目录

一、 实验目的

本实验以 visual studio 为实验环境，理解多边形的扫描转换原理、方法，掌握橡皮筋算法和扫描线填充算法，并自行设计添加新的功能，进一步掌握在 VC 集成环境中实现图形算法的方法与过程。

二、 实验内容

程序要实现以下要求：

1. 使用 OpenGL 库
2. 用类（或模版类）来表示数据结构
3. 多文档组织，至少要用头（.h）文件表示数据结构
4. 通过橡皮筋交互输入不同颜色、大小的多边形
5. 清屏重置多边形，以及撤销之前画出的点线
6. 多边形扫描算法中的顶点处理以每条边减去一个像素方法处理
7. 实现自相交多边形、多个多边形的扫描填充
8. 实现图形旋转，并设置旋转方向
9. 可以选取不同的填充模式，如等间隔扫描线填充
10. 通过菜单以及键盘交互
11. 通过文件读出需要交互输入的多边形。

三、 算法实现

1. 多边形填充算法

1.1 数据结构定义

点类，用来存储一个点的 x，y 值：

```
class point
{
public:
    int x;
    int y;
};
```

多边形类，用来存储一个多边形：

```
class polygon
{
public:
    vector<point> p;
    int color;
};
```

边类，包含四个参数。ymax 为边最高点的纵坐标；x 为边最低点的横坐标；dx 为斜率的倒数；Edge* next 指向下一条边：

```
class Edge
{
public:
    int ymax;
    float x;
    float dx;
    Edge* next;
};
```

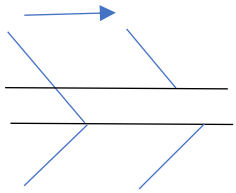
1.2 算法流程

（1） 生成有序的边分类表 ET；
（2） 初始化活化边表 AET；
（3） 扫描线从最低 y 坐标开始，循环下列步骤：
• 如果 ET[y] 非空，则将其中的所有边取出并插入到 AET 中，按 x（若 x 相等则按Δx）递增方向排序；
• 若 AET 非空，将 AET 中的边排序，按顺序两两配对并填色；
• 删去 AET 中满足 y=ymax 的边；
• 对于 AET 中所有边，赋值 x = x + Δx；
• y = y + 1，扫描线上移一个像素；
（4）扫描线达最高点，程序结束。

1.3　具体细节处理

（1）奇点处理

将奇点上面那条边的底部向上缩一个像素，如图所示：



相关处理代码如下：

```
//奇点处理
if (((y0 < y1) && (y1 < y2)) || ((y0 > y1) && (y1 > y2)))
{
    nowy++;
    x += dx;
}
```

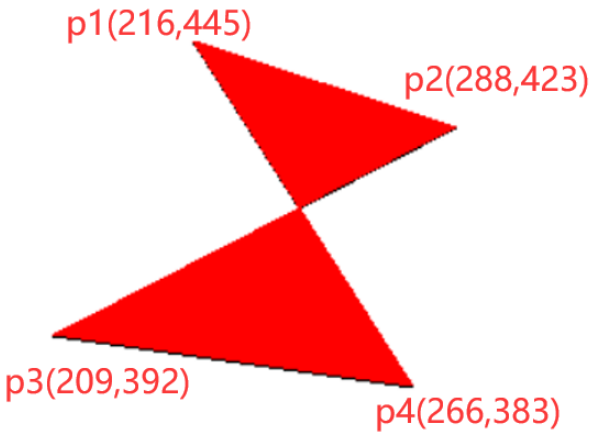
（2）交点处理

每次循环 y+1，都要用 sortAET()函数对 AET 表重新排序，即确保每一对交点都是 x 值小的排在 AET 表的前面。

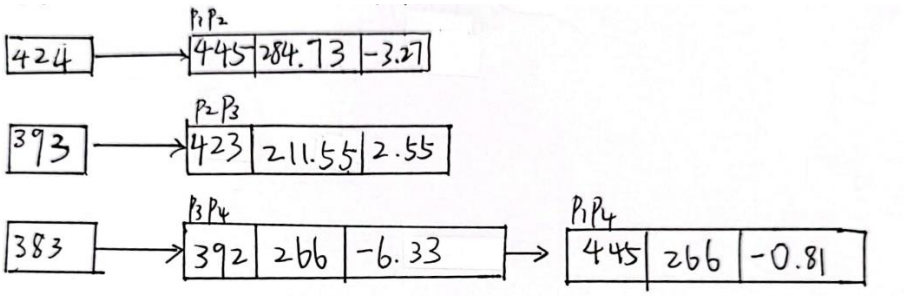
如果不在每轮都进行排序，就无法确保视觉效果上位于图形左侧的那条边排在 AET 表的前面，那么自相交多边形的交点以上部分会填充失败。

