# 实验2 多边形扫描转换与二维图形变换

## 一、实验目的与要求：

1. 掌握直线、圆、椭圆、多边形等基本图形的扫描转换方法和区域填充算法；
2. 了解反走样技术基本原理和OpenGL实现反走样方法；
3. 掌握二维图形基本几何变换方法；
4. 了解OpenGL实现图形变换方法。

## 二、实验内容与步骤：

### 1 任意斜率的直线扫描转换

// 中点Bresenham算法绘制任意斜率的直线段

void MidBhline2(int x0, int y0, int x1, int y1)

{

int P[2]={x0, y0}, s[2]={1, 1}, dx=x1-x0, dy=y1-y0, k=0; // k为主方向

if(dx<0) s[0]=-1, dx=-dx;

if(dy<0) s[1]=-1, dy=-dy;

if(dy>dx){k=dy, dy=dx, dx=k, k=1;}

int d=dx-2\*dy, d1=2\*dx-2\*dy, d2=-2\*dy;

glBegin(GL\_POINTS);

for (int i=0; i<=dx; i++)

{

glVertex2i(P[0], P[1]);

P[k]+=s[k];

if(d<0)

{

P[1-k] += s[1-k];

d+=d1;

}

else

{

d+=d2;

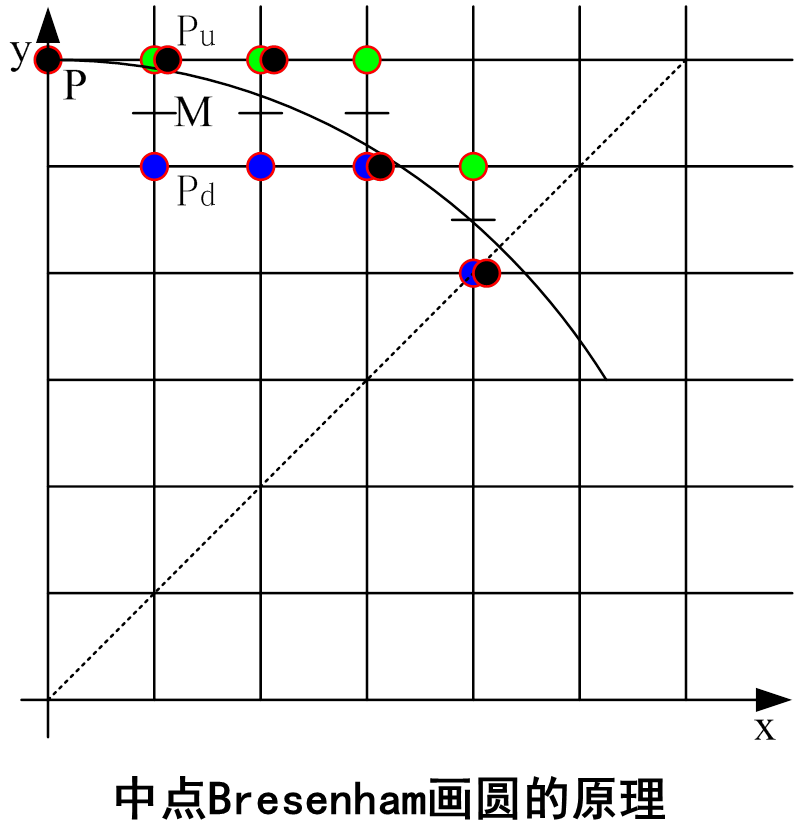
}

}

glEnd();

}

### 2 圆/椭圆的扫描转换

// 中点Bh算法绘制圆

void MidBhcircle( int r){

int x=0, y=r, d=1-r;

glBegin(GL\_POINTS);

while ( x<=y ){

glVertex2i ( x, y ); glVertex2i ( y, x ); // 八分画圆

glVertex2i ( -y, x ); glVertex2i ( -x, y );

glVertex2i ( -x, -y ); glVertex2i ( -y, -x );

glVertex2i ( y, -x ); glVertex2i ( x, -y );

if ( d<0)

d+=2\*x+3;

else{

d+=2\*(x-y)+5;

y--;

