一、课程设计目的

《计算机图形学课程设计》是实践性教学环节之一，是《计算机图形学》课程的辅助教学课程。通过课程设计，达到如下目的：

(1) 巩固和综合应用所学的计算机图形学理论知识，培养学生理论联系实际和独立思考问题的能力；

(2) 培养学生实际运用《计算机图形学》的知识设计计算机课题的思想和方法；

(3) 培养学生分析、设计和解决实际问题的能力，加强创新素质教育，激发学生的实际开发创造意识和能力；

(4) 培养学生调查研究、查阅技术文献、资料、手册以及编写技术文献的能力；

(5) 培养学生团结协作，共同完成相关课题的能力。

二、课程设计原理与内容

1.课程设计要求

要求学生在理解直线、圆、区域填充、图形裁剪等理论知识的基础上，借助于程序设计语言（如 VC++、VB 等）进行小型图形应用软件的开发，要求界面美观（包括菜单栏、工具栏）、功能相对完整、衔接自然、注重交互性。

应用 VC++以及 OpenGL 进行设计。

2.设计内容

主要知识点包括：

（1） 点、线、圆、多边形等计算机基本图元的的绘制
（2） 多边形的填充（提高题）
（3） 二维、缩放、旋转等计算机图形的几何变换 （必做）
（4） 图形的裁剪（提高题）
（5） 样条曲线的绘制（附加题）
（6） 图形的删除

3 算法原理

（1）DDA 算法

DDA 算法就是一个增量算法，即在一个迭代算法中，如果每一步的 x、y 值是用前一步的值加上一个增量来获得，则称为增量算法。

$$\begin{aligned} y_{i+1} &= kx_{i+1} + b \\ &= k(x_i + \Delta x) + b \\ &= kx_i + b + \Delta x \\ &= y_i + k\Delta x \end{aligned}$$

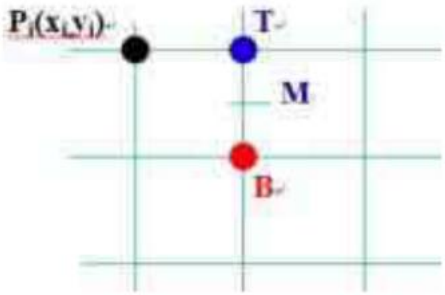
当 x 每递增 1，y 递增 k(即直线斜率)，即当$x_{i+1} = x_i + 1$时$y_{i+1} = y_i + k$

取整：$y_{i+1} = \text{round}(y_i + k + 0.5)$

（2）椭圆的中点画法

椭圆的中点画法确定一个象素后，接着在两个候选象素的中点计 算一个判别式的值，由判别式的符号确定更近的点。

先讨论椭圆弧的上半部分



设(x_p, y_p)已确定，则下一待选像素的中点是$(x_p + 1, y_p - 0.5)$,$d1 = F(x_p + 1, y_p - 0.5) = b^2(x_p + 1)^2 + a^2(y_p - 0.5)^2 - a^2b^2$

根据 d1 的符号来决定下一像素是取正右方的那个，还是右上方的那个。

若 d1＜0，中点在椭圆内，取正右方象素，判别式更新为：$d1' = F(x_p + 2, y_p - 0.5) = d1 + b^2(2x_p + 3)^2$,d1 的增量为$b^2(2x_p + 3)^2$

当 d1≥0，中点在椭圆外，取右下方象素，更新判别式：$d1' = F(x_p + 2, y_p -$

$1.5) = d1 + b^2(2x_p + 3)^2 + a^2(-2y_p + 2)^2$,d1 的增量为 $b^2(2x_p + 3)^2 + a^2(-2y_p + 2)^2$

（3）二维图形变换

二维图形变换主要是基于齐次坐标方程,通过一些简单的矩阵运算来实现:

二维齐次坐标变换的矩阵形式是: $\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$

矩阵的每个元素都有特殊含义。其中$\begin{pmatrix} a & b \\ d & e \end{pmatrix}$可以对图形进行缩放,旋转等变换$\begin{pmatrix} c \\ f \end{pmatrix}$;是对图形进行平移变换; (i)则是对图形整体进行缩放变换。

平移变换：

将一个图形在 X 方向中平移 tx 个单位,在 Y 方向平移 ty 个单位.其实现过程如下:

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

其中 x1,y1 是变换后的坐标,x,y 是变换前的坐标,通过上述变换,(x,y)被平移了 P(tx,ty). 在二维平面上任何复杂的变换都可以通过上述基本变换的组合来实现.在计算机上主要体现在矩阵的乘法运算,即将各个简单变换的矩阵逆序相乘,就可以得到一个总的变换矩阵.利用这个总的变换矩阵就可以对图形进行复合变换。

旋转变化：

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

放缩变换：

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

（4）Liang-Barsky 多边形裁剪算法：

算法的基本思想

以直线的参数方程为基础，对不同情况下的裁剪求得相应的参数值。

算法步骤

(1)输入直线段的两端点坐标以及窗口的四条边界坐标。

(2)若 Δx=0，则 p1=p2=0。进一步判断是否满足 q1<0 或 q2<0，若满足，则该直线段不在窗口内，转(7)。否则，满足 q1>0 且 q2>0，则进一步计算 u1 和 u2。转(5)。

(3)若 Δy=0，则 p3=p4=0。进一步判断是否满足 q3<0 或 q4<0，若满足，则该直线段不在窗口内，转(7)。否则，满足 q1>0 且 q2>0，则进一步计算 u1 和 u2。转(5)。

(4)若上述两条均不满足,则有 pk≠0(k=1,2,3,4)。此时计算 u1 和 u2。

(5)求得u1和u2后,进行判断:若u1>u2,则直线段在窗口外,转(7)。若u1<u2,利用直线的参数方程求得直线段在窗口内的两端点坐标。

(6)利用直线的扫描转换算法绘制在窗口内的直线段。

(7)算法结束。

（5）扫描线多边形区域填充算法

扫描线多边形区域填充算法是按扫描线顺序（由下到上），计算扫描线与多边形的相交区间，再用要求的颜色显示这些区间的象素，即完成填充工作。

区间的端点可以通过计算扫描线与多边形边界线的交点获得。对于一条扫描线，多边形的填充过程可以分为四个步骤：

（1）求交：计算扫描线与多边形各边的交点；

（2）排序：把所有交点按 x 值递增顺序排序；

（3）配对：第一个与第二个，第三个与第四个等等；每对交点代表扫描线与多边形的一个相交区间;

