

**面向对象程序设计**

**大 程 序 报 告**



大程名称： 基于OPENGL的geometry tool设计

**姓名 ： 沈祺昊**

**学号： 3200104734**

**电话：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 17816081795**

**指导老师： 李际军**

**2021~2022春夏学期 2022 年 6 月 10 日**

**报告撰写注意事项**

1. 图文并茂。文字通顺，语言流畅，无错别字。
2. 书写格式规范，排版良好，内容完整。
3. 存在拼凑、剽窃等现象一律认定为抄袭；0分
4. 蓝色文字为说明，在最后提交的终稿版本，请删除这些文字。

**目 录**

[1 大程序简介 4](#_Toc71541784)

[1.1 选题背景及意义 4](#_Toc71541785)

[1.2 目标要求 4](#_Toc71541786)

[1.3 术语说明 4](#_Toc71541787)

[2 需求分析 4](#_Toc71541788)

[2.1 业务需求 4](#_Toc71541789)

[2.2 功能需求 5](#_Toc71541790)

[2.3 数据需求 5](#_Toc71541791)

[2.4 性能需求 5](#_Toc71541792)

[3 类库已有功能分析 6](#_Toc71541793)

[3.1 总体架构设计 6](#_Toc71541794)

[3.2 类体系设计 6](#_Toc71541795)

[3.3 主要类设计 7](#_Toc71541796)

[3.4 源代码文件组织设计 7](#_Toc71541797)

[3.5 关键功能类及函数设计描述 8](#_Toc71541798)

[4 新设计类功能说明 9](#_Toc71541799)

[4.1 总体架构设计 9](#_Toc71541800)

[4.2 类模块体系设计 10](#_Toc71541801)

[4.3 数据结构类设计 10](#_Toc71541802)

[4.4 源代码文件组织设计 10](#_Toc71541803)

[4.5 重点类及函数设计描述 11](#_Toc71541804)

[5 部署运行和使用说明 14](#_Toc71541805)

[5.1 编译安装 14](#_Toc71541806)

[5.2 运行测试 16](#_Toc71541807)

[5.3 使用操作 **错误！未定义书签。**](#_Toc71541808)

[5.4 收获感言 21](#_Toc71541809)

[6 参考文献资料 22](#_Toc71541810)

Geometry tool大程序设计

# 大程序简介

## 选题背景及意义

OPENGL图形库作为C++重要的图形库之一，具有极大的应用价值。对C++初学者来说，它对于初学者理解对象的封装、继承有着较强的推动作用。通过用OPENGL实现一个类似geometry tool的几何绘图软件，在加强学生学习应用C++图形界面的同时，也加深学生对于C++类与对象的理解。

## 目标要求

用C++标准库以及OPENGL库实现一个几何图形绘制小程序，要求不仅仅是绘图，更要求其实用性，同时体现出C++面向对象的特点。小程序中要包含简单的菜单栏以及画布、坐标系，同时实现简单几何图形的绘制，并实现鼠标和键盘相应功能。

## 术语说明

类（class）和对象(object)是两种以计算机为载体的计算机语言的合称。对象是对客观事物的抽象，类是对对象的抽象。类是一种抽象的数据类型。

它们的关系是，对象是类的实例，类是对象的模板。

# 需求分析

## 业务需求

本大程序实现了一个带简易菜单栏的简易几何图形绘制程序。在程序中需要体现出C++的特点，同时实现鼠标、键盘等相应函数。

## 功能需求

## 数据需求

1、point类：绘图程序中最基本的类，有两个数据成员，x与y，用于表示点在画布上的坐标。

2、Line类：用于表示线段，使用公有继承point类，用来表示线段两端的点。

3、tangle类：用于表示三角形，使用公有继承point类，表示三角形的顶点。

4、circle类：用于表示原型，使用公有继承point类，用来表示圆心。

5、draw类：将绘制线段、圆等函数封装到一个类当中去。

6、text类：由于OPENGL库没有输出文字的功能，因此额外实现一个可以输出文字的类。

## 性能需求

性能上，要求大程序能够完成几何图形的绘制，画布的缩放操作，菜单栏功能切换等功能。同时应实现鼠标拖动等简单鼠标响应函数的实现。

# 类库已有功能分析

## 总体架构设计

## 类体系设计

## 主要类设计

## 源代码文件组织设计

<文件目录结构>

1）文件函数结构

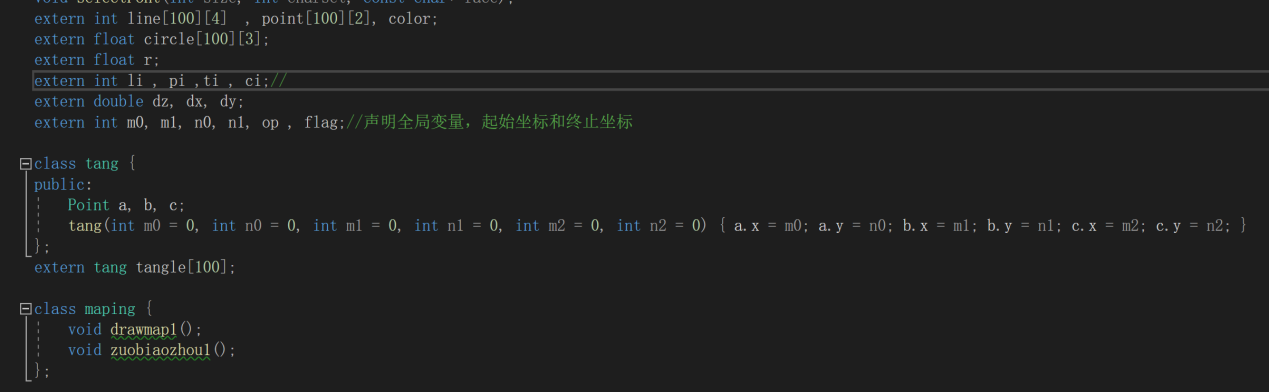


图一

大程序共有一个头文件，三个源文件。其中mouse event.cpp中是鼠标、键盘响应函数，drawmap.h中实现了绘图类函数的封装。

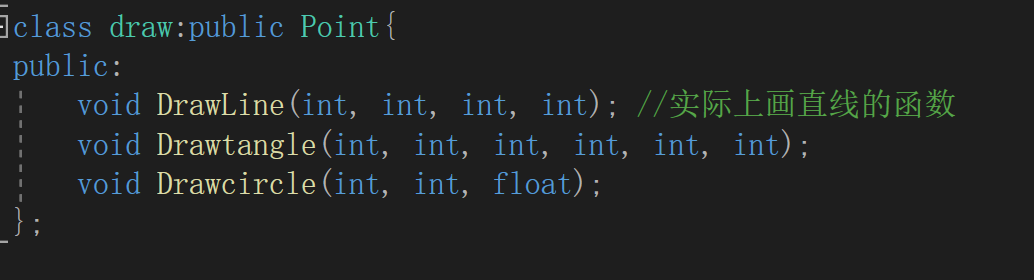
2）多文件构成机制

头文件drawmap.h中实现了类的封装，以及用external外部变量的定义与保护，实现了多个文件共用变量的功能。



图二

## 关键功能类及函数设计描述



绘图函数类



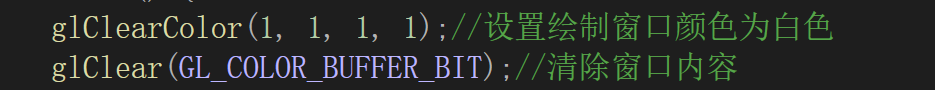
函数原型：mainMenu(int id);

功能描述；用于菜单栏工具的选择

参数描述： id表示选择工具的序号

返回值描述：void

函数算法描述：利用分支结构，实现工具的选择

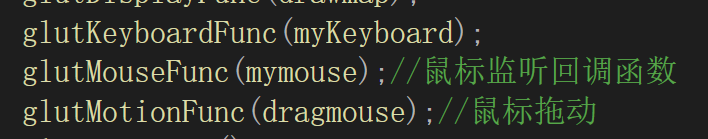


函数名称：glClearColor(float,float,float,float);

