

浙江大学

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

<b>1</b>	<b>大程序简介 .....</b>	<b>4</b>
1.1	选题背景及意义 .....	4
1.2	目标要求 .....	4
1.3	术语说明 .....	4
<b>2</b>	<b>需求分析 .....</b>	<b>4</b>
2.1	业务需求 .....	4
2.2	功能需求 .....	5
2.3	数据需求 .....	5
2.4	性能需求 .....	5
<b>3</b>	<b>类库已有功能分析 .....</b>	<b>6</b>
3.1	总体架构设计 .....	6
3.2	类体系设计 .....	6
3.3	主要类设计 .....	7
3.4	源代码文件组织设计 .....	7
3.5	关键功能类及函数设计描述 .....	8
<b>4</b>	<b>新设计类功能说明 .....</b>	<b>9</b>
4.1	总体架构设计 .....	9
4.2	类模块体系设计 .....	10
4.3	数据结构类设计 .....	10
4.4	源代码文件组织设计 .....	10
4.5	重点类及函数设计描述 .....	11
<b>5</b>	<b>部署运行和使用说明 .....</b>	<b>14</b>
5.1	编译安装 .....	14
5.2	运行测试 .....	16
5.3	使用操作 .....	错误！未定义书签。
5.4	收获感言 .....	21
<b>6</b>	<b>参考文献资料 .....</b>	<b>22</b>

# Geometry tool 大程序设计

## 1 大程序简介

### 1.1 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。

### 1.2 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。

### 1.3 术语说明

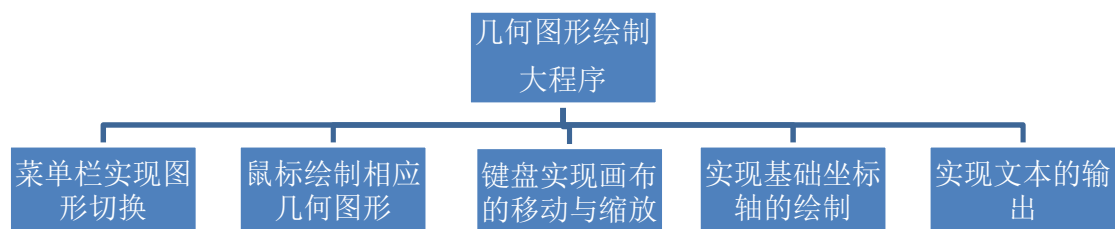
类 (class) 和对象 (object) 是两种以计算机为载体的计算机语言的合称。对象是对客观事物的抽象，类是对对象的抽象。类是一种抽象的数据类型。它们的关系是，对象是类的实例，类是对象的模板。