（4）填色：把相交区间内的象素置成多边形颜色;

（6）Bezier 曲线算法

贝塞尔曲线原理：贝塞尔曲线是计算机图形图像造型的基本工具，是图形造型运用得最多的基本线条之一。它通过控制曲线上的四个点（起始点、终止点以及两个相互分离的中间点）来创造、编辑图形。其中起重要作用的是位于曲线中央的控制线。这条线是虚拟的，中间与贝塞尔曲线交叉，两端是控制端点。移动两端的端点时贝塞尔曲线改变曲线的曲率（弯曲的程度）；移动中间点（也就是移动虚拟的控制线）时，贝塞尔曲线在起始点和终止点锁定的情况下做均匀移动。

三、流程图和主要源程序

1 程序设计方案

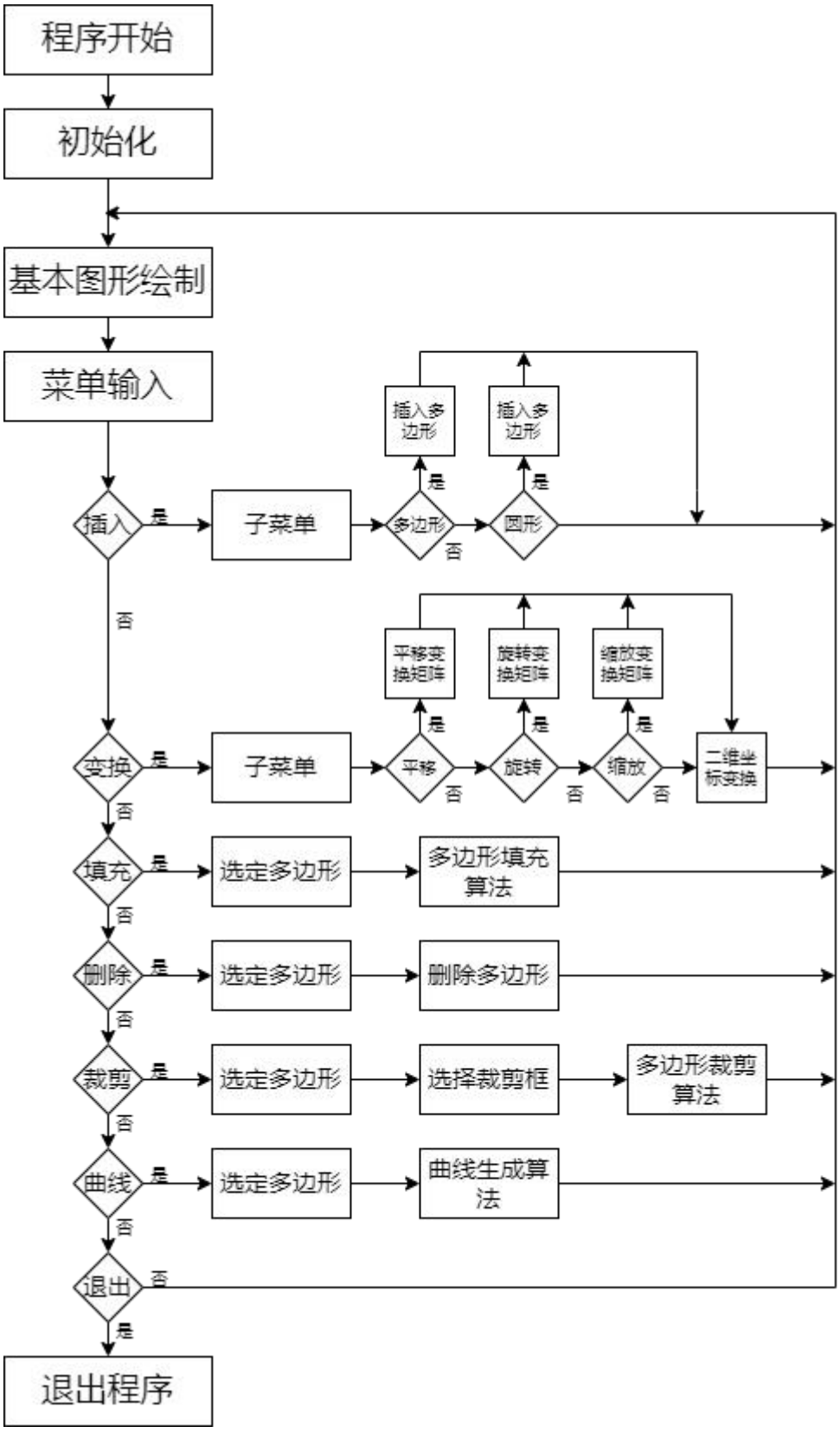
通过使用 OpenGL 提供的标准库函数进行窗口设置以及图形绘制界面，通过 glutMouseFunc，addmenu，glutMouseFunc，glutMotionFunc 等库实现交互。

综合图形学 DDA 画线和画圆的算法，以及其他图形绘制算法，用 glColor3f 函数调节绘制颜色以实现更好的交互。利用 windows.h 库中的 MessageBox 函数提示用户，在使用每一个功能时候的操作步骤，并在用户操作失误的时候也会做出相应的判断以及调整，并提醒用户。

该程序主要通过构建多边形类和椭圆类进行数据存储，通过输入多边形的顶点实现数据输入，二维变换主要是通过顶点乘以相应的变换矩阵进行变换；多边形填充算法和曲线绘制主要是通过用户指定的多边形，改变其相应的数值，以此在每次输出的时候都判断是否需要做更多的输出比如填充以及绘制曲线等。多边形裁剪是其中最复杂的功能，需要用户设置裁剪窗口然后再选定顶点才能进行裁剪。

