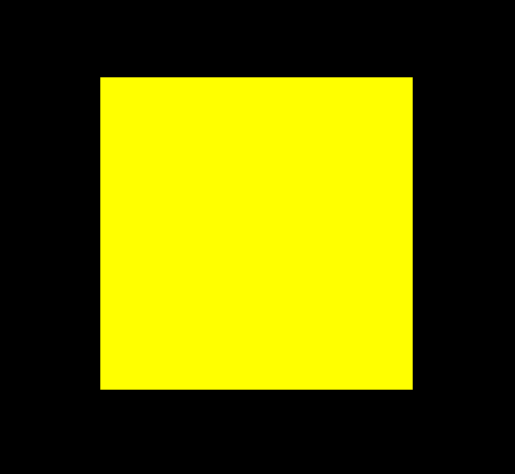


将glShadeModel(GL\_SMOOTH)改为glShadeModel(GL\_FLAT)，其余命令不变，结果又会如何？

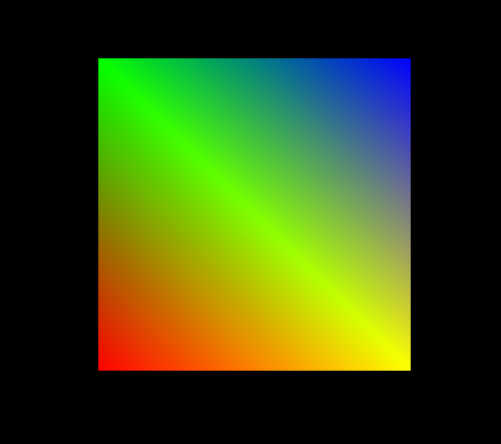


将glBegin(GL\_POLYGON)改为glBegin(GL\_QUADS)，分别尝试在SMOOTH和FLAT模式下的结果。

FLAT模式下：



SMOOTH模式下：



思考题2：以下程序段分别会显示什么样的多边形?

（a）

glPolygonMode( GL\_FRONT\_AND\_BACK, GL\_FILL);

glColor3f(1.0, 0.0, 0.0);

glBegin(GL\_POLYGON);

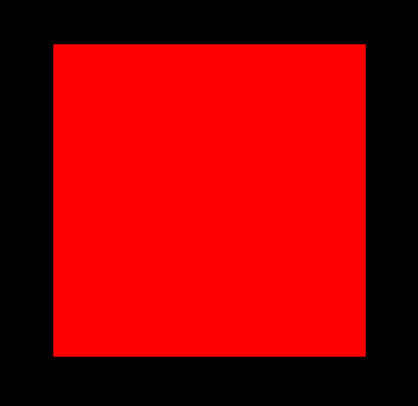
glVertex2f(-0.5, -0.5);

glVertex2f(-0.5, 0.5);

glVertex2f(0.5, 0.5);

glVertex2f(0.5, -0.5);

glEnd();



（b）

glPolygonMode( GL\_FRONT\_AND\_BACK, GL\_LINE);

glColor3f(1.0, 1.0, 0.0);

glBegin(GL\_POLYGON);

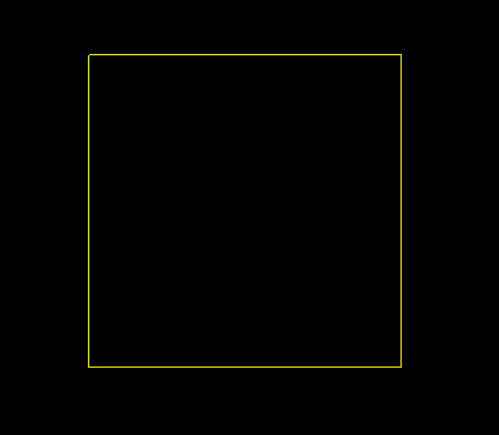
glVertex2f(-0.5, -0.5);

glVertex2f(-0.5, 0.5);

glVertex2f(0.5, 0.5);

glVertex2f(0.5, -0.5);

glEnd();



（c）

glPolygonMode( GL\_FRONT\_AND\_BACK, GL\_POINT);

glColor3f(1.0, 1.0, 0.0);

glBegin(GL\_POLYGON);