（3）写出活性边表的具体变化过程

通过在控制台打印出相应坐标确定有底边表及活性边表的具体变化过程。以如下图形为例进行分析：

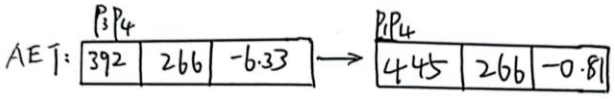


先画出有序边表，其中 p1p2 和 p2p3 的 ymin 都进行了加一操作：

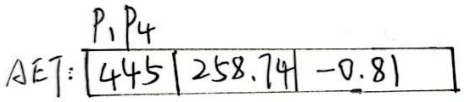


然后画出活性边表的具体变化过程：

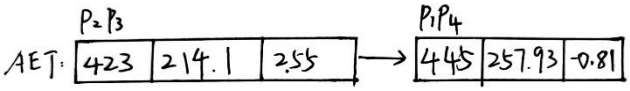
a. y=383 时，p3p4, p1p4 依次加入活性边表 AET 中，如下图:



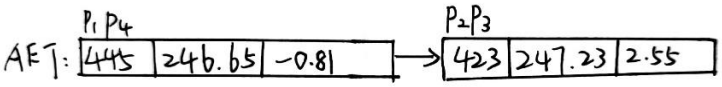
b. y=392 时，y=p3p4 的 Ymax,因此将之从 AET 中删除,此时：



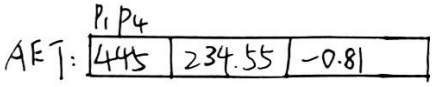
c. y=393 时，p2p3 加入 AET, 并进行排序，将 Xymin 值更小的 p2p3 放在前面:



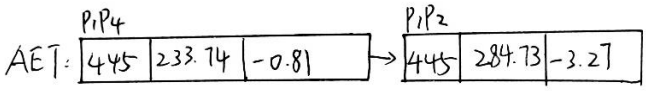
d. y=407 时，p1p2 与 p3p4 相交，依据 Xymin 重新排序后，p1p4 放在前面：



e. y=423 时，y=p2p3 的 Ymax,因此将之从 AET 中删除：



f. y=424 时，p1p2 加入 AET:



g. y=445 时，y=p1p2 和 p1p4 的 Ymax，将这两者均删除，此时 AET 为空：
AET:->

2. 撤销及清屏

（1） 撤销上一个点：取出点集中最新放入的一个元素。

```
case'1':    //返回上一步
    p.pop_back();
    break;
```

（2） 删除正在绘制的多边形：清除点集中所有元素。

```
case'2':    //删除正在绘制的多边形
    p.clear();
    break;
```

（3） 删除上一个已经绘制好的多边形：取出多边形容器中最新放入的一个元素。

```
case 1:// 删除上一个已经绘制好的多边形
    s.pop_back();
    break;
```

（4） 清屏：清除点集中所有元素，并且清空多边形容器。

```
case 2:// 清空画布
    p.clear();
    s.clear();
    glutPostRedisplay();
    break;
```

3. 填充颜色变换

执行如下语句，使 polygon_color 在数字 1，2，3 中变换：

```
case 3://颜色变换
    polygon_color = (polygon_color + 1) % 4;
    break;
```

在多边形填充算法中，通过以下语句，三个不同的数字分别对应到不同的颜色，以实现填充颜色的变换。

```
switch (color)
{
case 1:
    glColor3f(1.0, 0.0, 0.0);
    break;
case 2:
    glColor3f(0.0, 1.0, 0.0);
    break;
case 3:
    glColor3f(0.0, 0.0, 1.0);
    break;
}
```

4. 图形旋转及旋转方向变换

（1）控制旋转

通过改变旋转角度来控制图形是否进行视觉效果上的旋转。以下语句调整 angle 变量值为 0 或 10：

```
case 4://旋转开关
    if (angle == 0)
        angle = 10;
    else
        angle = 0;
    break;
```

每点击一次鼠标右键，就控制旋转一次。若 angle 为 0，则在视觉上图形不进行旋转；若 angle 为 10，则旋转 10°：

```
if (Transform)
{
    // 旋转
    glRotatef(angle, 0, 0, direction);

    Transform = !Transform;
}
```