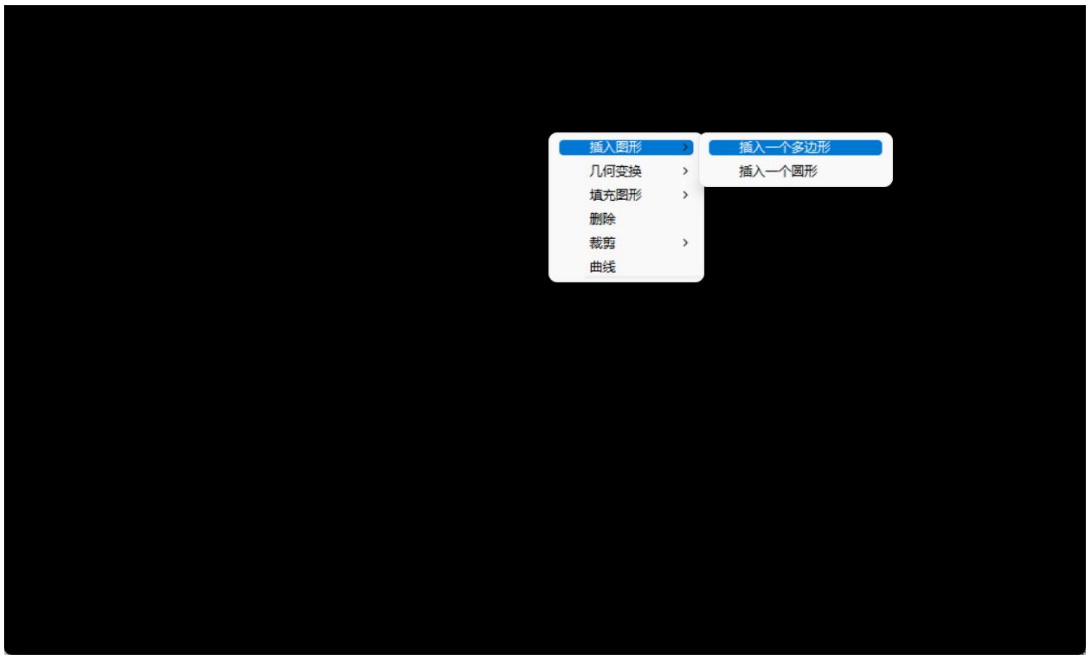
总的来说，该程序设计无论是在任务完成和交互方面功能都较为完善，具体流程图如下图所示。

2 流程图

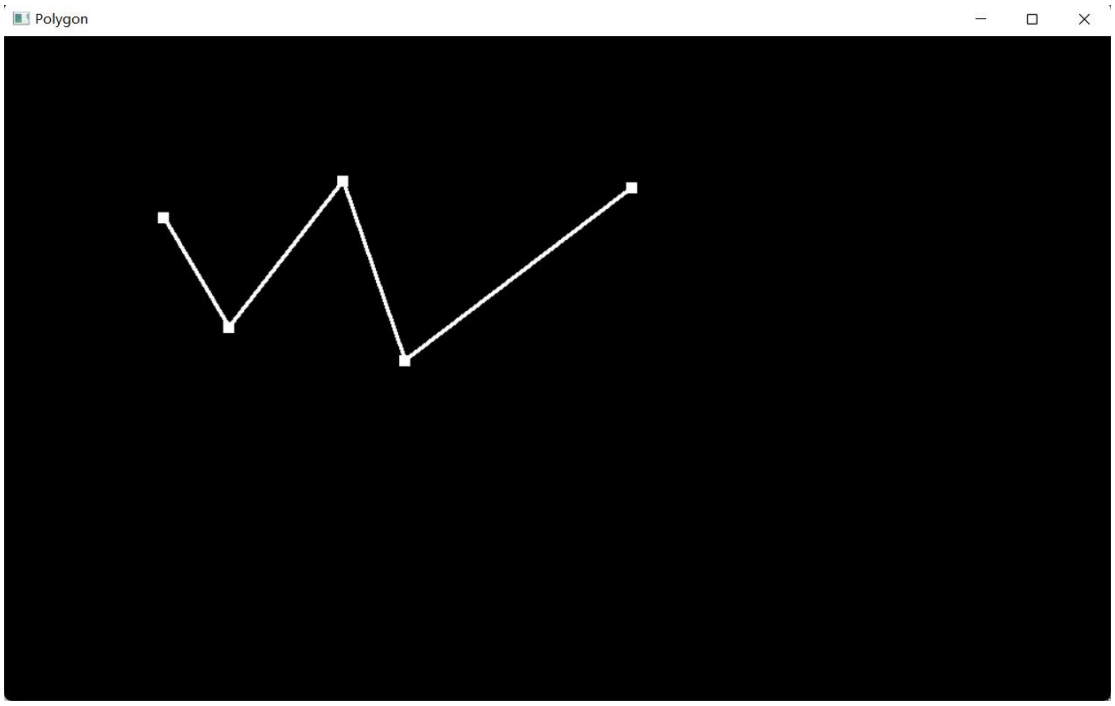

程序开始
初始化
基本图形绘制
菜单输入
插入
是
子菜单
多边形
是
插入多边形
否
圆形
是
插入多边形
否
变换
是
子菜单
平移
是
平移变换矩阵
否
旋转
是
旋转变换矩阵
否
缩放
是
缩放变换矩阵
否
二维坐标变换
否
填充
是
选定多边形
多边形填充算法
否
删除
是
选定多边形
删除多边形
否
裁剪
是
选定多边形
选择裁剪框
多边形裁剪算法
否
曲线
是
选定多边形
曲线生成算法
否
退出
否
是
退出程序