功能描述；用于设置画布颜色，与glClear配合使用可以将画布清除

参数描述： 前3个float分别是RGB颜色值，第四个是不透明度

返回值描述：void



函数描述：OPENGL库自带的响应函数，大程序中共用到了上图三种：

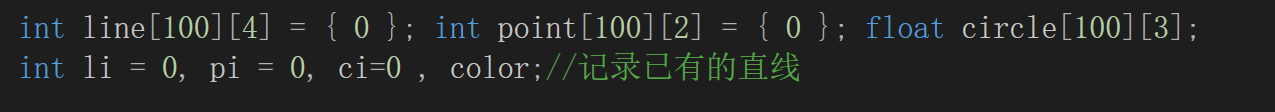
键盘响应、鼠标点击响应、鼠标拖动响应。

# 新设计类功能说明

## 总体架构设计

## 类模块体系设计

## 数据结构类设计



利用数组，记录已经绘制的几何图形

作用：在实现鼠标拖动的函数中，实现的机制是不断清除拖动前的线段痕迹，如果直接使用屏幕清除函数，会导致先前已经绘制的函数被清除，因此需要先将它们记录下来，这样在鼠标拖动时就可以保证之前已存在的图形不会被清除。

## 源代码文件组织设计

<文件目录结构>

1）文件函数结构



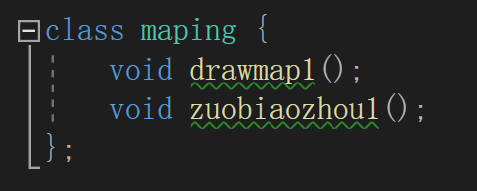
图一

大程序共有一个头文件，三个源文件。其中mouse event.cpp中是鼠标、键盘响应函数，drawmap.h中实现了绘图类函数的封装。

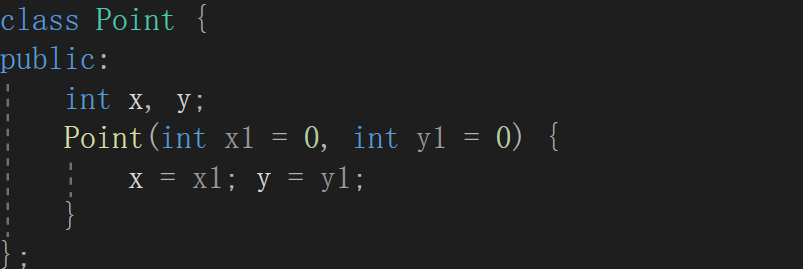
2）多文件构成机制

头文件drawmap.h中实现了类的封装，以及用external外部变量的定义与保护，实现了多个文件共用变量的功能。

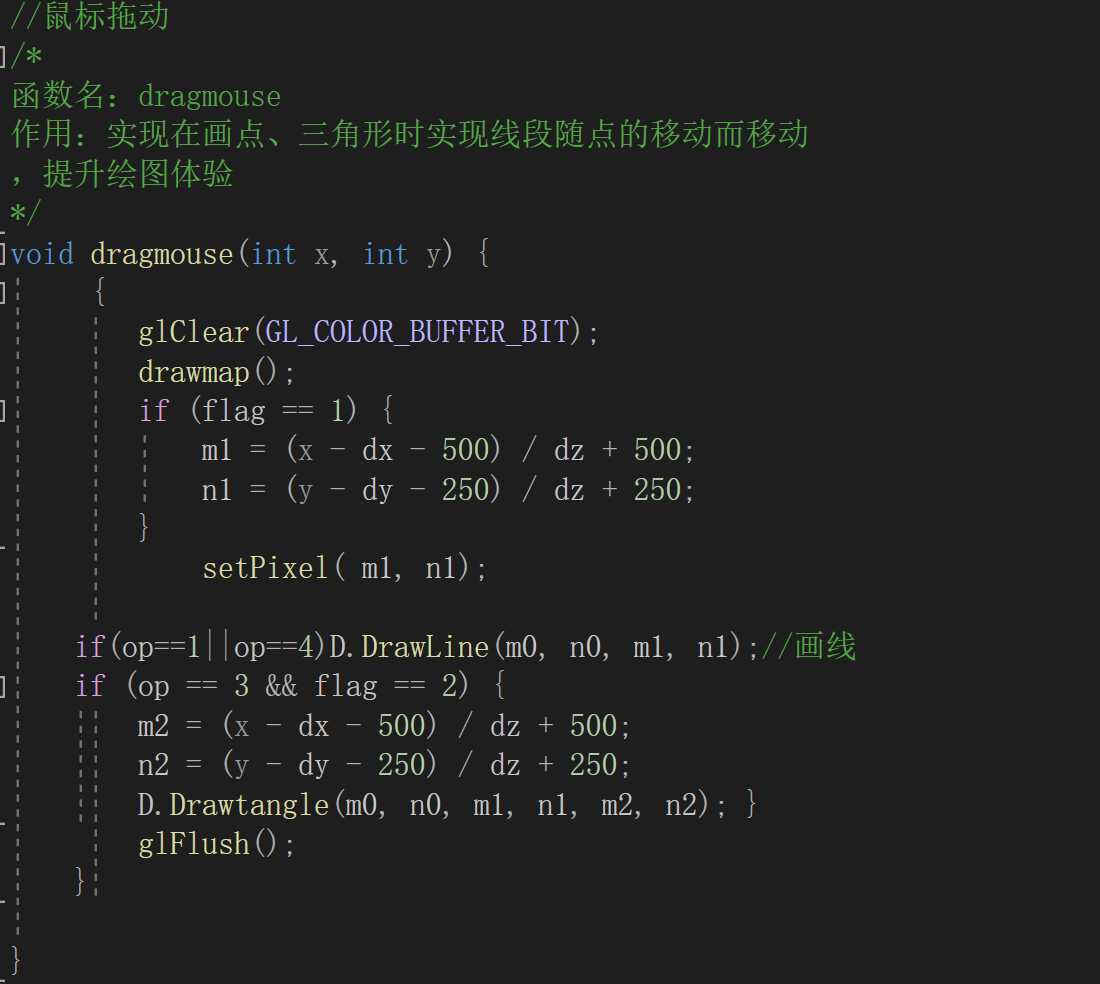
## 重点类及函数设计描述



坐标图类：用于绘制屏幕中最初的坐标系。



点类：用于记录绘图时鼠标点击的位置，是绘制所有几何图形的基础

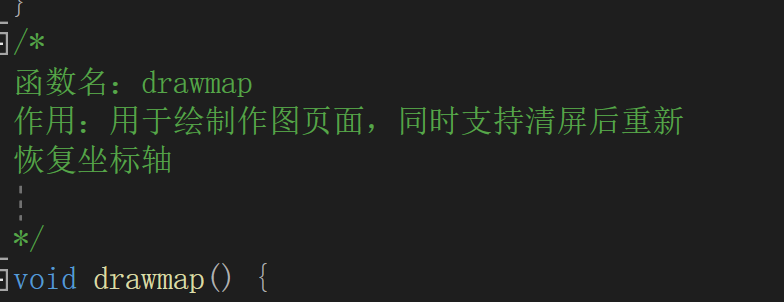


函数原型：dragmouse(int x,int y);