## 2 需求分析

### 2.1 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。

## 2.2 功能需求



## 2.3 数据需求

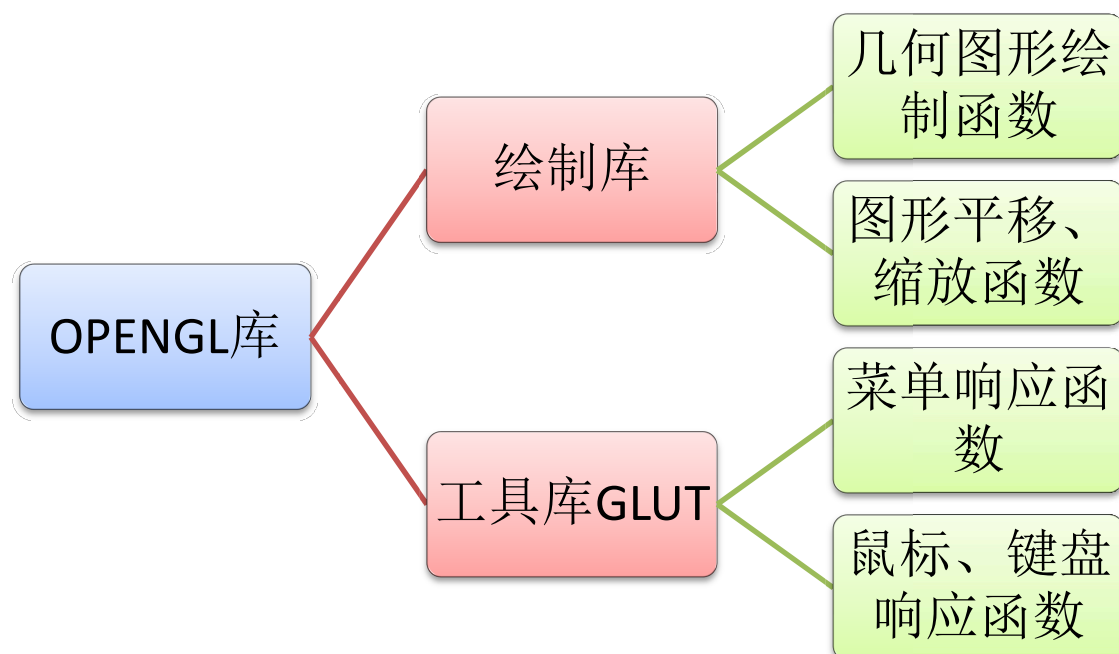
- 1、point 类：绘图程序中最基本的类，有两个数据成员，x 与 y，用于表示点在画布上的坐标。
- 2、Line 类：用于表示线段，使用公有继承 point 类，用来表示线段两端的点。
- 3、triangle 类：用于表示三角形，使用公有继承 point 类，表示三角形的顶点。
- 4、circle 类：用于表示原型，使用公有继承 point 类，用来表示圆心。
- 5、draw 类：将绘制线段、圆等函数封装到一个类当中去。
- 6、text 类：由于 OPENGL 库没有输出文字的功能，因此额外实现一个可以输出文字的类。

## 2.4 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。

### 3 类库已有功能分析

#### 3.1 总体架构设计



#### 3.2 类体系设计



### 3.3 主要类设计



### 3.4 源代码文件组织设计

<文件目录结构>

1) 文件函数结构

.vs	2022/5/2 10:00	文件夹	
packages	2022/4/30 10:10	文件夹	
x64	2022/4/30 10:11	文件夹	
drawmap.cpp	2022/5/17 8:34	C++ Source File	10 KB
drawmap.h	2022/5/17 15:30	C Header File	2 KB
GeometryTool.sln	2022/4/30 10:09	Visual Studio Sol...	2 KB
main.cpp	2022/5/17 15:30	C++ Source File	3 KB
mouse event.cpp	2022/5/17 15:21	C++ Source File	5 KB
packages.config	2022/4/30 10:10	XML Configurati...	1 KB
鼠标交互.vcxproj	2022/5/1 18:03	VC++ Project	9 KB
鼠标交互.vcxproj.filters	2022/5/1 18:03	VC++ Project Fil...	2 KB
鼠标交互.vcxproj.user	2022/4/30 10:09	Per-User Project...	1 KB

图一

大程序共有有一个头文件，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。

```

extern int line[100][4] , point[100][2], color;
extern float circle[100][3];
extern float r;
extern int li , pi , ti , ci;
extern double dz, dx, dy;
extern int m0, m1, n0, n1, op , flag; //声明全局变量, 起始坐标和终止坐标

class tang {
public:
    Point a, b, c;
    tang(int m0 = 0, int n0 = 0, int m1 = 0, int n1 = 0, int m2 = 0, int n2 = 0) { a.x = m0; a.y = n0; b.x = m1; b.y = n1; c.x = m2; c.y = n2; }
};
extern tang tangle[100];

class maping {
public:
    void drawmap1();
    void zuobiao Zhou1();
};

```

图二

### 3.5 关键功能类及函数设计描述

```

class draw:public Point{
public:
    void DrawLine(int, int, int, int); //实际上画直线的函数
    void Drawtangle(int, int, int, int, int, int);
    void Drawcircle(int, int, float);
};