}

x++;

}

glEnd();

}

*// 中点Bh算法绘制椭圆*

*void MidBhellipse ( int a, int b){*

*int x, y;*

*float d1, d2;*

*x=0; y=b;*

*d1=b\*b+a\*a\*(-b+0.25);*

*glBegin(GL\_POINTS);*

*glVertex2i ( x, y ); glVertex2i ( -x, -y );*

*glVertex2i ( -x, y ); glVertex2i ( x, -y );*

*while ( b\*b\*(x+1)<a\*a\*(y-0.5)){*

*if ( d1<=0 ) {*

*d1+=b\*b\*(2\*x+3);*

*x++;*

*}*

*else{*

*d1+=b\*b\*(2\*x+3)+a\*a\*(-2\*y+2);*

*x++;*

*y--;*

*}*

*glVertex2i ( x, y ); glVertex2i ( -x, -y );*

*glVertex2i ( -x, y ); glVertex2i ( x, -y );*

*} /\* 上半部\*/*

*d2=b\*b\*(x+0.5)\*(x+0.5)+a\*a\*(y-1)\*(y-1)-a\*a\*b\*b;*

*while ( y>0 ){*

*if ( d2<=0) {*

*d2+=b\*b\*(2\*x+2)+a\*a\*(-2\*y+3);*

*x++; y--;*

*}*

*else {*

*d2+=a\*a\*(-2\*y+3); y--;*

*}*

*glVertex2i ( x, y ); glVertex2i ( -x, -y );*

*glVertex2i ( -x, y ); glVertex2i ( x, -y );*

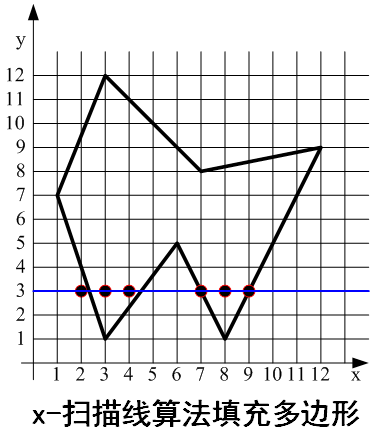
*}*

*glEnd();*

*}*

### 3 多边形的扫描转换

#### 3.1 X扫描线法



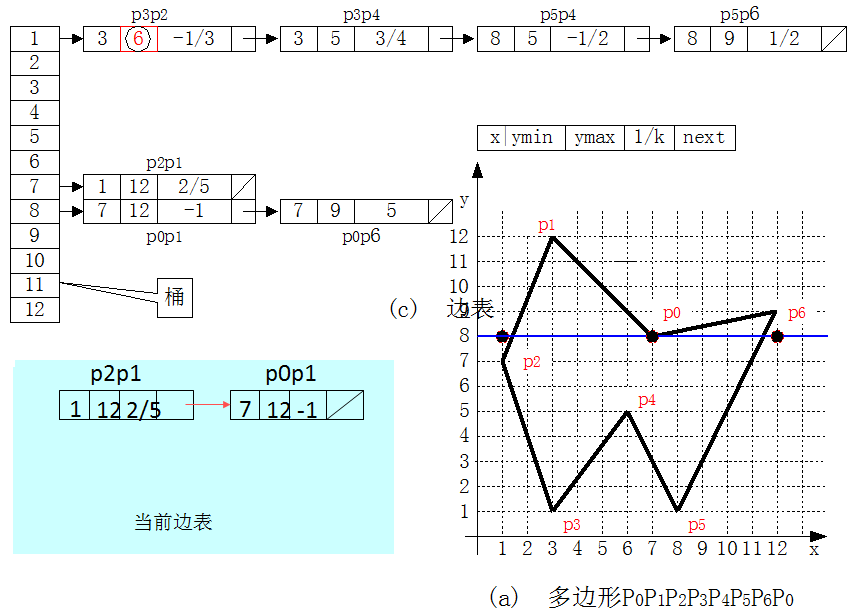