功能描述；用于绘图时鼠标拖动

参数描述： x,y分别表示鼠标的横纵坐标位置

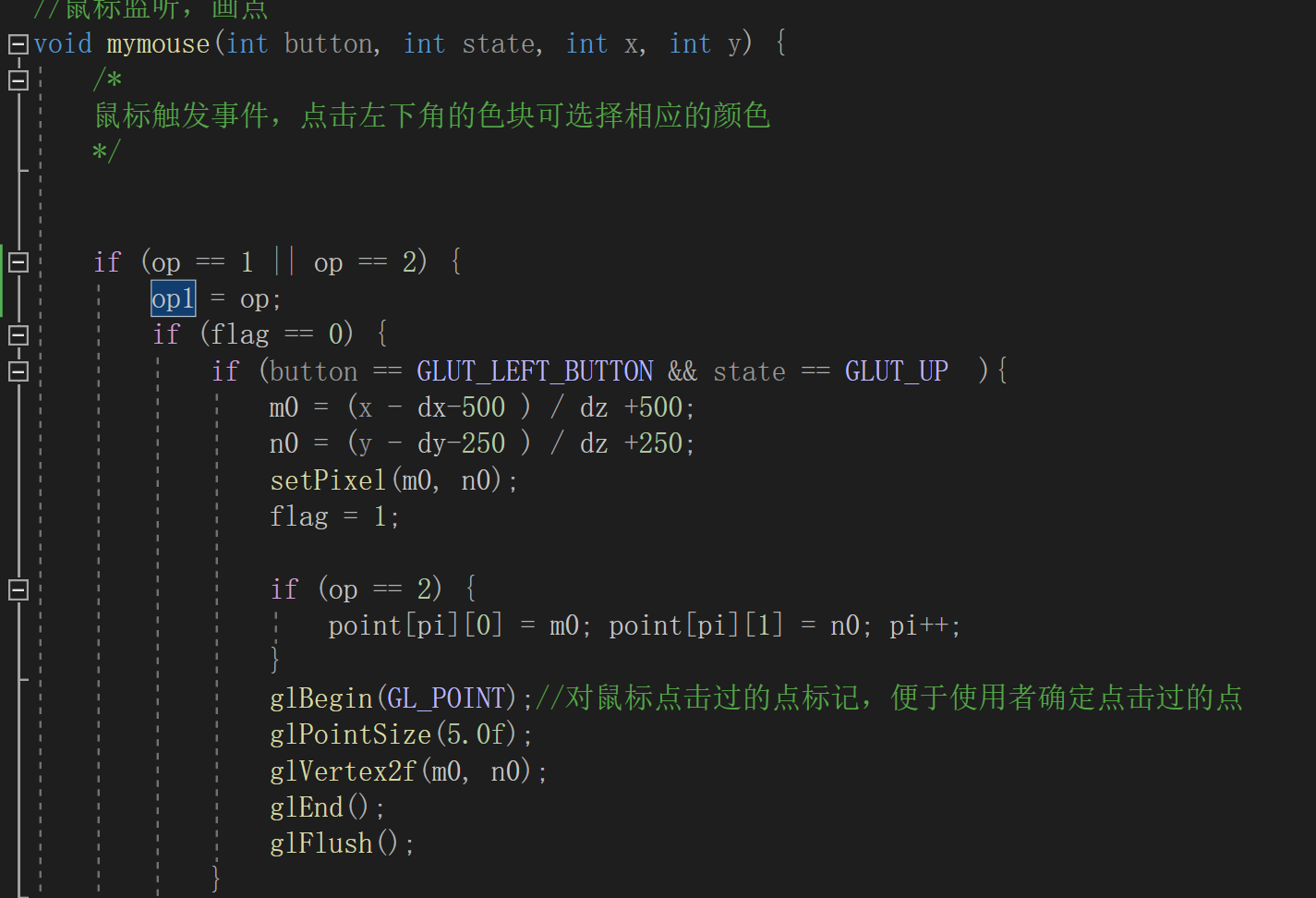
返回值描述：void



函数原型：drawmap();

功能描述；用于绘制坐标系，在最初以及刷新屏幕时会用到

返回值描述：void



函数名称：mymouse(int button, int state, int x,int y);

功能描述；用于绘制几何图形，op的值与菜单栏选择的绘制图形有关

Op=1:绘制线段

Op=2:绘制点

Op=3:绘制三角形

Op=4:绘制圆

Op=5:清除屏幕

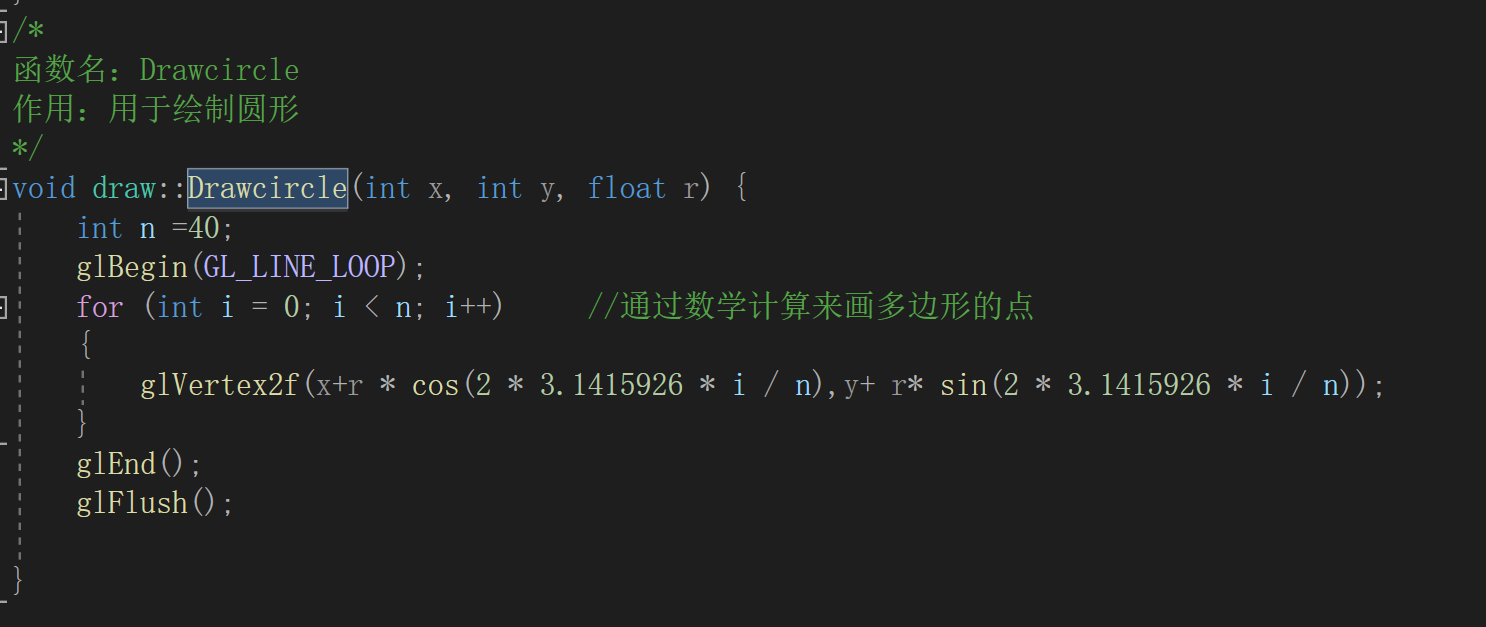
Op=6:选择绘制图形颜色

参数描述：button代表触发事件，比如鼠标左键、右键。

State表示状态，比如鼠标左键按下或抬起

X,Y表示有效触发的区域

返回值描述：void

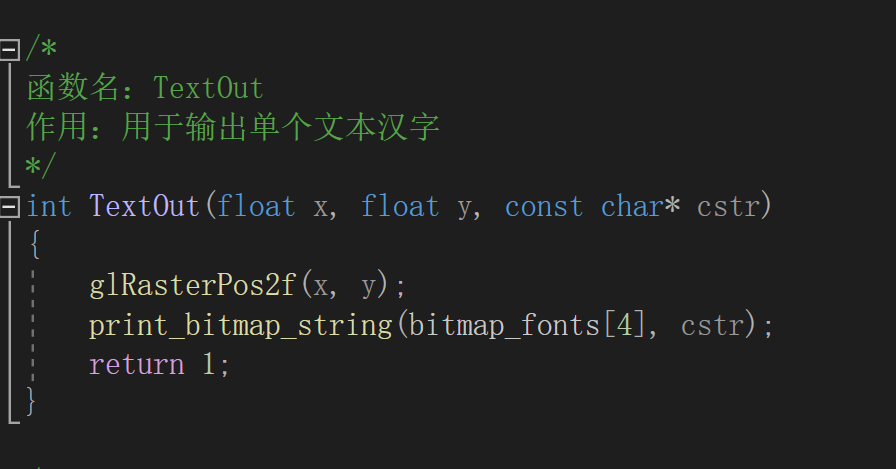


函数名称：Drawcircle(int x,int y,float r);