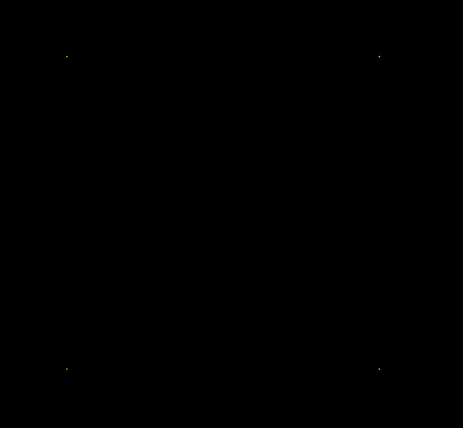
glVertex2f(-0.5, -0.5);

glVertex2f(-0.5, 0.5);

glVertex2f(0.5, 0.5);

glVertex2f(0.5, -0.5);

glEnd();



思考题3：下面多边形的哪些边不可见?

glPolygonMode( GL\_FRONT\_AND\_BACK, GL\_LINE);

glColor3f(1.0, 1.0, 0.0);

glBegin(GL\_POLYGON);

glVertex2f(-0.5, -0.5);

glEdgeFlag(GL\_FALSE);

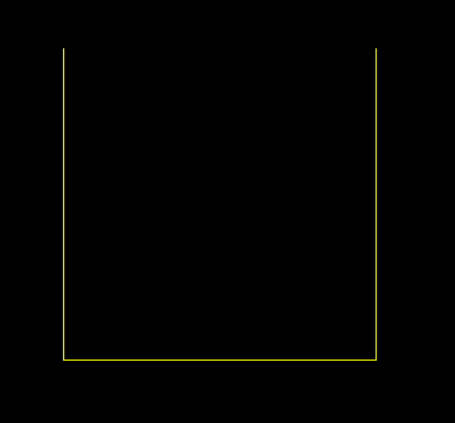
glVertex2f(-0.5, 0.5);

glEdgeFlag(GL\_TRUE);

glVertex2f(0.5, 0.5);

glVertex2f(0.5, -0.5);

glEnd();



2. 阅读并实现如下示例程序，学习掌握多边形的多种绘制方式。

#include<windows.h>

#include<GL/glut.h>

void myinit(void)

{

/\*设置背景颜色\*/

glClearColor(1.0,1.0,1.0,0.0);

}

void DrawPolygon()

{ //绘制多边形

glBegin(GL\_POLYGON);

glVertex2f(20.0,10.0);

glVertex2f(60.0,30.0);

glVertex2f(70.0,45.0);

glVertex2f(40.0,75.0);

glVertex2f(10.0,60.0);

glEnd();

}

void display(void)

{

GLubyte fly[]={

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x03,0x80,0x01,0xC0,0x06,0xC0,0x03,0x60,

0x04,0x60,0x06,0x20,0x04,0x30,0x0C,0x20,

0x04,0x18,0x18,0x20,0x04,0x0C,0x30,0x20,

0x04,0x06,0x60,0x20,0x44,0x03,0xC0,0x22,

0x44,0x01,0x80,0x22,0x44,0x01,0x80,0x22,

0x44,0x01,0x80,0x22,0x44,0x01,0x80,0x22,

0x44,0x01,0x80,0x22,0x44,0x01,0x80,0x22,

0x66,0x01,0x80,0x66,0x33,0x01,0x80,0xCC,

0x19,0x81,0x81,0x98,0x0C,0xC1,0x83,0x30,

0x07,0xe1,0x87,0xe0,0x03,0x3f,0xfc,0xc0,

0x03,0x31,0x8c,0xc0,0x03,0x33,0xcc,0xc0,

0x06,0x64,0x26,0x60,0x0c,0xcc,0x33,0x30,

0x18,0xcc,0x33,0x18,0x10,0xc4,0x23,0x08,

0x10,0x63,0xC6,0x08,0x10,0x30,0x0c,0x08,

0x10,0x18,0x18,0x08,0x10,0x00,0x00,0x08};

glClear(GL\_COLOR\_BUFFER\_BIT);

/\*设置线段的颜色\*/

glColor3f(0.0,0.0,0.0);

/\*第一个多边形采用点绘制\*/

glPolygonMode(GL\_FRONT,GL\_POINT);

glTranslatef(20.0,10.0,0.0); //平移

DrawPolygon();

/\*第二个多边形采用线绘制\*/

glPolygonMode(GL\_FRONT,GL\_LINE);

glTranslatef(90.0,0.0,0.0); //平移

DrawPolygon();