X－扫描线算法——算法步骤：

1.确定多边形所占有的最大扫描线数，得到多边形顶点的最小和最大y值（ymin和ymax）。

2.从y=ymin到y=ymax，每次用一条扫描线进行填充。

3.对一条扫描线填充的过程可分为求交、排序、交点配对、区间填色等四个过程。

#### 3.2 有效边表法



有效边表算法——算法步骤：

(1)初始化：构造边表，AET表置空；

(2)将第一个不空的ET表中的边与AET表合并；

(3)由AET表中取出交点对进行填充。填充之后删除y=ymax的边；

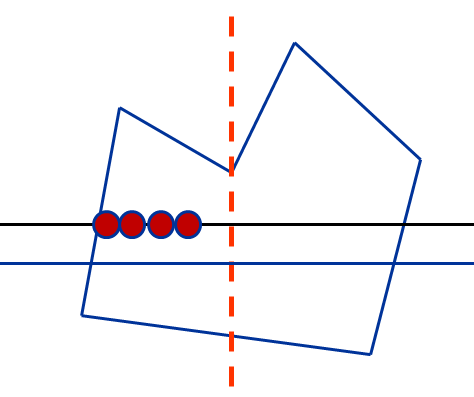
(4)yi+1=yi+1,根据xi+1=xi+1/k计算并修改AET表，同时合并ET表中y=yi+1桶中的边，按次序插入到AET表中，形成新的AET表；

(5)AET表不为空则转(3)，否则结束。

#### 3.3 边缘填充算法→栅栏填充算法→边标志算法

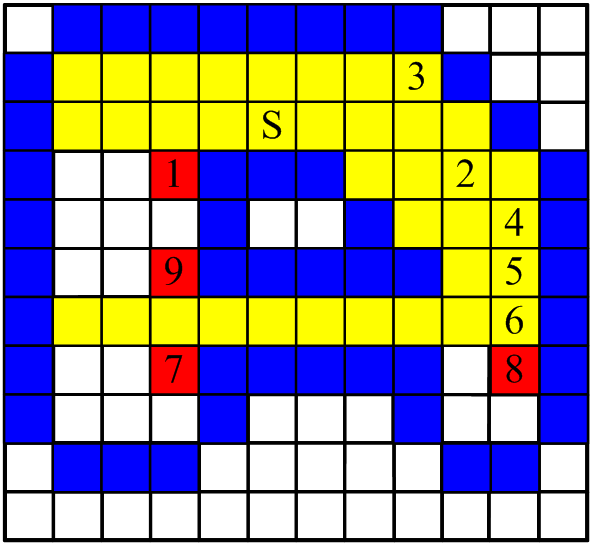
边缘填充算法：按任意顺序处理多边形的每条边。处理时，先求出该边与扫描线的交点，再对扫描线上交点右方的所有象素取反。

栅栏填充算法：利用过多边形顶点且与扫描线垂直的直线作为栅栏，按任意顺序处理多边形的每一条边，但处理每条边与扫描线的交点时，将交点与栅栏之间的象素取反。



边标志算法：①打标记：对多边形的每一条边进行扫描转换，即对多边形边界所经过的象素作一个边界标志。②填充。对每条与多边形相交的扫描线，按从左到右的顺序，逐个访问该扫描线上的象素，根据标志决定象素点的内外状态。

### 4 多边形区域填充



栈结构实现8-连通边界填充算法的算法步骤为：

种子象素入栈；当栈非空时重复执行如下三步操作：

(a)栈顶象素出栈；

(b)将出栈象素置成填充色；

(c)检查出栈象素的8-邻接点，若其中某个像素点不是边界色且未置成多边形色，则把该像素入栈。

扫描线种子填充算法：当给定种子点时，首先填充种子点所在的扫描线上的位于给定区域的一个区段，然后确定与这一区段相通的上下两条扫描线上位于给定区域内的区段，并依次保存下来。反复这个过程，直到填充结束。