功能描述；用于绘制圆，由于本身没有直接绘制圆形的函数，这里用极短的线段来模拟圆弧。

参数描述：x,y代表圆心坐标,r代表半径，值都可以通过鼠标触发事件获得。

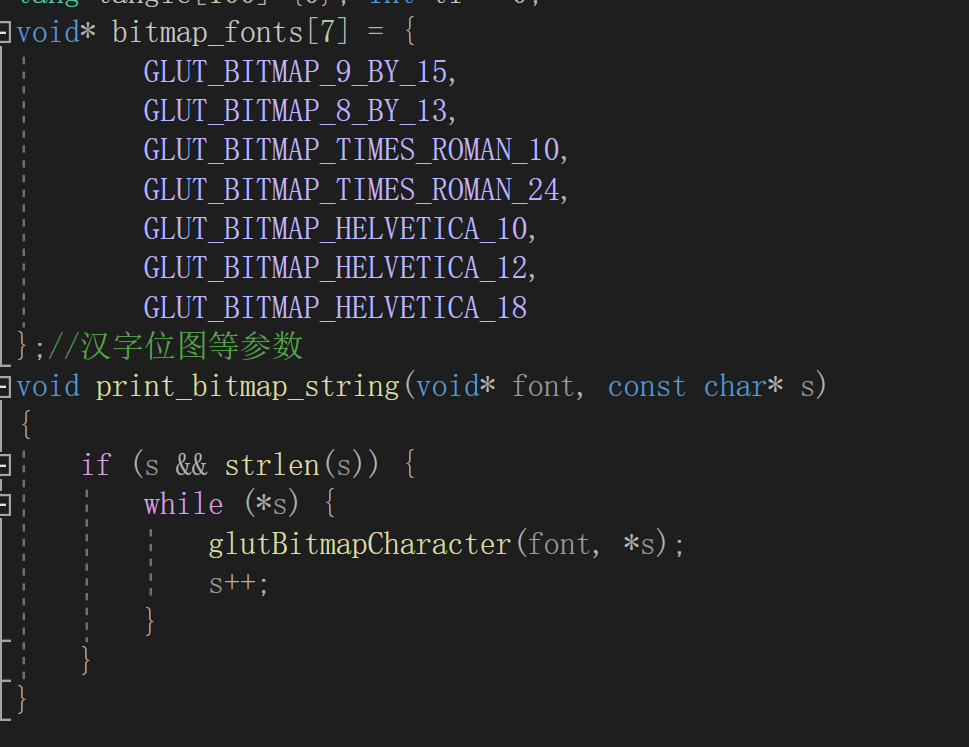
返回值描述：void



函数名称：Drawcircle(float x,float y,const char\* cstr);

功能描述；用于输出文本汉字

参数描述：x,y代表输出位置,cstr代表要输出的汉字，传入bitmap位图后可以打印出汉字。



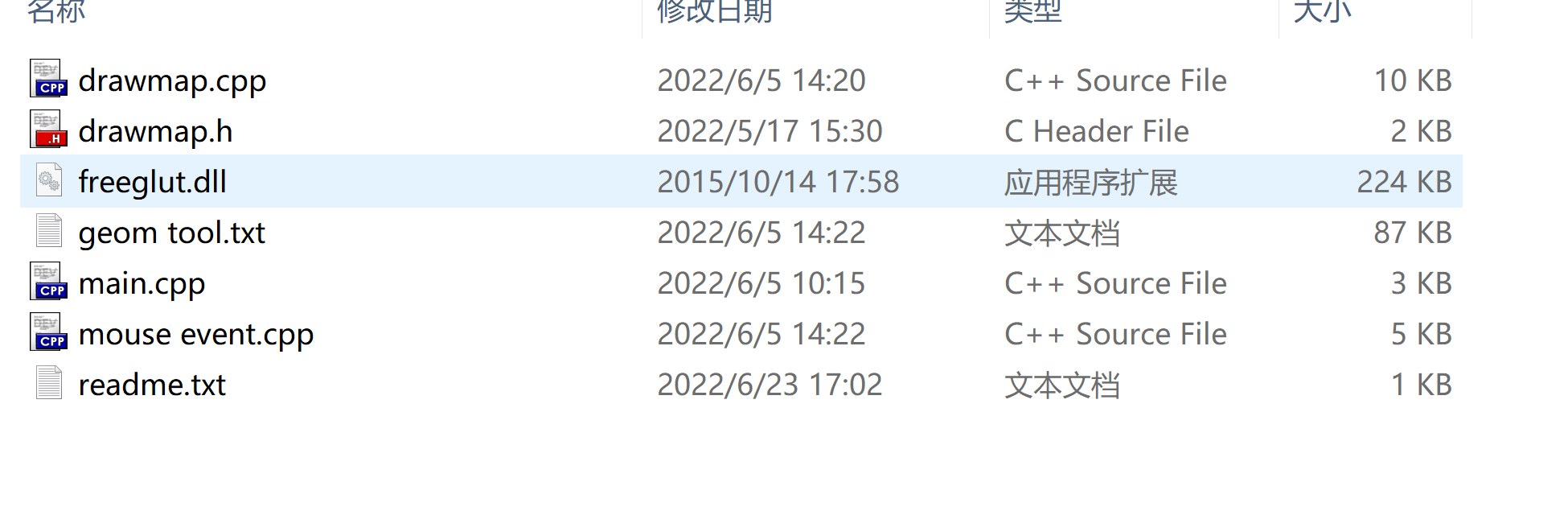
函数名称：print\_bitmap\_string(void\* font,const char\*s);

功能描述；用于输出文本汉字

参数描述：font代表点阵，s表示要输出的字符。

# 部署运行和使用说明

## 编译安装



Code文件夹目录

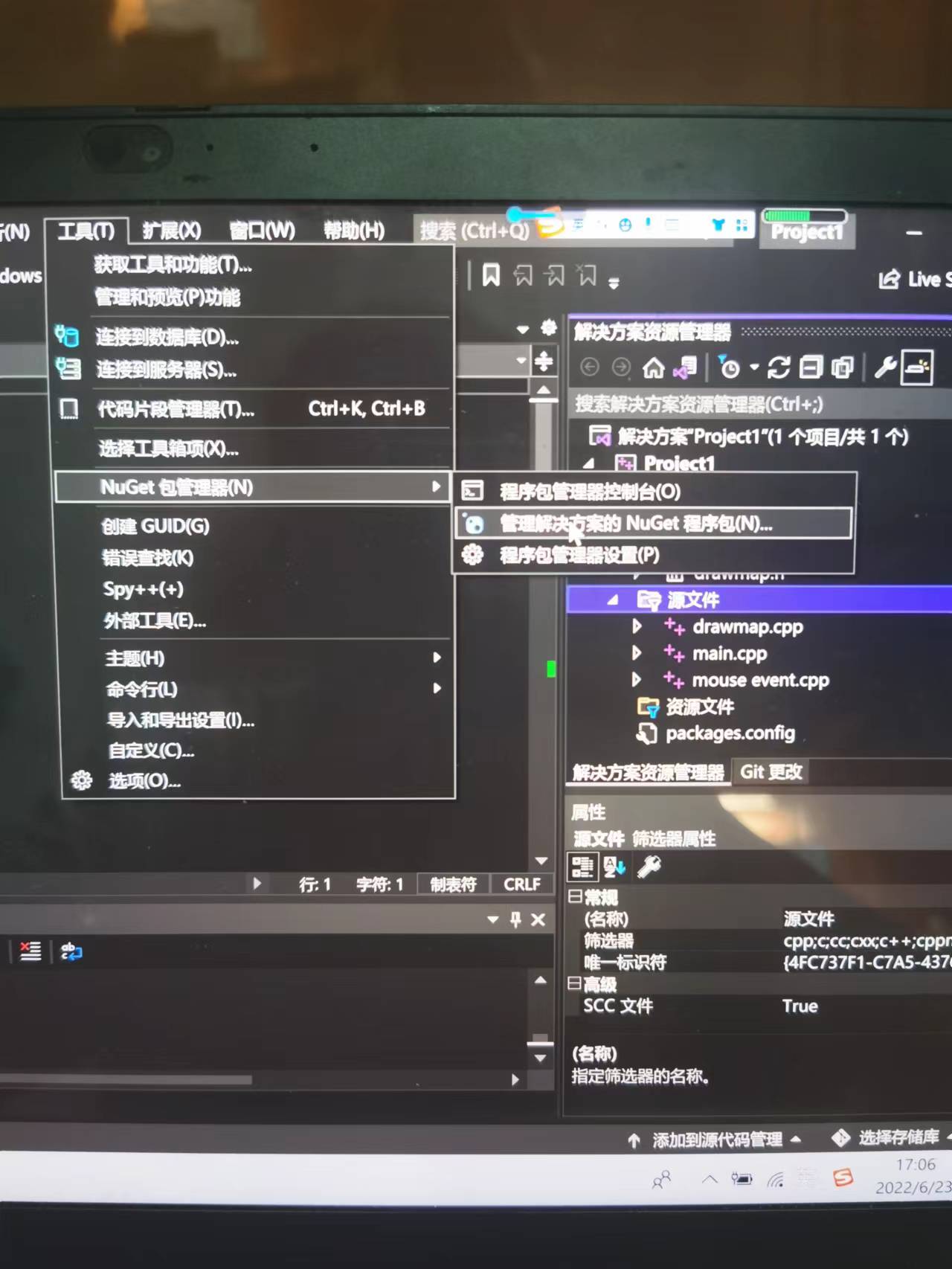
由于PTA无法提交全部文件，故code文件夹内包含头文件、源文件，后缀名修改为txt的可执行文件（geomtool.txt）。