/\*第三个多边形采用填充模式绘制\*/

glPolygonMode(GL\_FRONT,GL\_FILL);

glTranslatef(90.0,0.0,0.0); //平移

DrawPolygon();

/\*第四个多边形为逆时针方向，并舍弃其背面\*/

glFrontFace(GL\_CW); //指定多边形的正面，GL\_CW指定顶点顺时针方向为多边形的正面

glCullFace(GL\_BACK); //指明何种多边形在转换成屏幕坐标时要删除，GL\_BACK背面

glEnable(GL\_CULL\_FACE); //激活GL\_CULL\_FACE，即glCullFace

glTranslatef(-190.0,80.0,0.0);

DrawPolygon();

/\*第五个多边形采用点划模式绘制\*/

glFrontFace(GL\_CCW); //GL\_CCW指定顶点的逆时针方向为多边形的反面

glEnable(GL\_POLYGON\_STIPPLE); //激活GL\_POLYGON\_STIPPLE

glPolygonStipple(fly);

glTranslatef(90.0,80.0,0.0);

DrawPolygon();

glFlush();

}

void myreshape(int w,int h)

{ //设置视图

glViewport(0,0,(GLsizei)w,(GLsizei)h);

// 设置矩阵的当前模式为投影

glMatrixMode(GL\_PROJECTION);

// 加载单位矩阵

glLoadIdentity();

// 绘制裁切面

gluOrtho2D(0.0,(GLdouble)w,0.0,(GLdouble)h);

}

int main(int argc,char\*\*argv)

{

glutInit(&argc,argv);

// 设置图形显示模式为RGBA和单缓冲模式

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

// 设置窗口大小

glutInitWindowSize(300,300);

// 设置窗口位置

glutInitWindowPosition(150,150);

// 新建窗口

glutCreateWindow("Polygon");

myinit();

// 回调绘图函数

glutDisplayFunc(display);

// 再次绘图

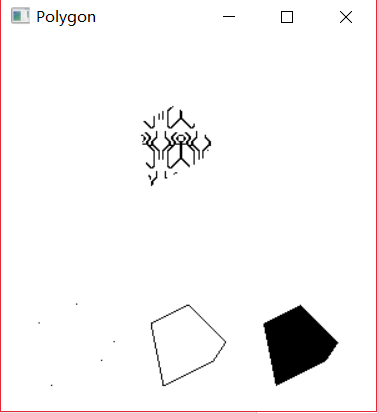
glutReshapeFunc(myreshape);

// 执行glut绘图程序

glutMainLoop();

return 0;

}



3. 对顶点为V1V2V3的三角形进行平移、旋转、比例、错切等基本变换。三角形顶点的坐标为：V1(20.0,20.0)，V2(80.0,30.0)，V3(50.0,70.0)。

代码：

*def* transfer():

    glOrtho(-100, 100, -100 ,100, -100, 100)

    glClear(GL\_COLOR\_BUFFER\_BIT)

    glColor3f(1.0, 1.0, 1.0)

    glBegin(GL\_LINES)

    glVertex2f(-100.0, 0)

    glVertex2f(100.0, 0.0)

    glEnd()

    glBegin(GL\_LINES)

    glVertex2f(0.0, 100.0)

    glVertex2f(0.0, -100.0)

    glEnd()

    """ 绘制原始三角形 """

    glMatrixMode(GL\_MODELVIEW)

    glLoadIdentity()

    glColor3f(255, 0, 255)

    triangle()

    """ 旋转变换三角形 """

    glRotatef(200.0, 0.0, 0.0, 1.0)

    glColor3f(255, 69, 0)

    triangle()

    """ 平移变化三角形 """

    glLoadIdentity()

    glColor3f(0, 255, 80)

    glTranslatef(-100, 0.0, 0.0)

    triangle()

    """ 错切变换三角形 """

    m = np.array([1.0, 0.0, 1., 0.,

                  0.0, 1.0, 1., 0.,

                  0., 0., 1., 0.,

                  0.0, 0., 0., 1.], np.float64)

    glLoadIdentity()

    glMultMatrixf(m)

    glColor3f(50, 50, 50)

    triangle()

    """ 比例变换三角形 """

    glLoadIdentity()

    glColor3ub(255, 228, 181)

    glScaled(0.5, 0.5, 1.0)

    triangle()

    glFlush()

实现效果：

紫色三角形为原始图形，白色为错切结果，浅蓝色为平移结果，黄色为旋转结果，肉色为比例变换结果