```

绘图函数类

```

/*
mainMenu函数用于菜单的选择, 用于绘制图形的确定以及画布的初始化;
1为线段, 2为点, 3为三角形, 4为圆, 5为清屏, 6为选择颜色
*/
void mainMenu(int id)
{
    switch (id)
    {
    case 1:
        op = 1;
        drawmap();
        break;
    case 2:
        op = 2;
        drawmap();
        break;
    case 3:
        op = 3;
        drawmap();
        break;
    case 4:
        op = 4;
        drawmap();
        break;
    }
}

```

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



```
glClearColor(1, 1, 1, 1); //设置绘制窗口颜色为白色  
glClear(GL_COLOR_BUFFER_BIT); //清除窗口内容
```

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

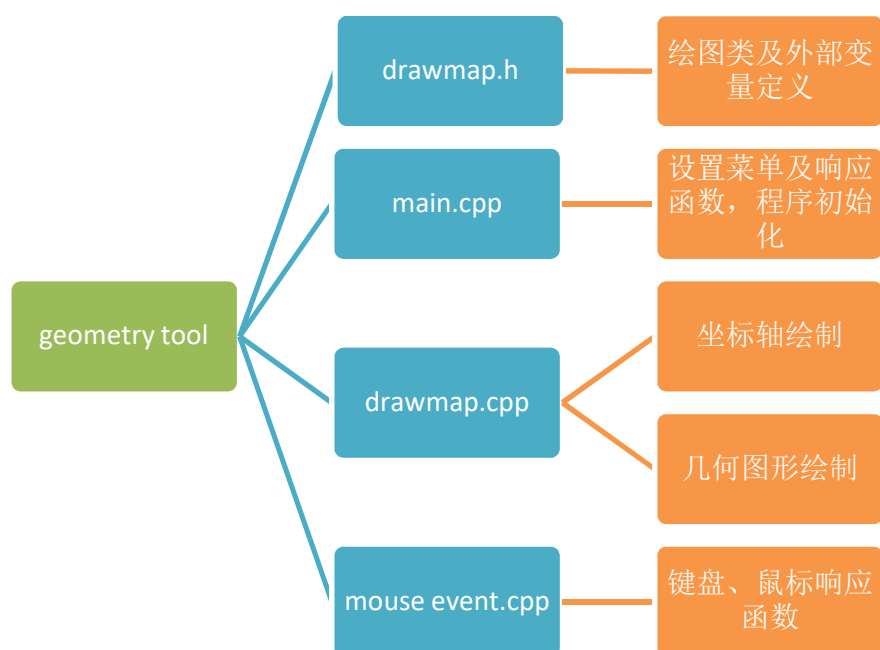
返回值描述: `void`

```
glutKeyboardFunc(myKeyboard);  
glutMouseFunc(mymouse); //鼠标监听回调函数  
glutMotionFunc(dragmouse); //鼠标拖动
```

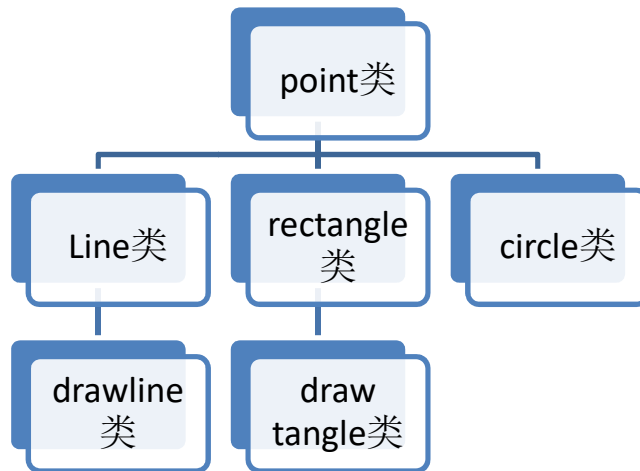
函数描述: OPENGGL 库自带的响应函数, 大程序中共用到了上图三种: 键盘响应、鼠标点击响应、鼠标拖动响应。