编译过程：

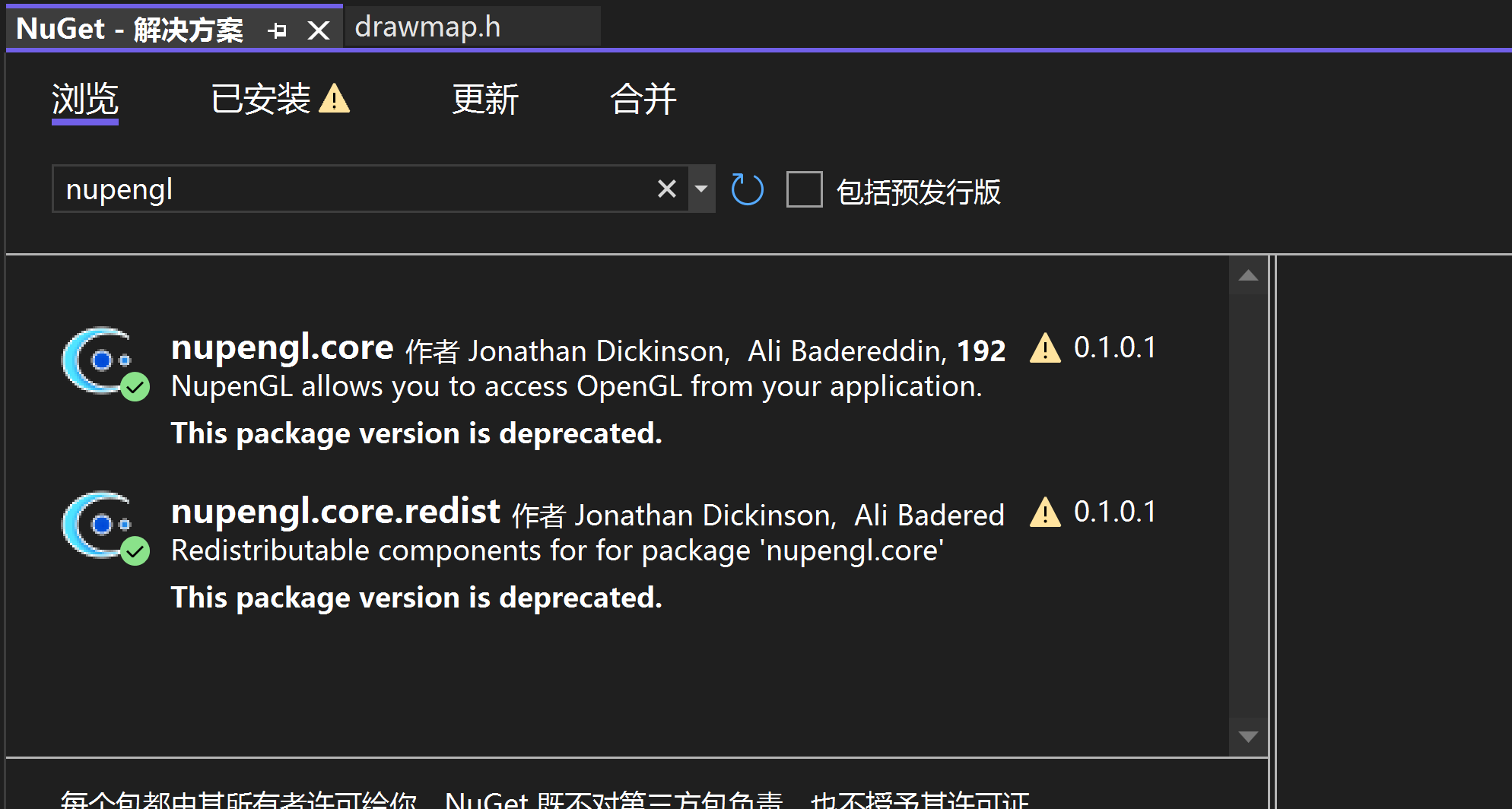
1、新建VS工程，选择空项目



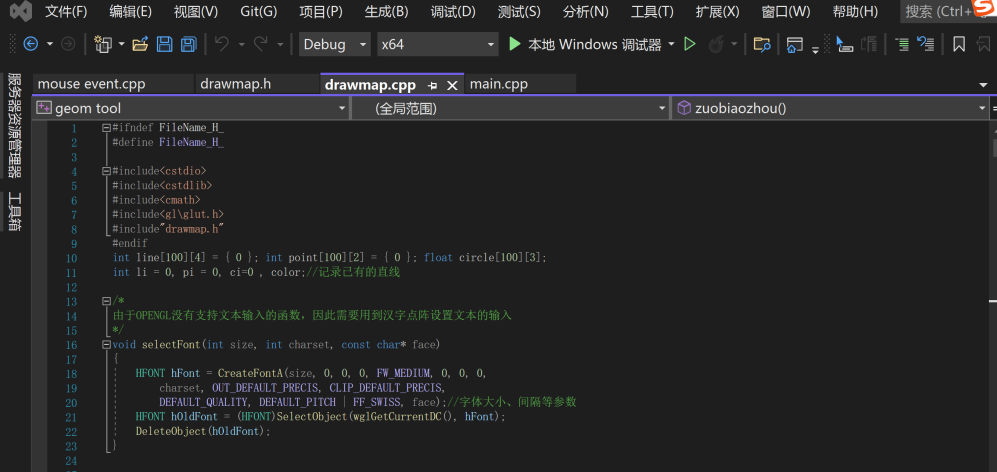
2、成功建立后打开工具中的NuGet包管理器



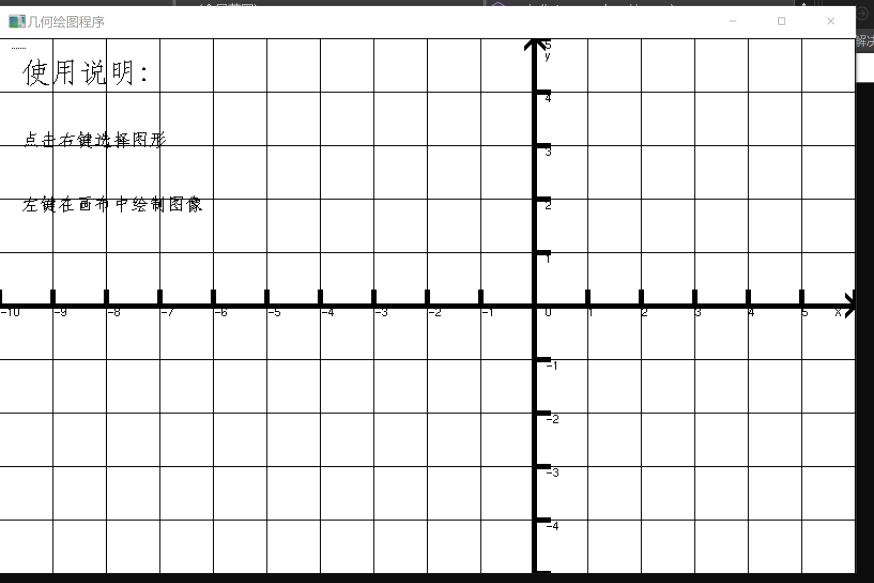
3、搜索nupengl并下载



4、将code文件夹中的1个头文件和3个源文件导入工程，之后编译运行即可。



点击运行（不调试）运行大程序，进入如下界面，表示编译运行成功。