3 功能介绍

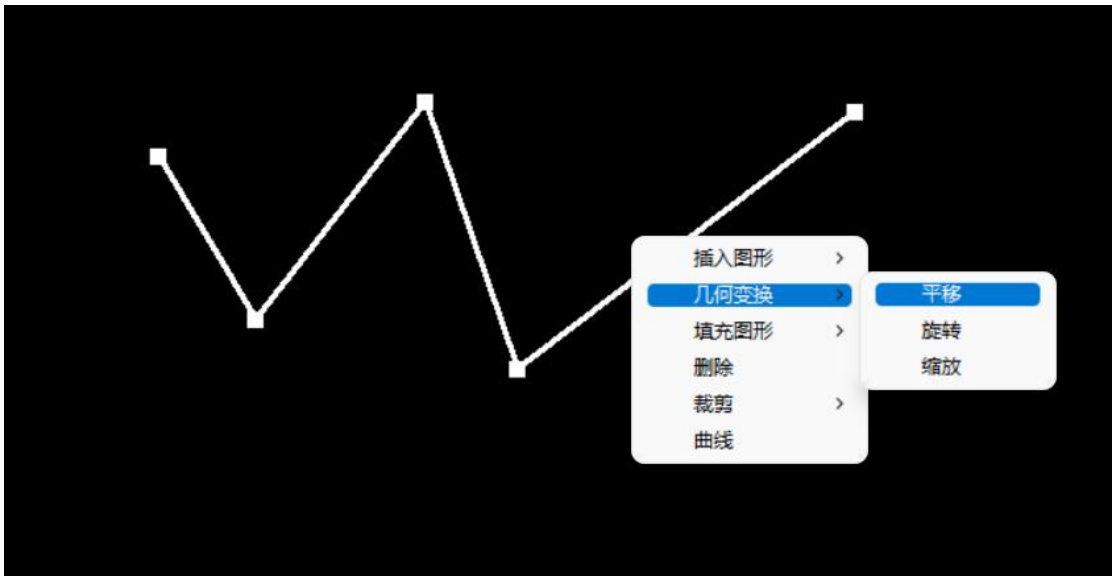
插入多边形



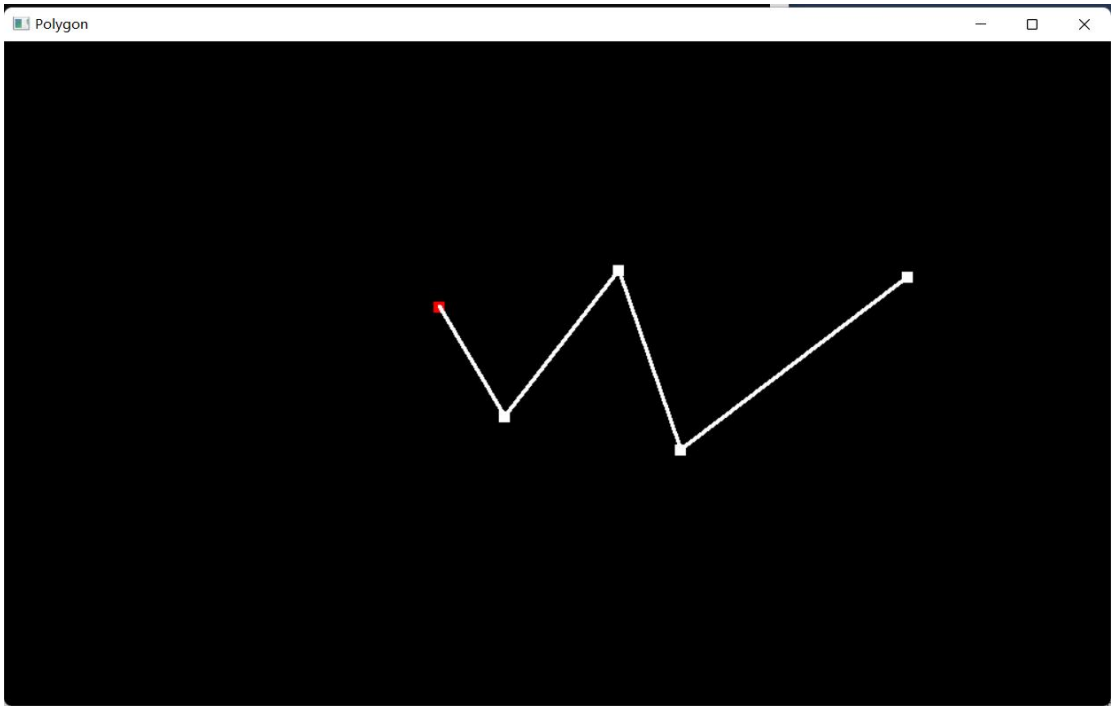
绘制出一段曲线



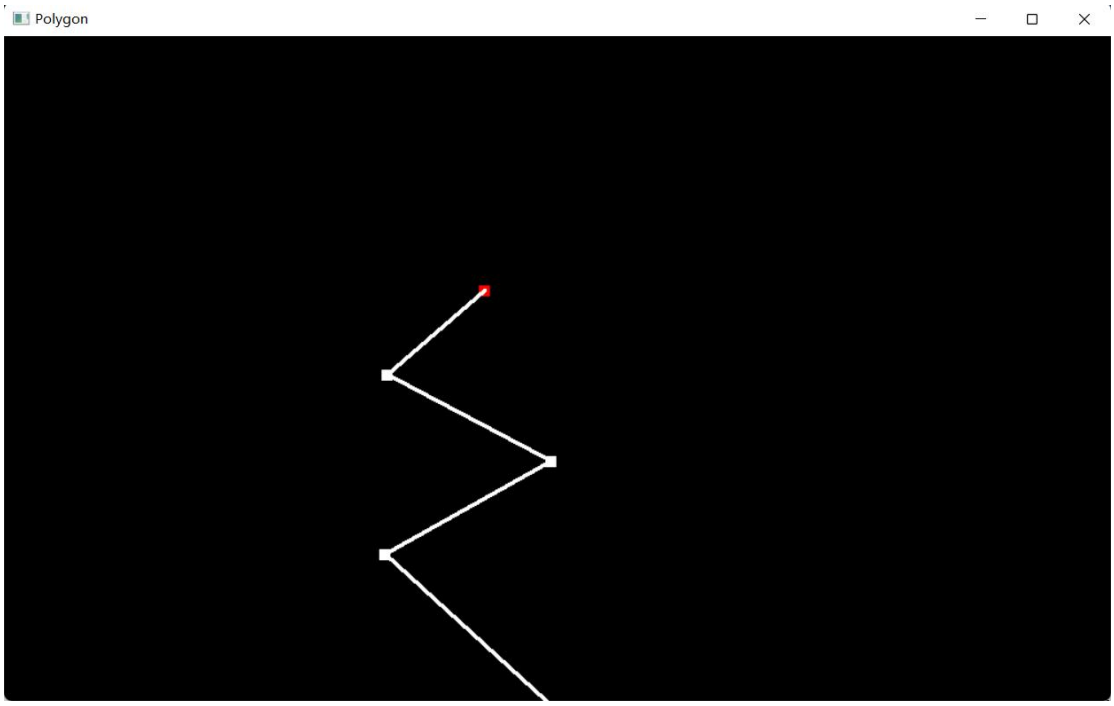
平移操作



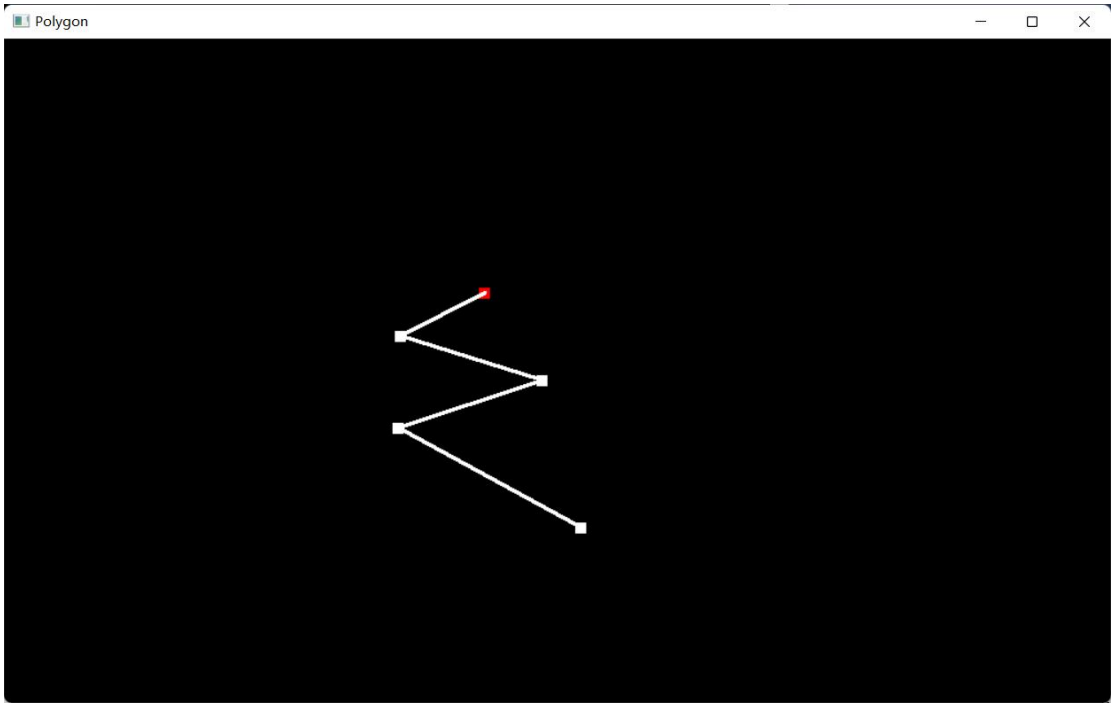
选定，并拖动



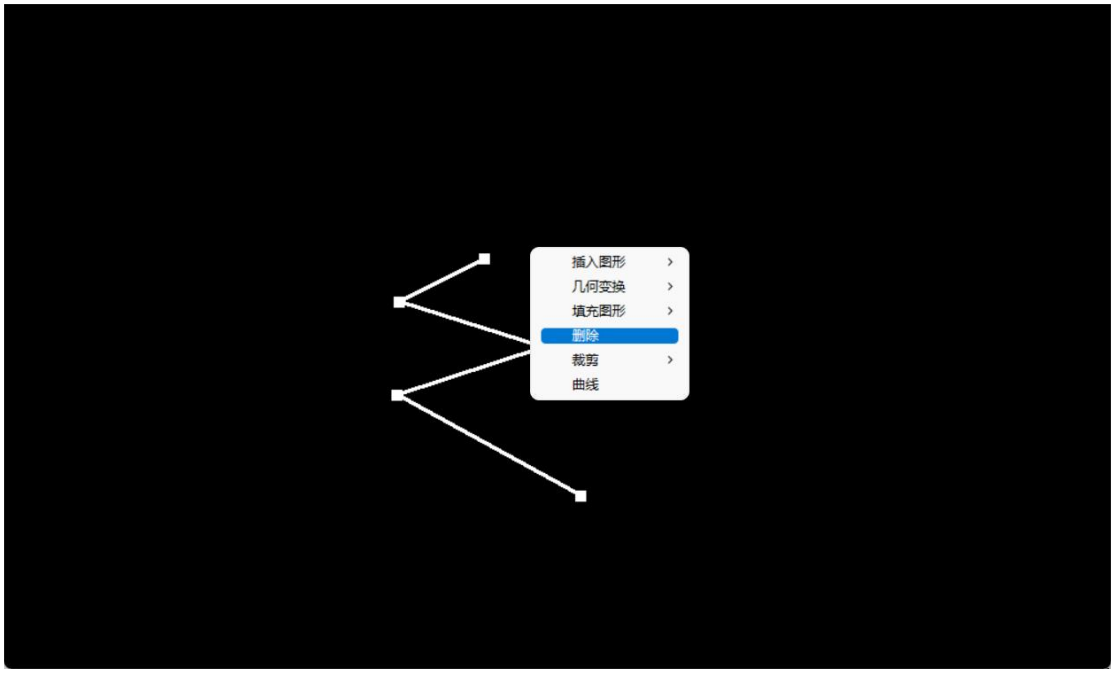