## 4 新设计类功能说明

### 4.1 总体架构设计



## 4.2 类模块体系设计



## 4.3 数据结构类设计

```
int line[100][4] = { 0 }; int point[100][2] = { 0 }; float circle[100][3];
int li = 0, pi = 0, ci=0, color;//记录已有的直线
```

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。

## 4.4 源代码文件组织设计

<文件目录结构>

1) 文件函数结构

.vs	2022/5/2 10:00	文件夹	
packages	2022/4/30 10:10	文件夹	
x64	2022/4/30 10:11	文件夹	
drawmap.cpp	2022/5/17 8:34	C++ Source File	10 KB
drawmap.h	2022/5/17 15:30	C Header File	2 KB
GeometryTool.sln	2022/4/30 10:09	Visual Studio Sol...	2 KB
main.cpp	2022/5/17 15:30	C++ Source File	3 KB
mouse event.cpp	2022/5/17 15:21	C++ Source File	5 KB
packages.config	2022/4/30 10:10	XML Configurati...	1 KB
鼠标交互.vcxproj	2022/5/1 18:03	VC++ Project	9 KB
鼠标交互.vcxproj.filters	2022/5/1 18:03	VC++ Project Fil...	2 KB
鼠标交互.vcxproj.user	2022/4/30 10:09	Per-User Project...	1 KB

图一

大程序共有有一个头文件，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。

## 4.5 重点类及函数设计描述

```
class maping {
    void drawmap1();
    void zuobiaozhou1();
};
```

坐标图类：用于绘制屏幕中最初的坐标系。

```
class Point {
public:
    int x, y;
    Point(int x1 = 0, int y1 = 0) {
        x = x1; y = y1;
    }
};
```

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

```
/* 鼠标拖动
函数名: dragmouse
作用: 实现在画点、三角形时实现线段随点的移动而移动
, 提升绘图体验
*/
void dragmouse(int x, int y) {
    {
        glClear(GL_COLOR_BUFFER_BIT);
        drawmap();
        if (flag == 1) {
            m1 = (x - dx - 500) / dz + 500;
            n1 = (y - dy - 250) / dz + 250;
        }
        setPixel(m1, n1);

        if (op == 1 || op == 4) D.DrawLine(m0, n0, m1, n1); //画线
        if (op == 3 && flag == 2) {
            m2 = (x - dx - 500) / dz + 500;
            n2 = (y - dy - 250) / dz + 250;
            D.Drawtriangle(m0, n0, m1, n1, m2, n2);
            glFlush();
        }
    }
}
```

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

```

}
/*
函数名: drawmap
作用: 用于绘制图页面, 同时支持清屏后重新
恢复坐标轴
*/
void drawmap() {

```

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

```

// 鼠标监听, 画点
void mymouse(int button, int state, int x, int y) {
    /*
    鼠标触发事件, 点击左下角的色块可选择相应的颜色
    */

    if (op == 1 || op == 2) {
        op1 = op;
        if (flag == 0) {
            if (button == GLUT_LEFT_BUTTON && state == GLUT_UP ) {
                m0 = (x - dx-500) / dz +500;
                n0 = (y - dy-250) / dz +250;
                setPixel(m0, n0);
                flag = 1;

                if (op == 2) {
                    point[pi][0] = m0; point[pi][1] = n0; pi++;
                }

                glBegin(GL_POINTS); // 对鼠标点击过的点标记, 便于使用者确定点击过的点
                glPointSize(5.0f);
                glVertex2f(m0, n0);
                glEnd();
                glFlush();
            }
        }
    }
}

```

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: Drawcircle
作用: 用于绘制圆形
*/
void draw::Drawcircle(int x, int y, float r) {
    int n = 40;
    glBegin(GL_LINE_LOOP);
    for (int i = 0; i < n; i++) //通过数学计算来画多边形的点
    {
        glVertex2f(x+r * cos(2 * 3.1415926 * i / n), y+ r* sin(2 * 3.1415926 * i / n));
    }
    glEnd();
    glFlush();
}

```

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: TextOut
作用: 用于输出单个文本汉字
*/
int TextOut(float x, float y, const char* cstr)
{
    glRasterPos2f(x, y);
    print_bitmap_string(bitmap_fonts[4], cstr);
    return 1;
}