（2）旋转方向变换

改变 direction 值为 1 或-1：

```
case 5:// 旋转方向
    if (direction == 1)
        direction = -1;
    else if (direction == -1)
        direction = 1;
    break;
```

旋转角度为 10 时，direction 控制旋转方向：

```
// 旋转
glRotatef(angle, 0, 0, direction);
```

5. 间隔扫描填充

通过如下语句，打开或关闭开关：

```
case 6:
    interval_flag = !interval_flag;
    break;
```

在对多边形填充之前，通过以下语句，判断是否进行间隔扫描。如果开关值为 1，则以 4 为一次间隔，进行扫描填色：

```
if (interval_flag)
{
    if (paint_interval % 8 == 1 || paint_interval % 8 == 2 || paint_interval % 8 == 3 || paint_interval % 8 == 4)
        glVertex2i(x, i);
}
else
    glVertex2i(x, i);
```

6. 读入文件

通过如下语句控制读文件开关：

```
case 7:
    readtxt = !readtxt;
    break;
```

在 void my_readtxt() 函数中，读入 myPolygon.txt 文件，文件中的第一行数字作为多边形的个数。

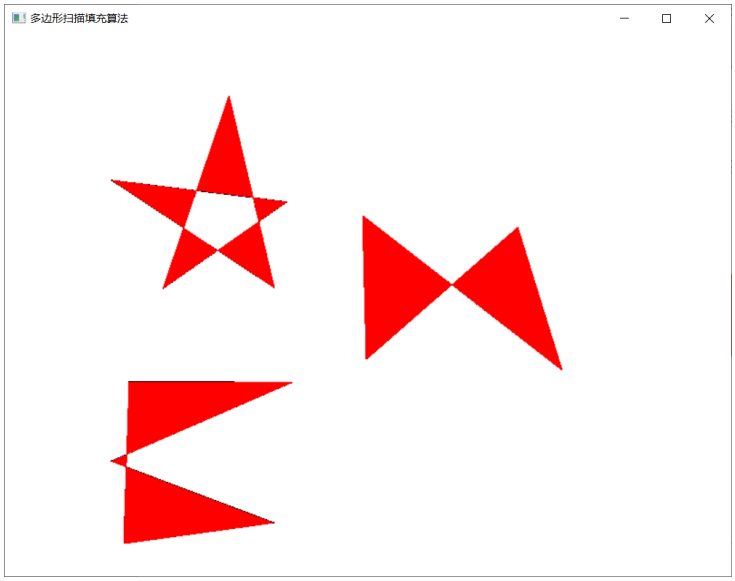
对于每一个多边形，第一行表示颜色，第二行表示边的条数，接下来几行表示每个点的 x 和 y 值。有了这些数据，就可以画出表示的多边形。

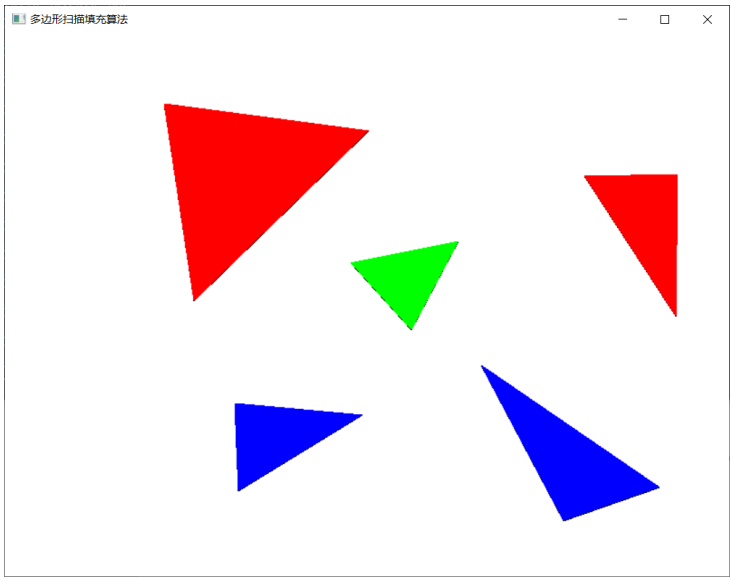
```
fscanf_s(fp, "%d", &(pcnt));
for (int j = 0; j < pcnt; j++)
{
    int lines;
    fscanf_s(fp, "%d", &(polygon_color));
    fscanf_s(fp, "%d", &(lines));
    for (int i = 0; i < lines; i++)
    {
        int xx, yy;
        fscanf_s(fp, "%d", &(xx));//输出数据到数组
        fscanf_s(fp, "%d", &(yy));
```