旋转：选定并拖动：

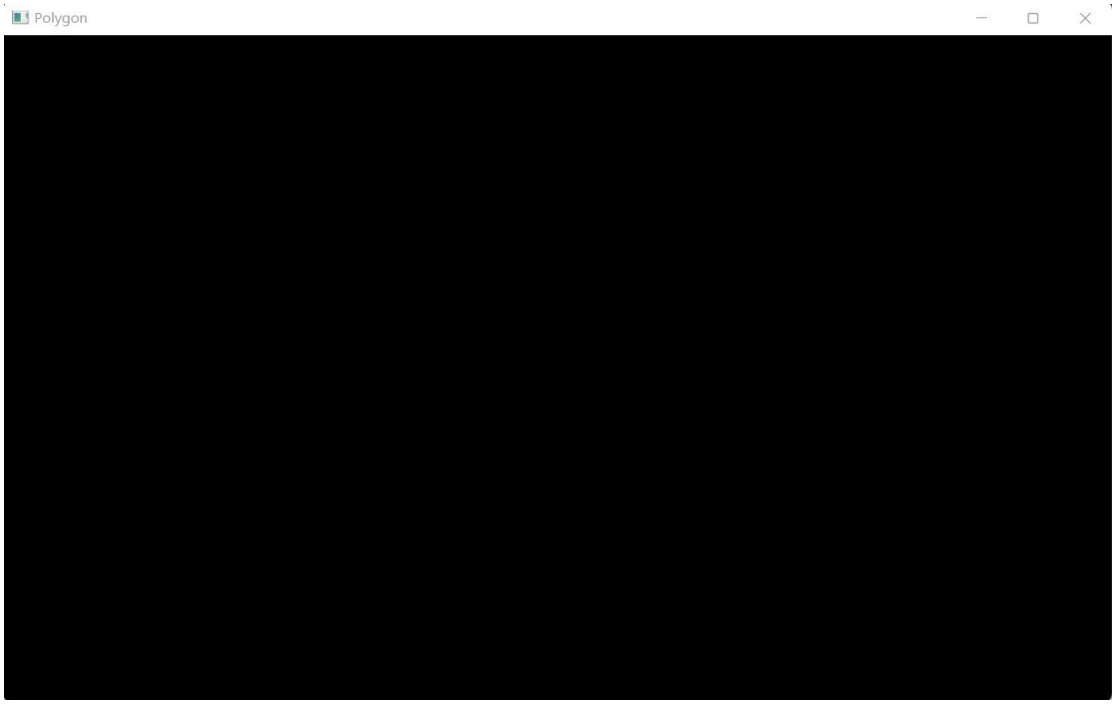


缩放：选定并拖动

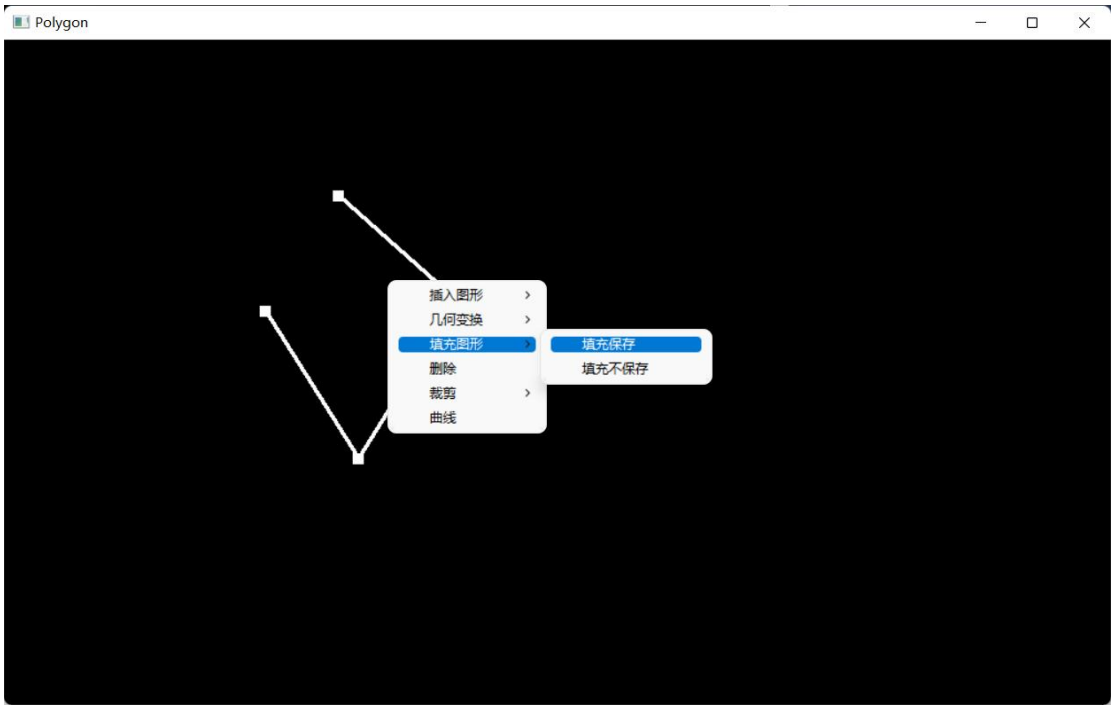


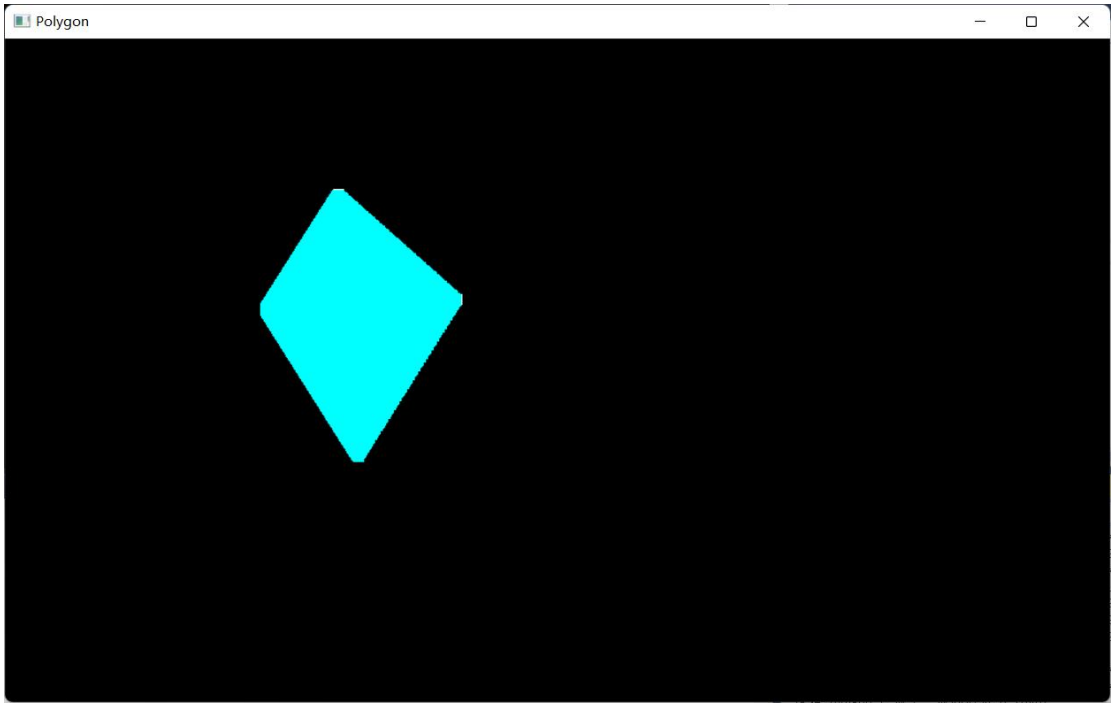
删除图形：



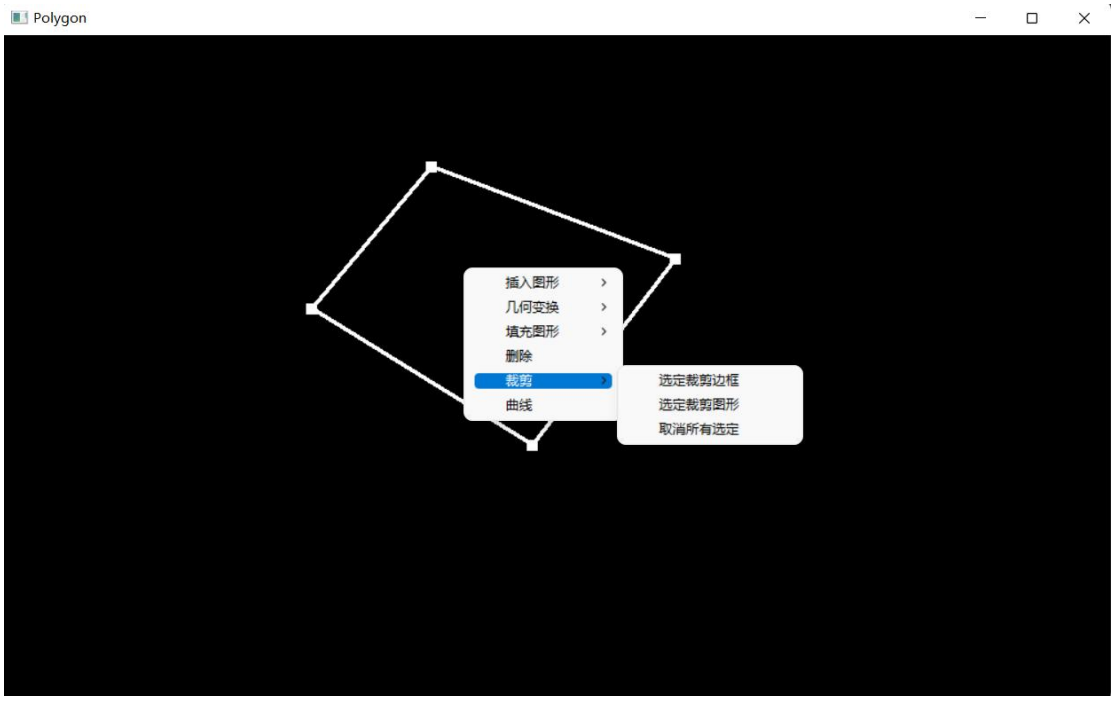


再绘制一个，并进行多边形填充：