```

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。

```

//汉字位图等参数
void* bitmap_fonts[7] = {
    GLUT_BITMAP_9_BY_15,
    GLUT_BITMAP_8_BY_13,
    GLUT_BITMAP_TIMES_ROMAN_10,
    GLUT_BITMAP_TIMES_ROMAN_24,
    GLUT_BITMAP_HELVETICA_10,
    GLUT_BITMAP_HELVETICA_12,
    GLUT_BITMAP_HELVETICA_18
};
void print_bitmap_string(void* font, const char* s)
{
    if (s && strlen(s)) {
        while (*s) {
            glutBitmapCharacter(font, *s);
            s++;
        }
    }
}

```

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。

## 5 部署运行和使用说明

### 5.1 编译安装

名称	修改日期	类型	大小
 drawmap.cpp	2022/6/5 14:20	C++ Source File	10 KB
 drawmap.h	2022/5/17 15:30	C Header File	2 KB
 freeglut.dll	2015/10/14 17:58	应用程序扩展	224 KB
 geom tool.txt	2022/6/5 14:22	文本文档	87 KB
 main.cpp	2022/6/5 10:15	C++ Source File	3 KB
 mouse event.cpp	2022/6/5 14:22	C++ Source File	5 KB
 readme.txt	2022/6/23 17:02	文本文档	1 KB

#### Code 文件夹目录

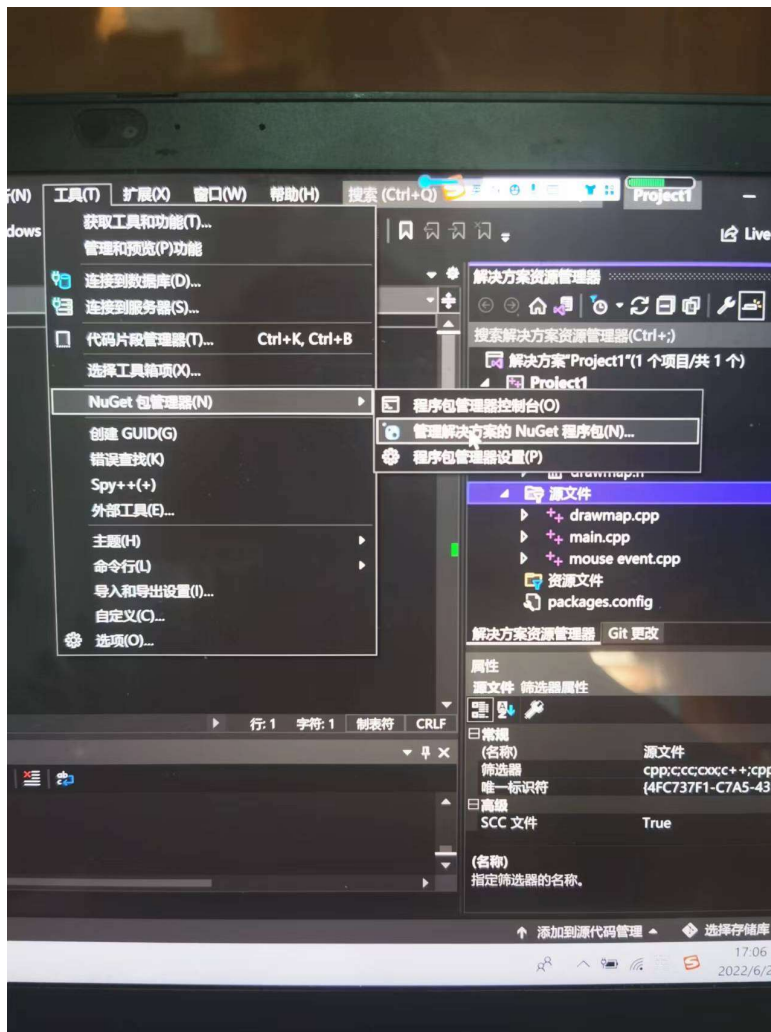
由于 PTA 无法提交全部文件，故 code 文件夹内包含头文件、源文件，后缀名修改为 txt 的可执行文件（geomtool.txt）。