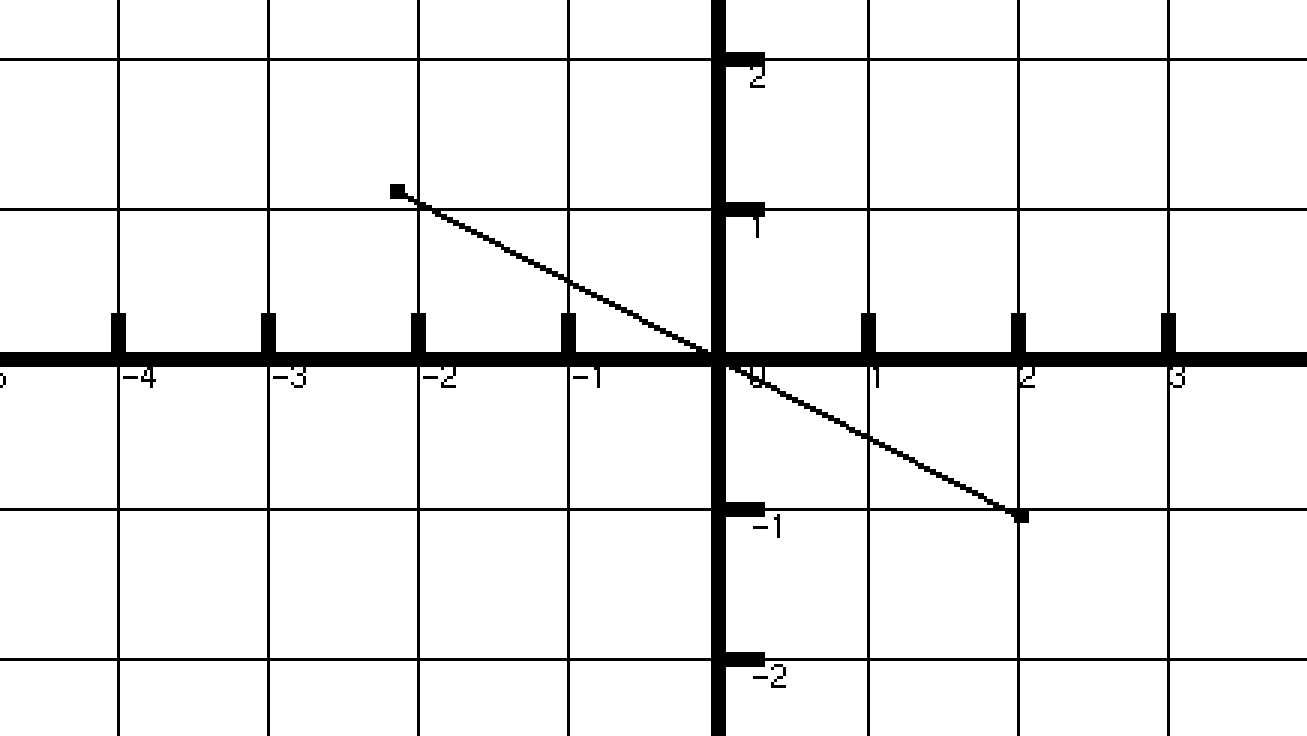


## 运行测试及使用操作

鼠标事件：

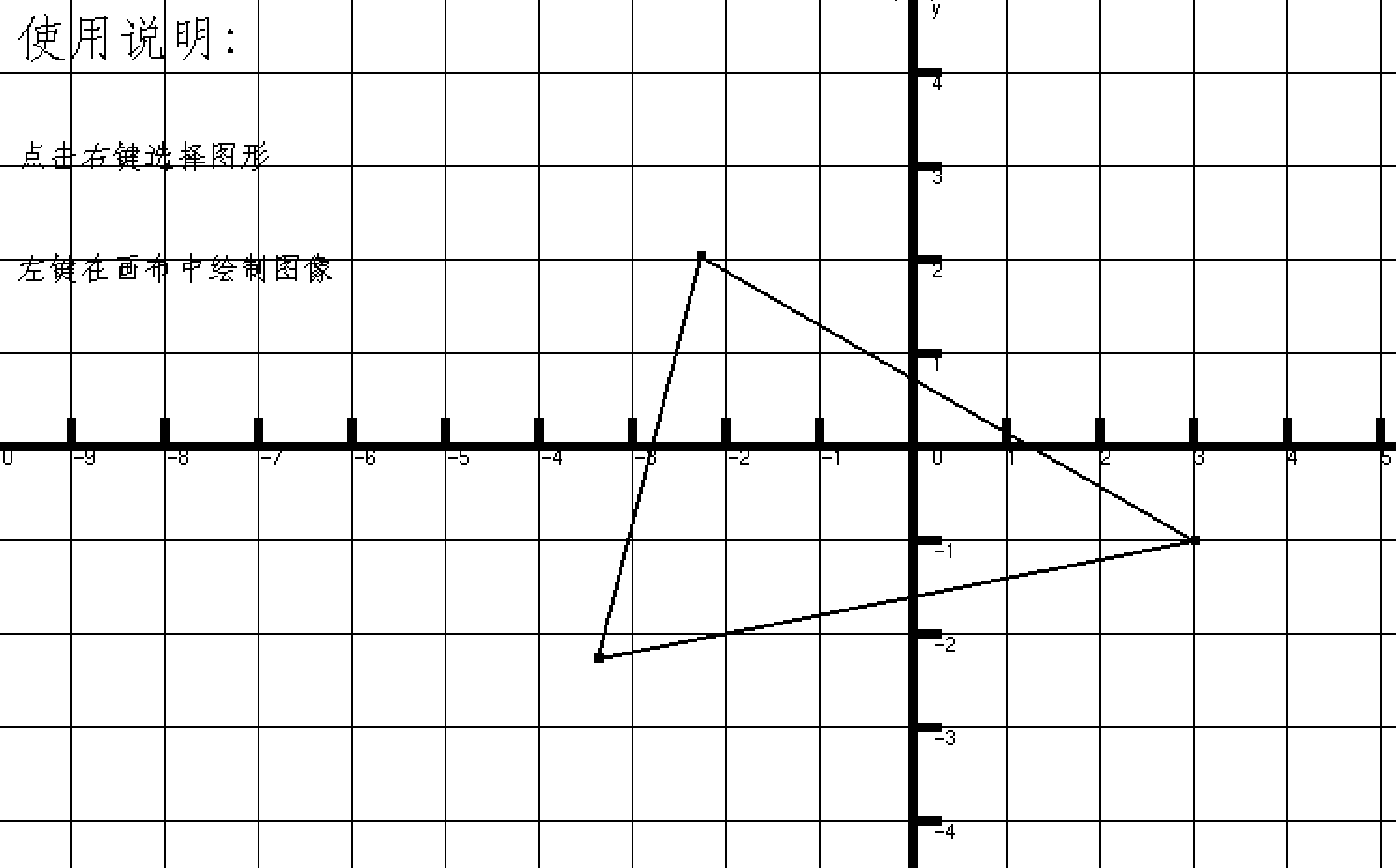
5.2.1绘制线段：

鼠标左键点击可以确定线段顶点



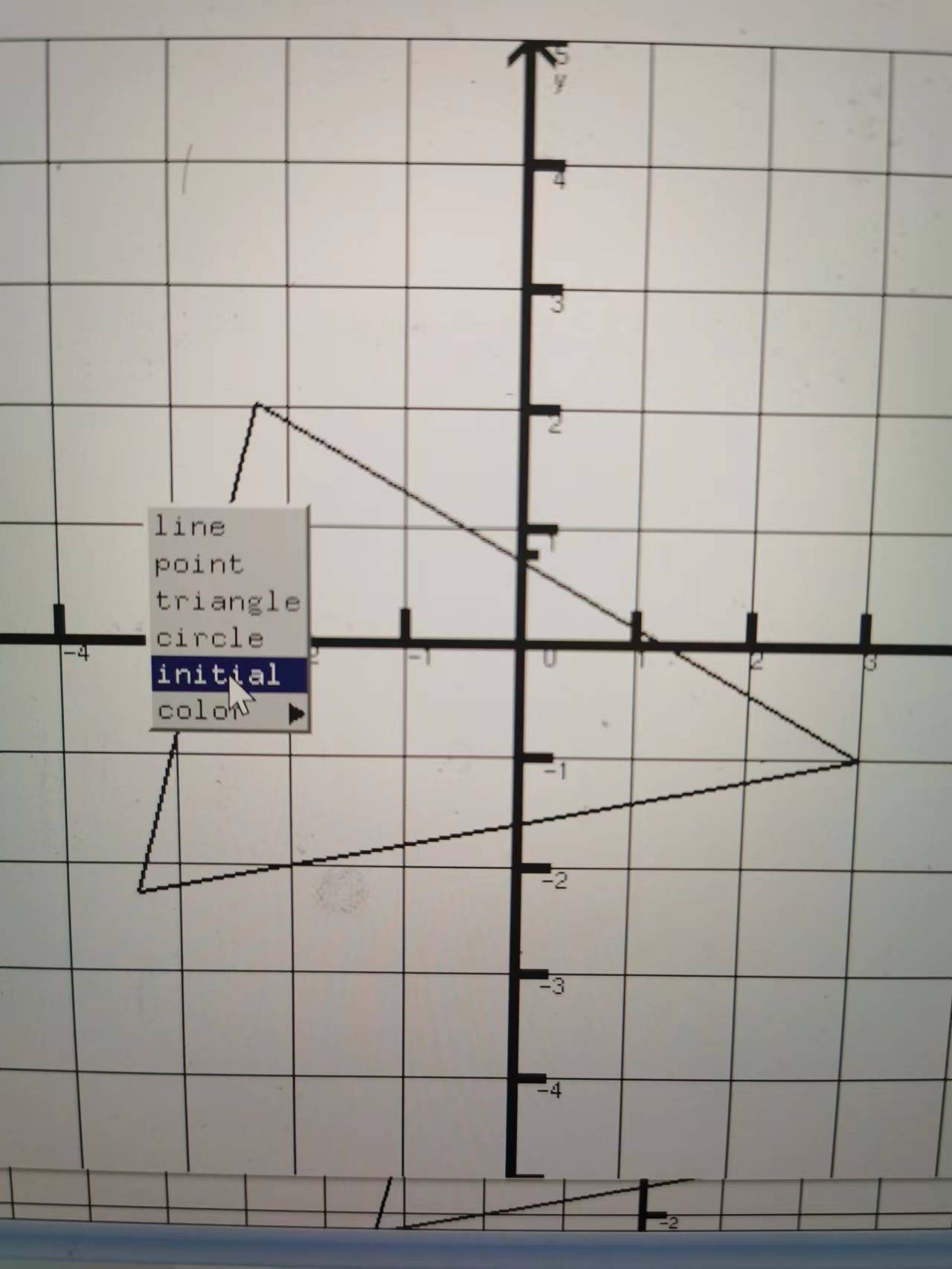
5.2.2、绘制三角形：

用鼠标左键点击确定三个顶点



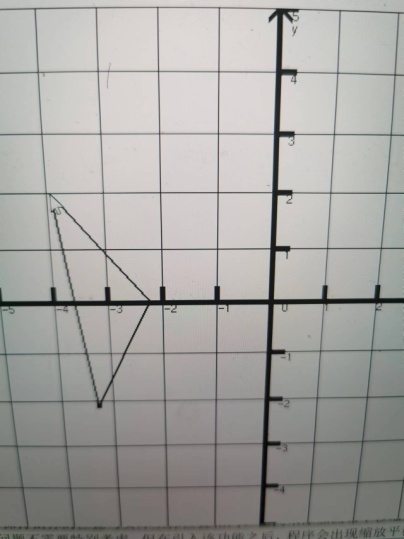
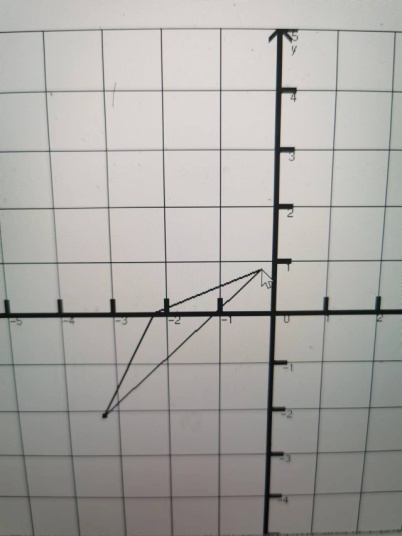
5.2.3、清除：