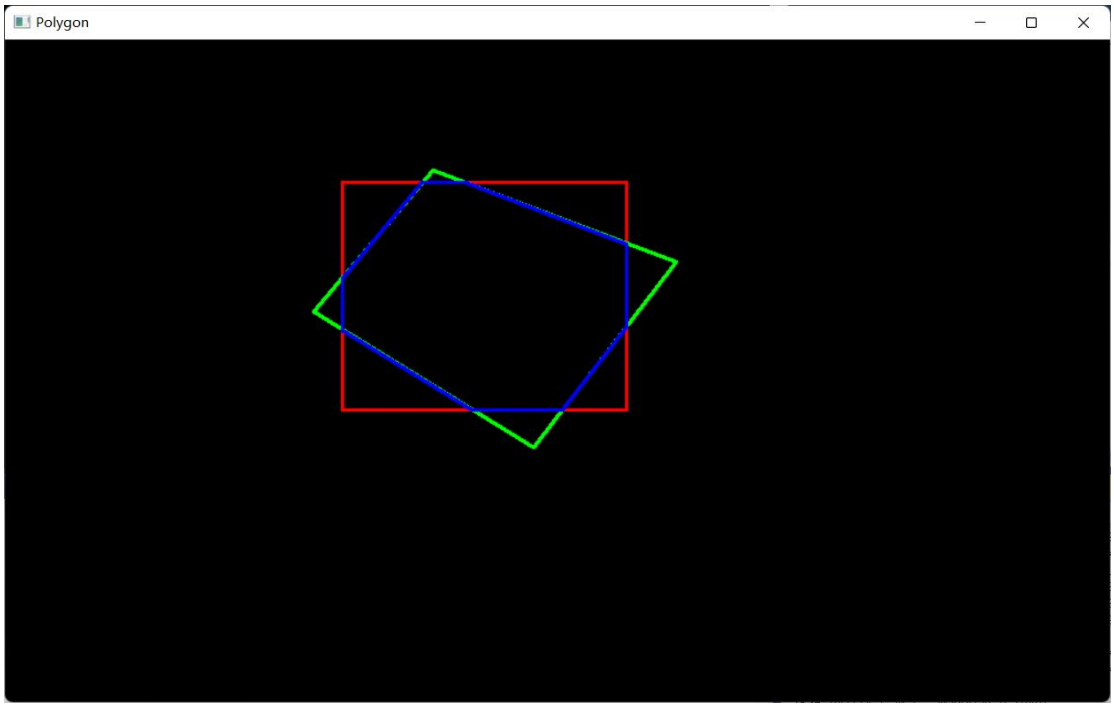




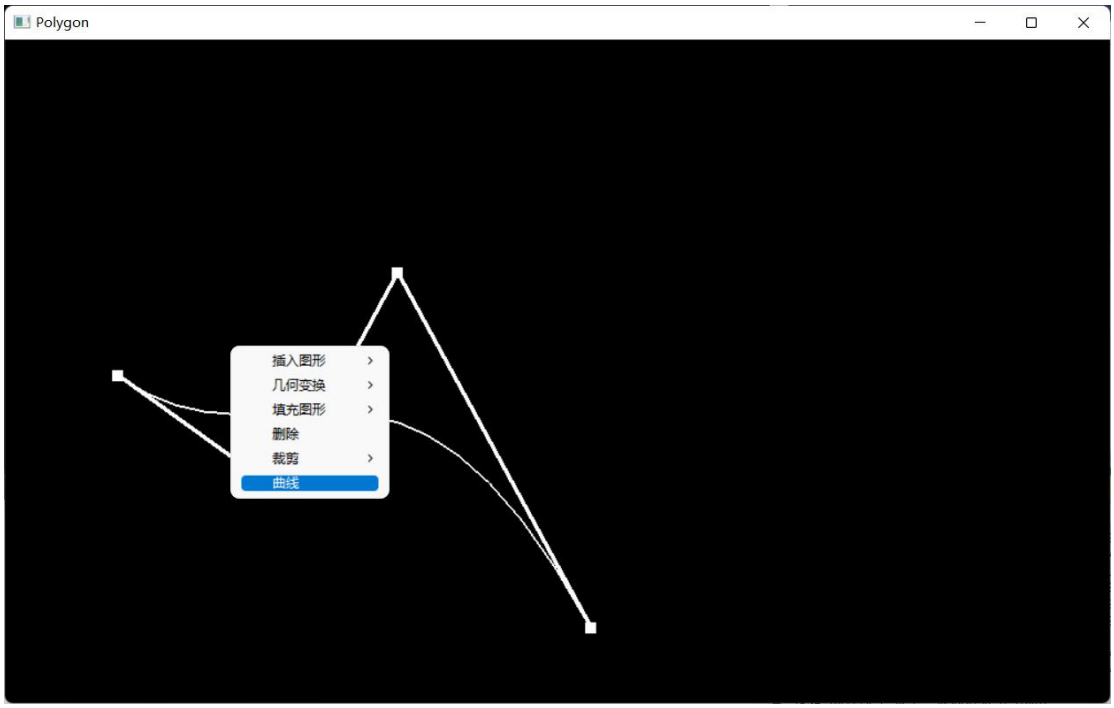
删除，并选择多边形裁剪：



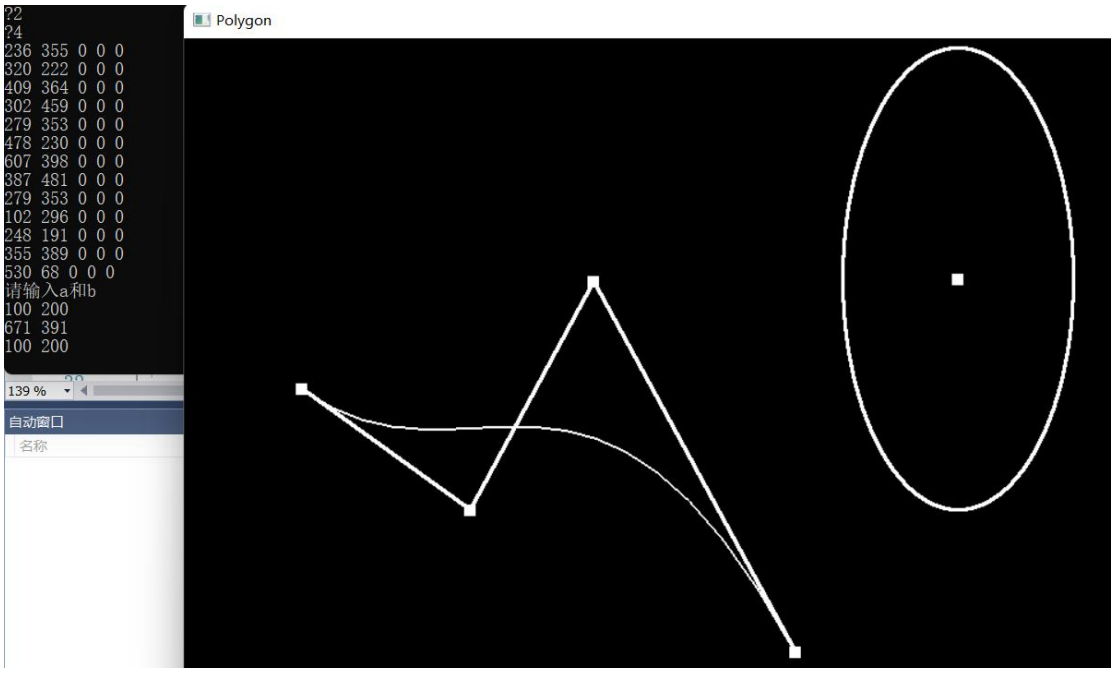
选定两个顶点后选择多边形：



取消选定，绘制曲线：



绘制椭圆：选定中点后输入 a 和 b



四、问题讨论

1 系统不足