编译过程：

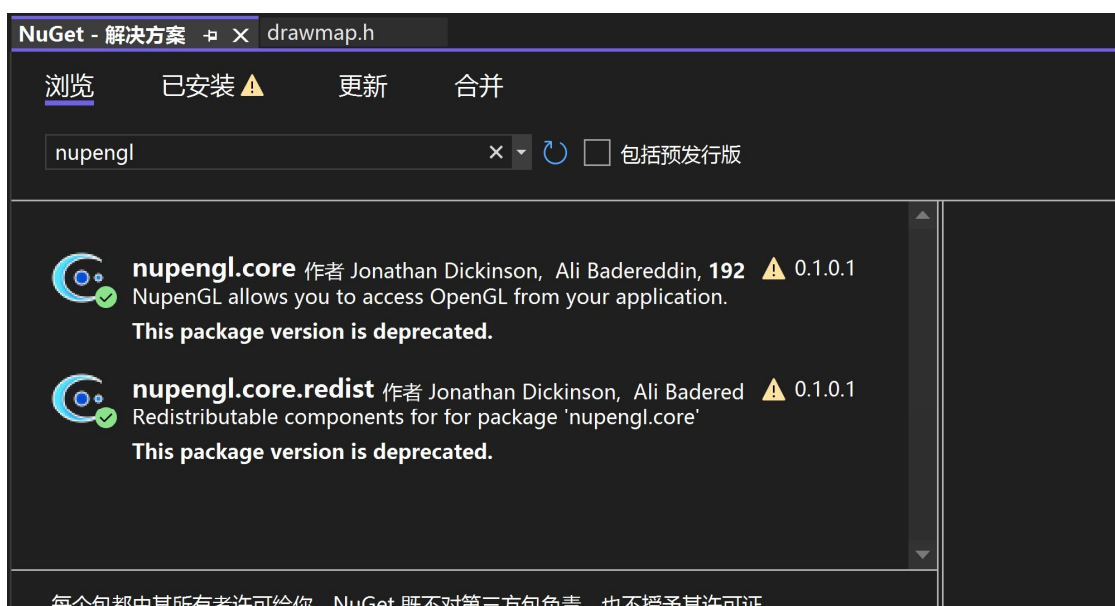
1、新建 VS 工程，选择空项目



## 2、成功建立后打开工具中的 NuGet 包管理器

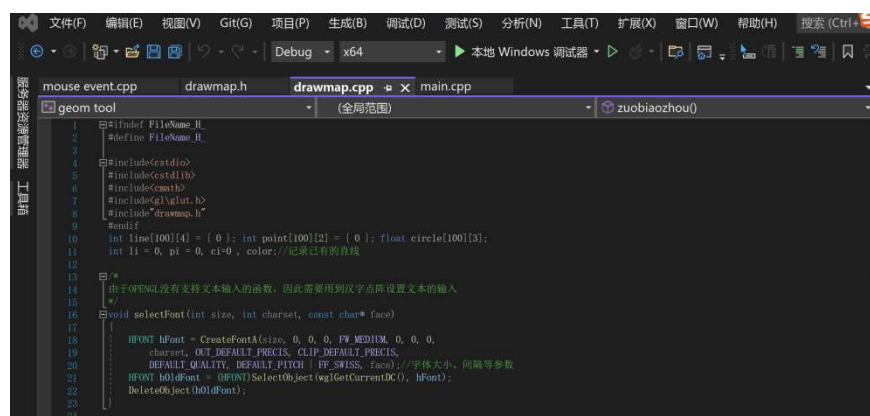


## 3、搜索 nupengl 并下载



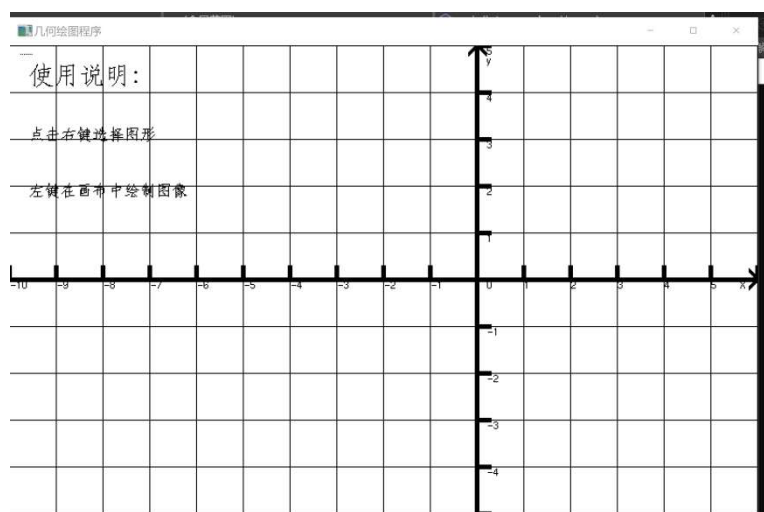


4、将 code 文件夹中的 1 个头文件和 3 个源文件导入工程，之后编译运行即可。



```
1 #ifndef FileName_H
2 #define FileName_H
3
4 #include<stdio.h>
5 #include<stdlib.h>
6 #include<math.h>
7 #include<gl\glut.h>
8 #include"drawmap.h"
9 #endif
10 int line[100][4] = { 0 }; int point[100][2] = { 0 }; float circle[100][3];
11 int li = 0, pi = 0, ci=0, color;//记录已有的直线
12
13 /*
14 由于OPENGL没有支持文本输入的函数，因此需要用到汉字点阵设置文本的输入
15 */