四、 实验结果

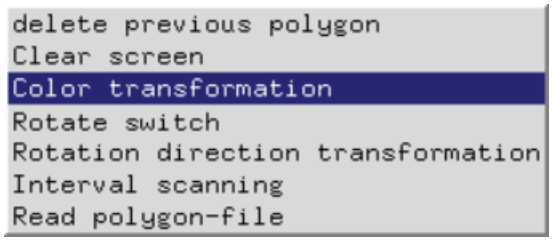
本程序可以通过鼠标中建调出菜单交互，也可以通过键盘按下不同数字进行交互。 接下来介绍项目实现的一些功能。

1. 通过橡皮筋交互输入不同颜色、大小的多边形，且要实现自相交多边形、多个多边形的扫描填充





其中，通过在菜单中点击“color transformation”，实现多边形颜色转换。

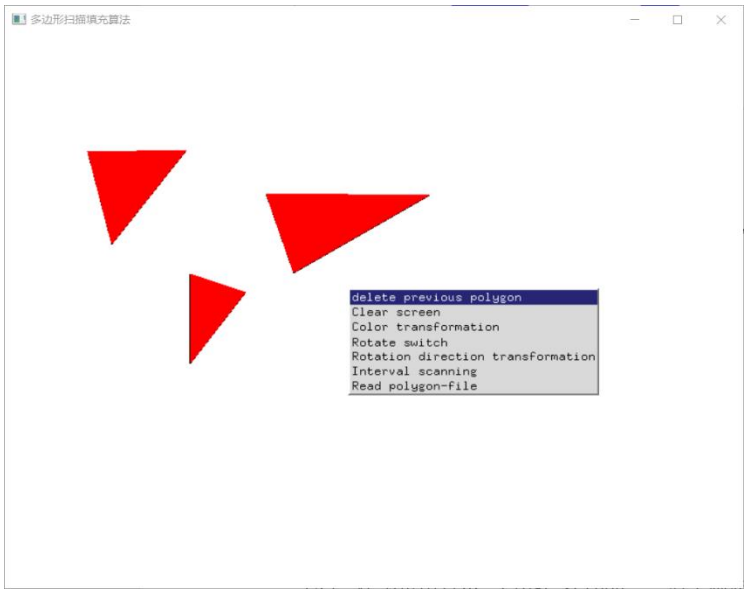