点击菜单栏的initial，清除屏幕中的几何图形

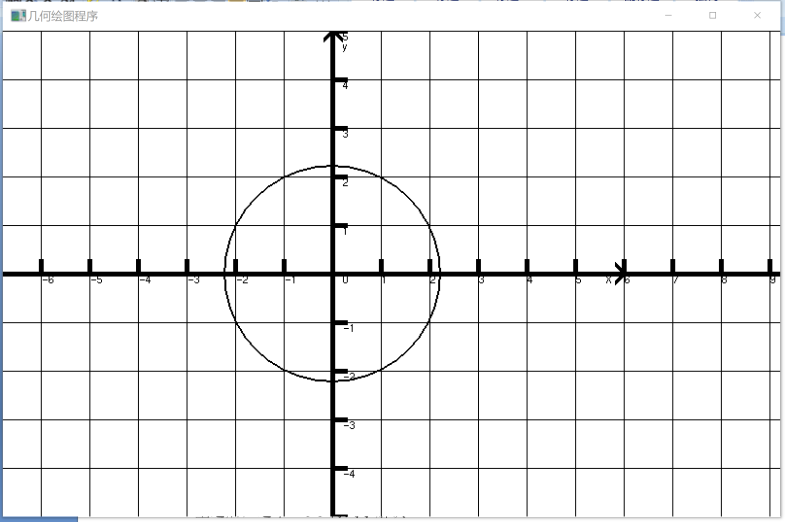


5.2.4：鼠标拖动：

长按左键实现拖动。

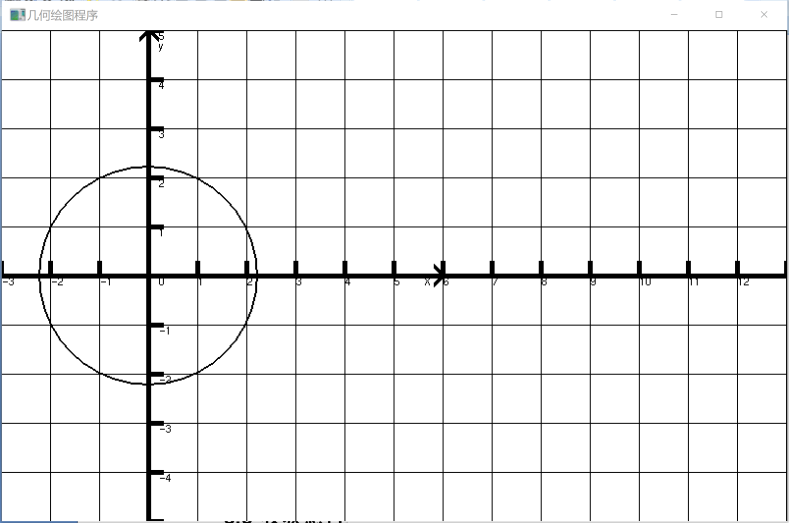
键盘事件：



以这张图为对比

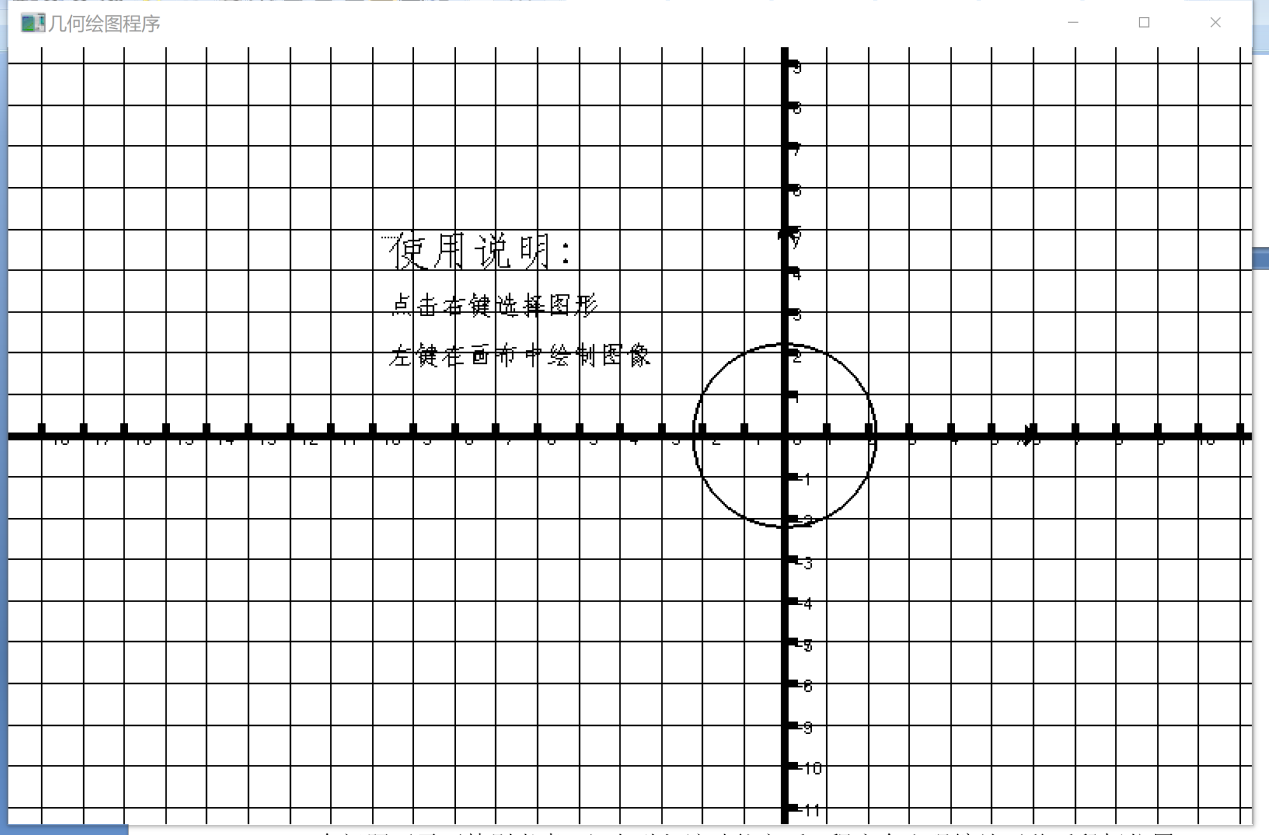
5.2.5：平移：

例如：按W右移

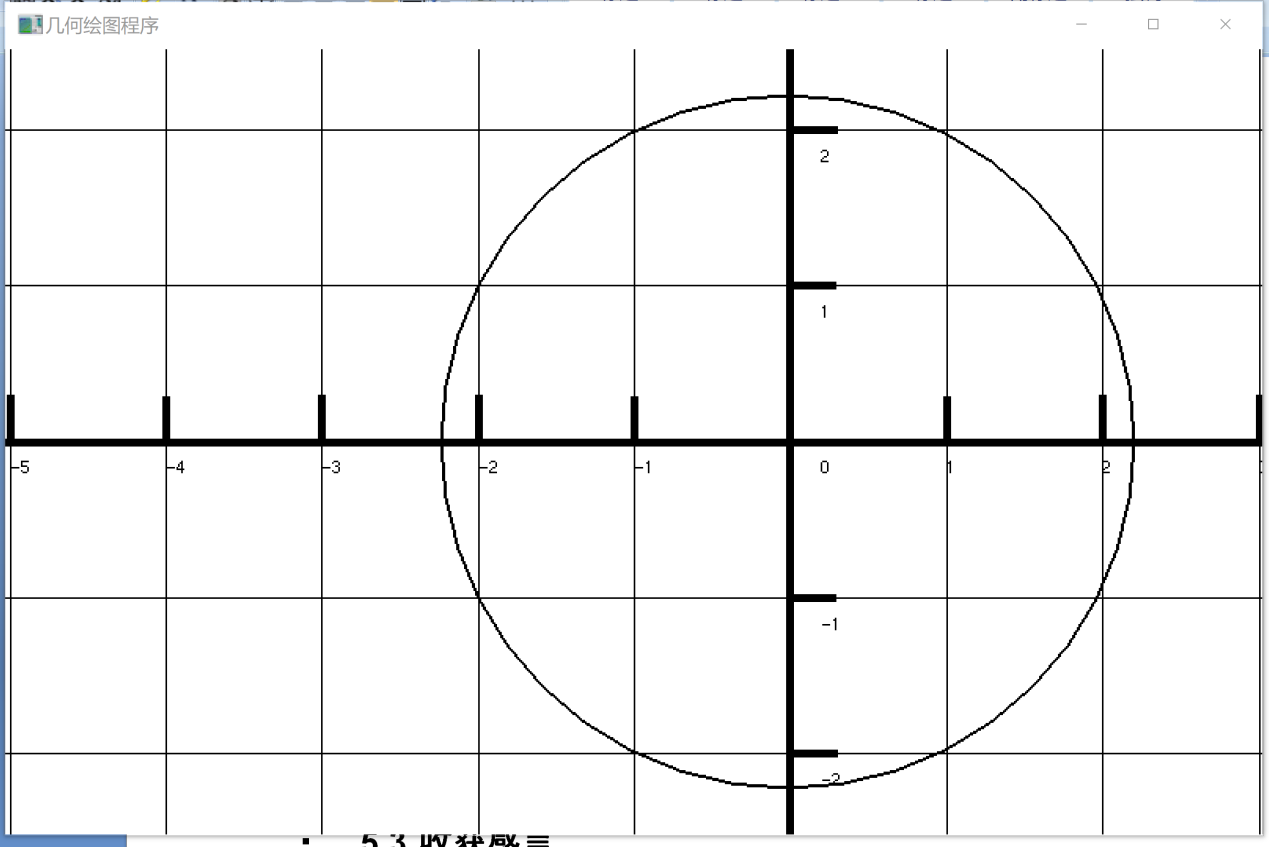


5.2.6、缩放：

按X缩小



按Z放大



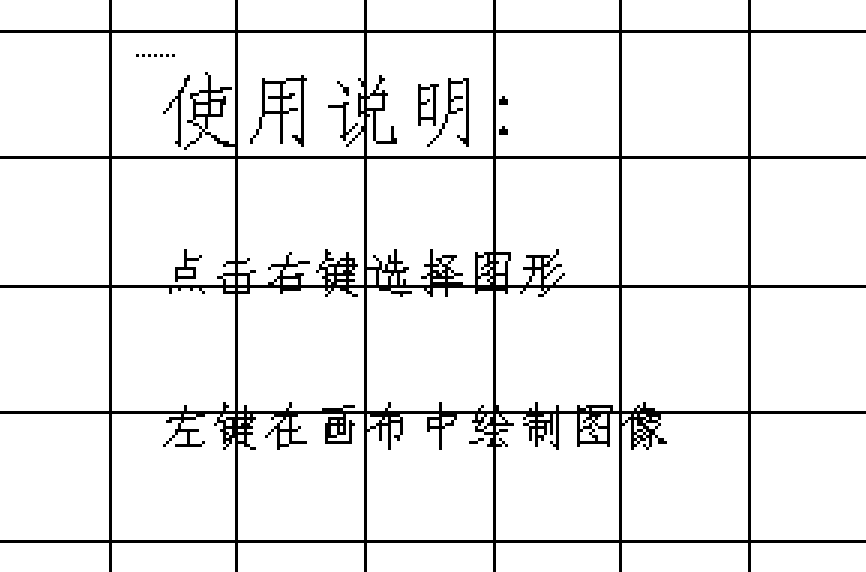
5.2.7、选择颜色：



注意：颜色仅用于区别当前正在画的图形，当新绘制几何图形时，其余图形

会重新变为黑色。

5.2.8、文本显示



利用汉字点阵实现了在屏幕上显示使用说明

## 收获感言

完成了本次大程序，在加深了C++语言理解的同时，我也感受到了封装类的使用对于一个较大程序的便利。比如绘图函数的封装以及继承，使得绘制多边形变得十分简单。大程序中最具有挑战性的是两点：

1、文本函数的实现。由于OPENGL库本身不具有输出文本的功能，因此如果能够自己实现文本的输出，无疑可以使几何绘制程序锦上添花，在网上查阅了相应的资料后，我掌握了汉字点阵的知识，并成功将其运用到程序当中，实现了文本的输出。

2、保持鼠标的位置与实际坐标位置相同。在尚未添加画布平移缩放功能时，这个问题不需要特别考虑，但在引入该功能之后，程序会出现缩放平移后鼠标位置发生偏移，此时再想绘图会变得十分困难。分析后发现是由于每次平移固定距离，但缩放后两点之间的距离发生了改变，也就导致了鼠标偏移。在引入缩放倍数变量之后，这个BUG得到了解决。

# 参考文献资料

1、OOP课程提供的OPENGL使用详细说明。

2、OpenGL函数库及配套工具集

网址：https://blog.csdn.net/fengdos/article/details/78186981

3、OpenGL开发库的详细介绍

网址：https://blog.51cto.com/u\_15329201/3417738

4、Opengl鼠标交互函数glutMouseFunc()函数介绍

网址：https://blog.51cto.com/u\_12136715/2953725