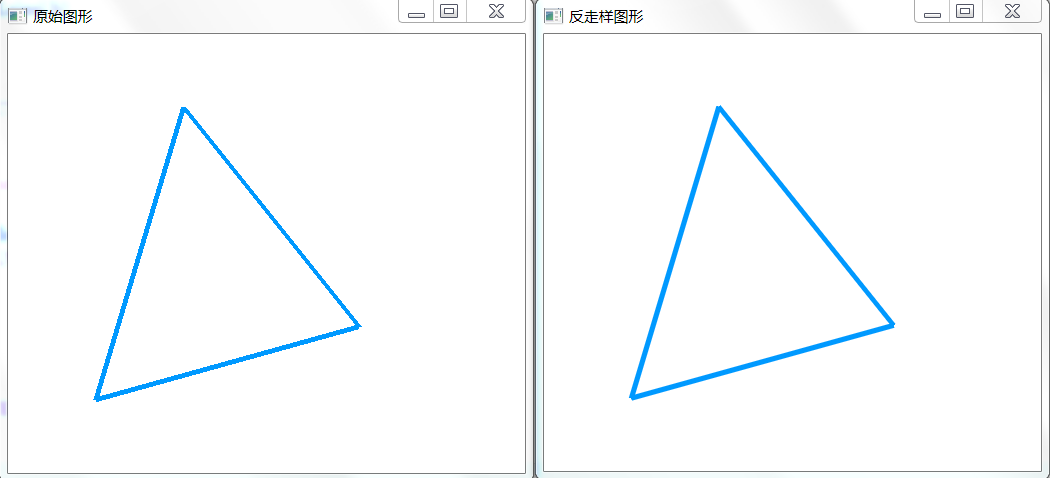
### 5 OpenGL反走样

glEnable(GL\_LINE\_SMOOTH); //使用反走样

glHint(GL\_LINE\_SMOOTH\_HINT, GL\_DONT\_CARE); // 反走样函数

glEnable (GL\_BLEND); //启用混合函数

glBlendFunc(GL\_SRC\_ALPHA, GL\_ONE\_MINUS\_SRC\_ALPHA);



在RGBA颜色模式（A表示透明度）中，已知源像素的颜色值为（Sr，Sg，Sb，Sa），目标像素的颜色值为（Dr，Dg，Db，Da），颜色混合后像素的颜色为：

（*RS*·Sr +*RD*·Dr，*GS*·Sg +*GD*·Dg，*BS*·Sb +*BD*·Db，*AS*·Sa +*AD*·Da）

【示例源码详见antialiasing.cpp】

### 6 二维图形的几何变换



#### 6.1 平移变换



#### 6.2 比例变换



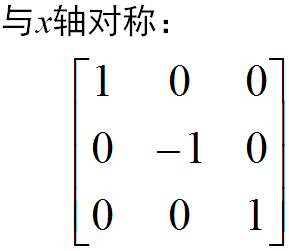
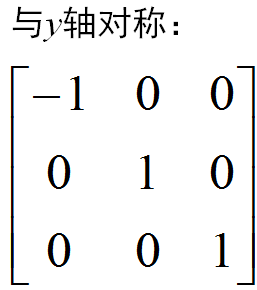
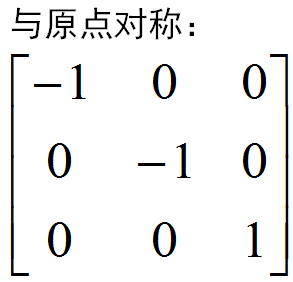
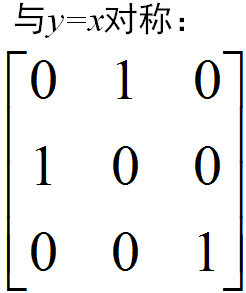
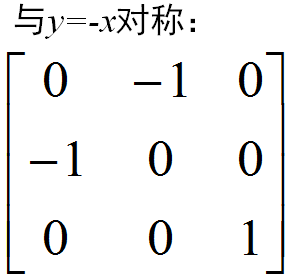
整体比例变换：