在本次课程设计中，也是很不人性化的一点，就是没用用户自定义的颜色改变的实现方法，只有一种颜色改变的方案，没有设置对话框来设置多种颜色供选择的，所以改编后的颜色比较单一，此为本次设计方案的不足之一：然后就是算法主要是针对多边形的，对于椭圆形的操作是在有限，此为设计方案不足之二。因为实验重要的只是思想过程，效果的不美观是因为学习的可视化操作实现并不足，所以不能设计出真正很人性化的程序。

2 改进方案

通过修改我的程序中的右键更改颜色的函数，更改成生成效果是用对话框提示替换颜色的函数就可以达到可以选择改变的颜色了，查找程序中的此函数进行修改，可以达到选择颜色修改的效果。至于椭圆的操作在每一次循环遍历多边形的动态数组的时候再另外加一个循环遍历椭圆中心的动态数组，并判断有关操作即可。

3 心得体会

在本次课程设计过程中，基本掌握了 OpenGL 提供的标准库函数，Bresenham 画线和画椭圆的方法，了解的很多有关于图形学的知识。不过只是皮毛而已，从中锻炼了自己的动手做实验的能力，但同时也让自己看清了自己的水平，以便在以后的生活里多加强有关这方面的学习，从而提升自己在图形学方面的知识水平。