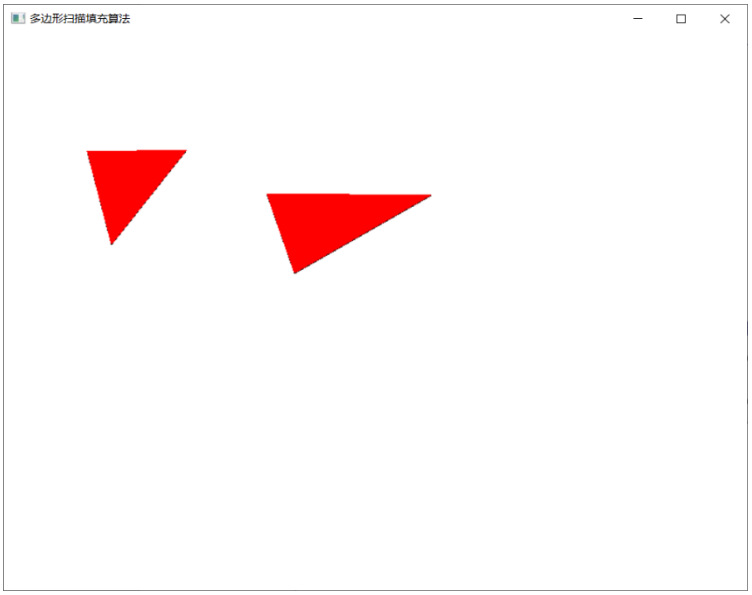
2. 清屏重置多边形，以及撤销之前画出的点线

（1） 在键盘上按下数字 1，撤销鼠标点击的上一个点

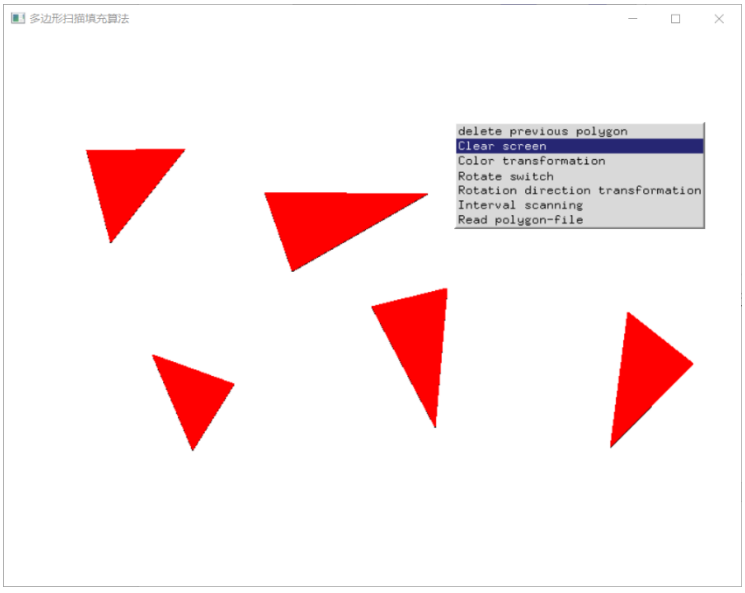
（2） 在键盘上按下数字 2，删除正在绘制的多边形的所有点和边

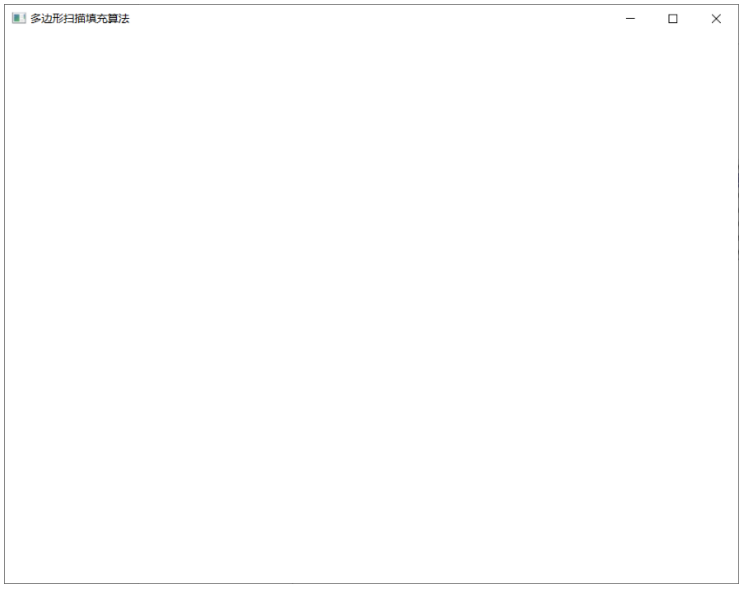
（3） 在菜单中点击“delete previous polygon”，删除上一个绘制好的多边形