#### 6.3 旋转变换

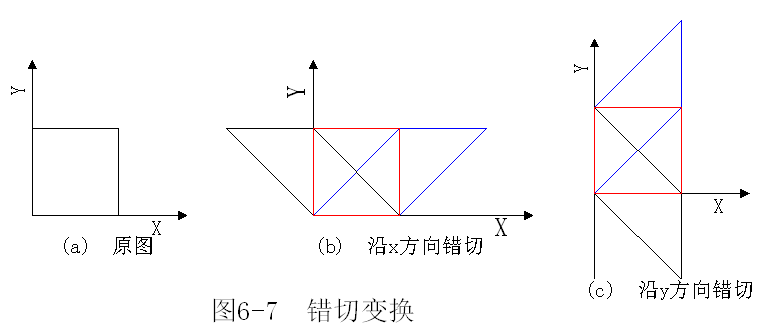


#### 6.4 对称变换



#### 6.5 错切变换





#### 6.6 相对任意参考点的复合变换

相对某个参考点(xF,yF)作二维几何变换，其变换过程为：(分解成基本的几何变换）

(1) 平移(-xF,-yF) 。

(2) 针对原点进行二维几何变换。

(3) 反平移(+xF,+yF) 。

#### 6.7 相对任意方向的复合变换

相对任意方向作二维几何变换，其变换的过程是：

(1) 旋转变换

(2) 针对坐标轴进行二维几何变换；

(3) 反向旋转

#### 6.8 坐标系之间的变换



### 7 OpenGL基本几何变换

* 平移

void glTranslated(f)(GLdouble x，GLdouble y，GLdouble z)；

* 旋转

void glRotated(f)(GLdouble angle，GLdouble x，GLdouble y，GLdouble z )；

* 比例

void glScaled(f)(GLdouble x，GLdouble y，GLdouble z)；

* 模型视图矩阵

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glPushMatrix(); // 模型视图矩阵的入栈

…… // 平移、旋转、缩放等变换

glPopMatrix(); // 出栈

* 获取模型视图矩阵

float mat[16]; // 列优先的矩阵

glGetFloatv(GL\_MODELVIEW\_MATRIX, mat);

glMultMatrixf (mat); // 矩阵乘法

【示例源码详见testCG2.cpp】

## 三、实验练习：

###### 1 编程验证抛物线 100x = y2 的扫描转换算法；

###### 2 编程实现有效边表法的多边形扫描转换；

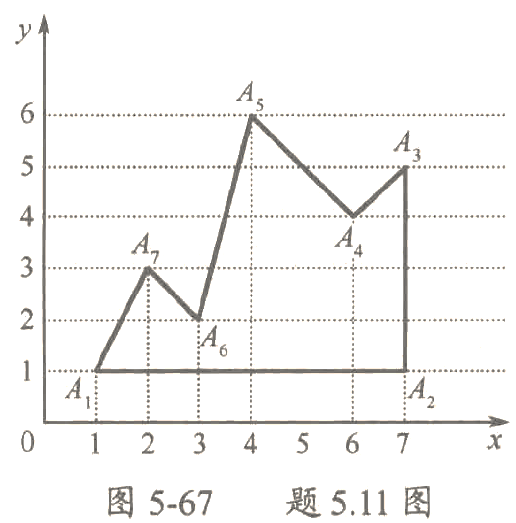
###### 3 编程实现基于种子填充算法的区域填充；

###### 4 基于矩阵运算，编程实现多边形的平移、比例、旋转、对称和错切等二维几何变换；

###### 5 验证教材中的OpenGL例子。

## 四、提交报告及要求

题目：课本p142-T5.11：对如图的多边形采用扫描转换算法（改进的有效边表算法）进行填充，试写出该多边形的ET表和当扫描线y=4时的有效边表AET。[可选：尝试编写有效边表法程序予以验证（为了便于OpenGL显示，建议坐标放大100倍）]



**提交要求说明：**

**1）编写上机报告（Word或pdf文档）并线上提交到作业；**

**2）内容上分为*实验目的、实验原理（模仿课本图5-27写出多边形的边表及有效边表）、实验步骤及源码、实验结果及分析*等四个部分，详细具体，言简意赅；**

**3）*截止日期：3月23日（下周四）前*。**