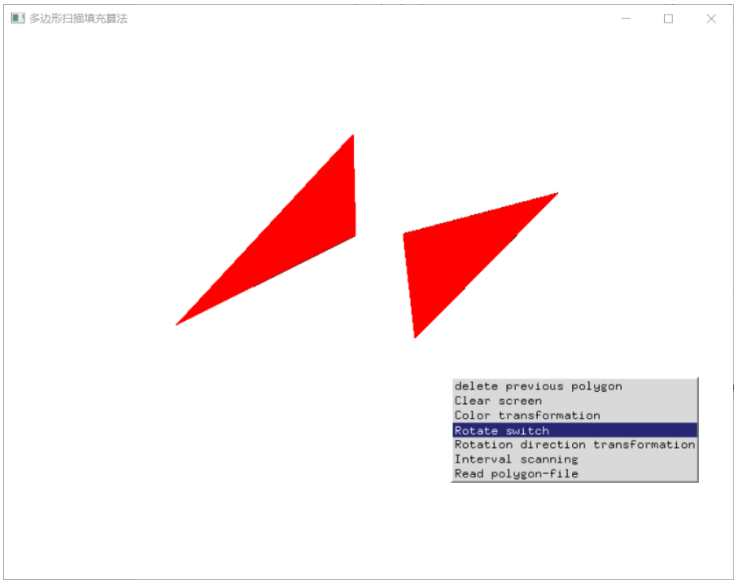
（4） 在菜单中点击“clear screen”，清空画布

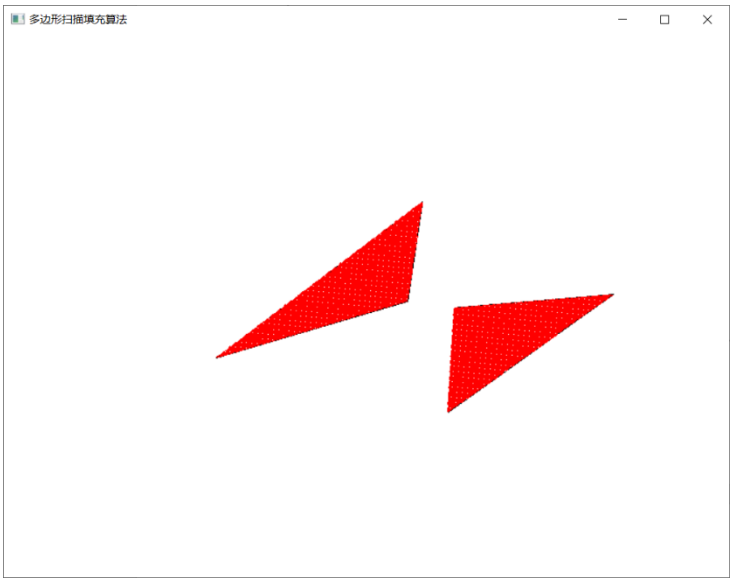


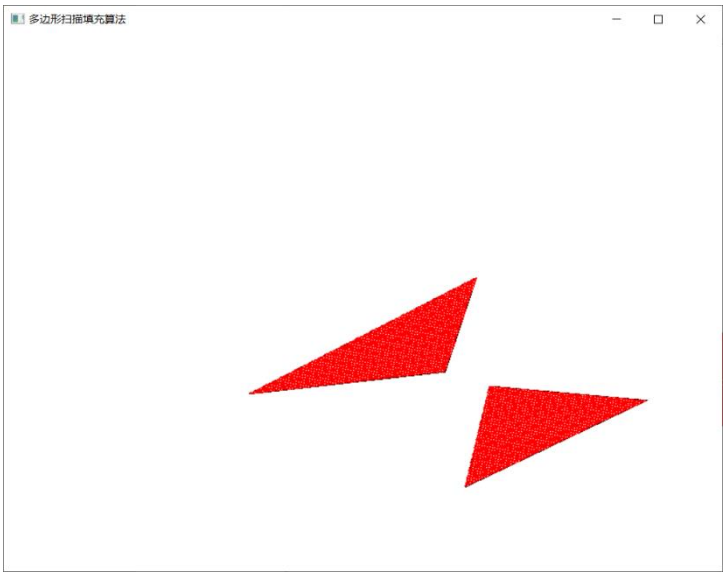


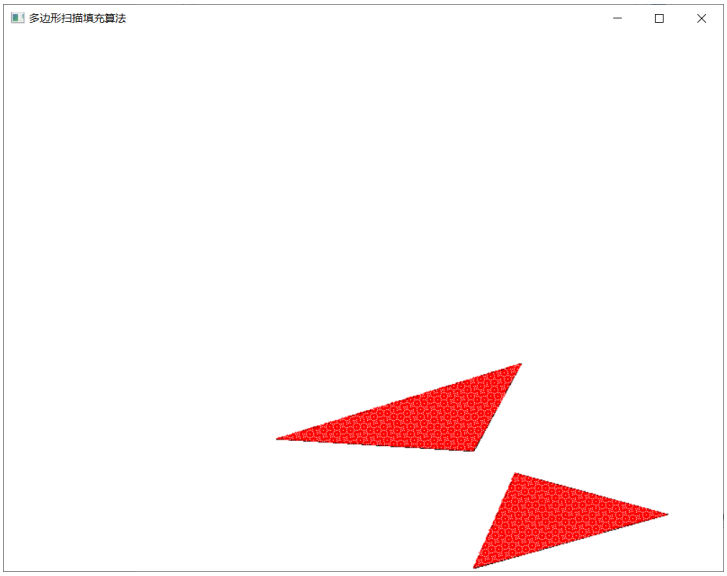
3. 实现图形旋转，并设置旋转方向

（1）在菜单中点击“rotate switch”，打开旋转开关

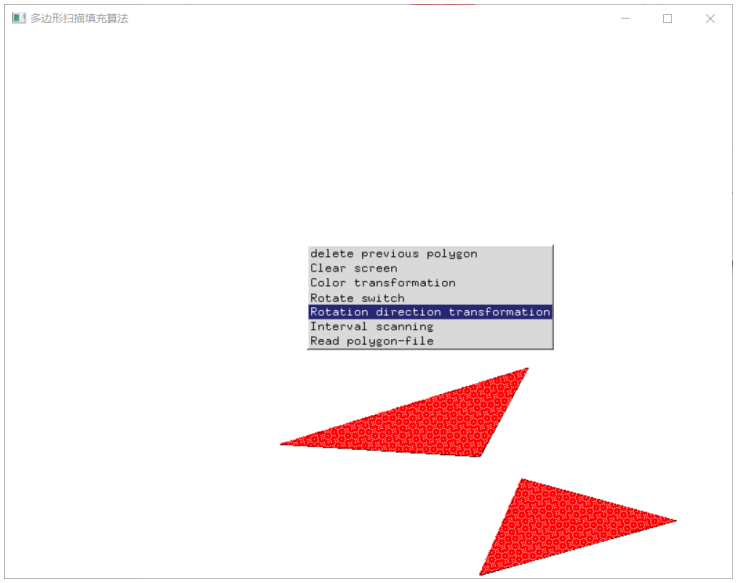


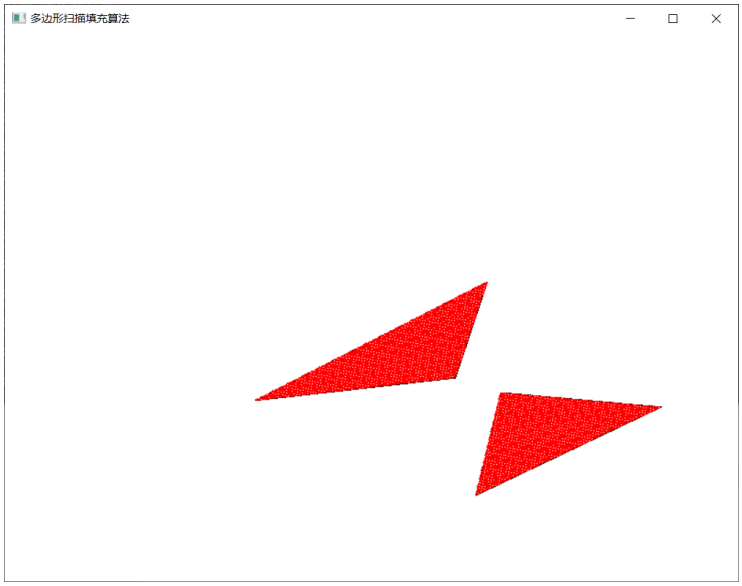






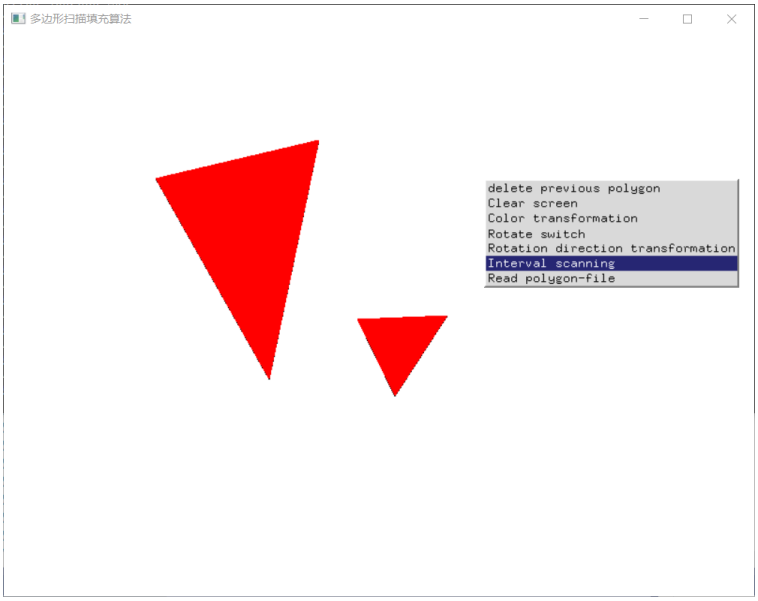
（2）在菜单中点击“rotation direction transformation”，改变旋转方向

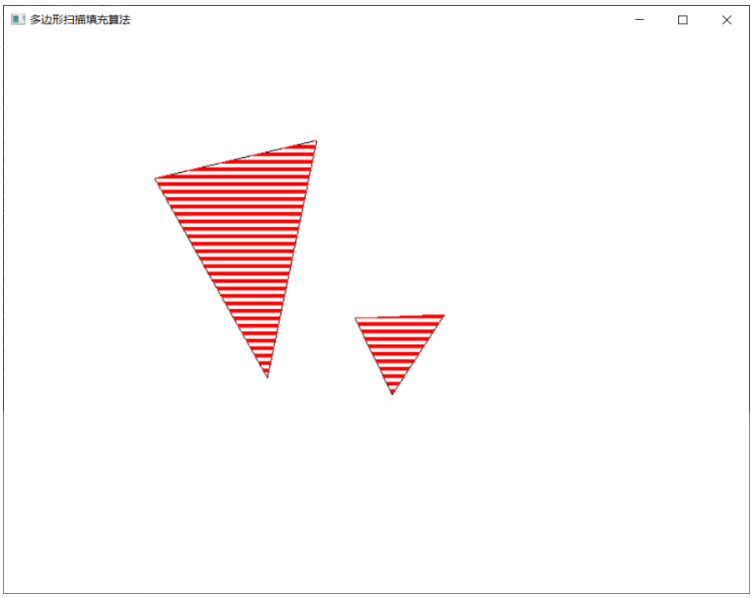




4. 可以选取不同的填充模式，如等间隔扫描线填充

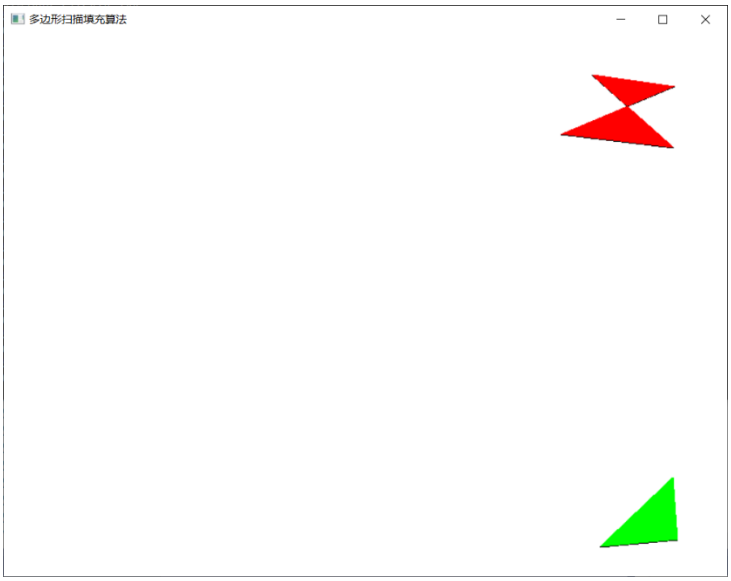
在菜单中点击“interval scanning”，实现扫描线填充。

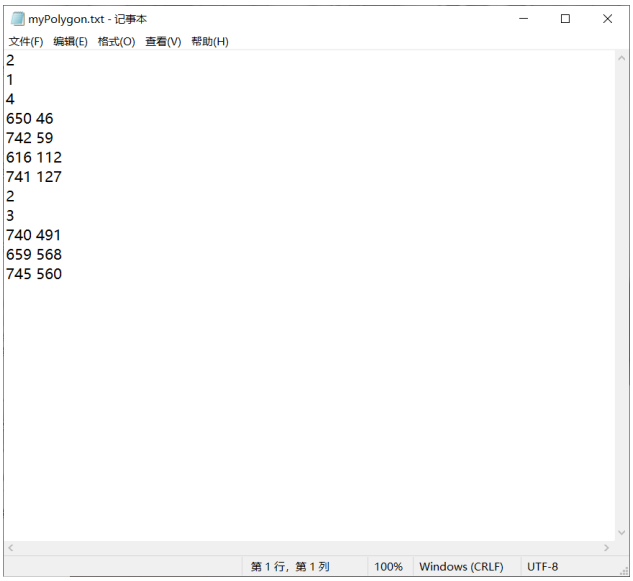




5. 通过文件读出需要交互输入的多边形

在菜单中点击“read polygon-file”，从文件中读入多边形。





经检验，可以看到，读入显示的结果正确。