16 void selectFont(int size, int charset, const char* face)
17 {
18     HFONT hFont = CreateFont(size, 0, 0, 0, FW_MEDIUM, 0, 0, 0,
19                             charset, OUT_DEFAULT_PRECIS, CLIP_DEFAULT_PRECIS,
20                             DEFAULT_QUALITY, DEFAULT_PITCH | FF_SWISS, face);//字体大小、风格等参数
21     HFONT hOldFont = (HFONT)SelectObject(wglGetCurrentDC(), hFont);
22     DeleteObject(hOldFont);
23 }
24
```

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。



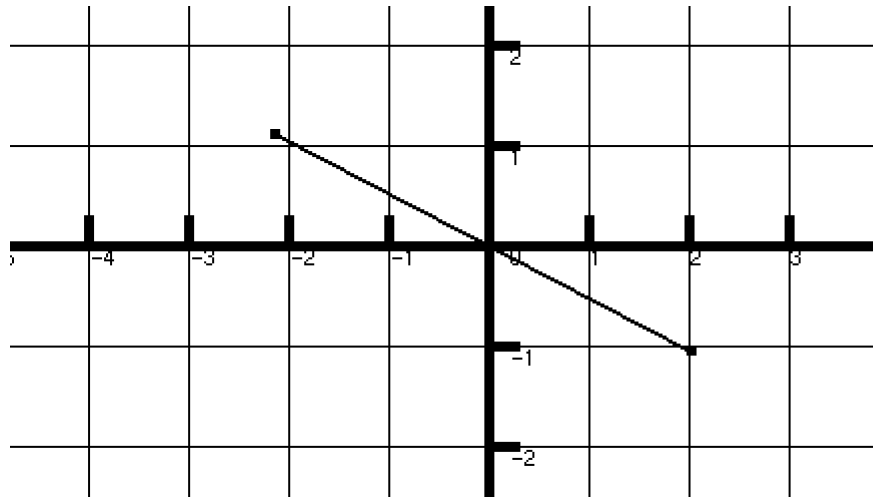
## 5.2 运行测试及使用操作

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





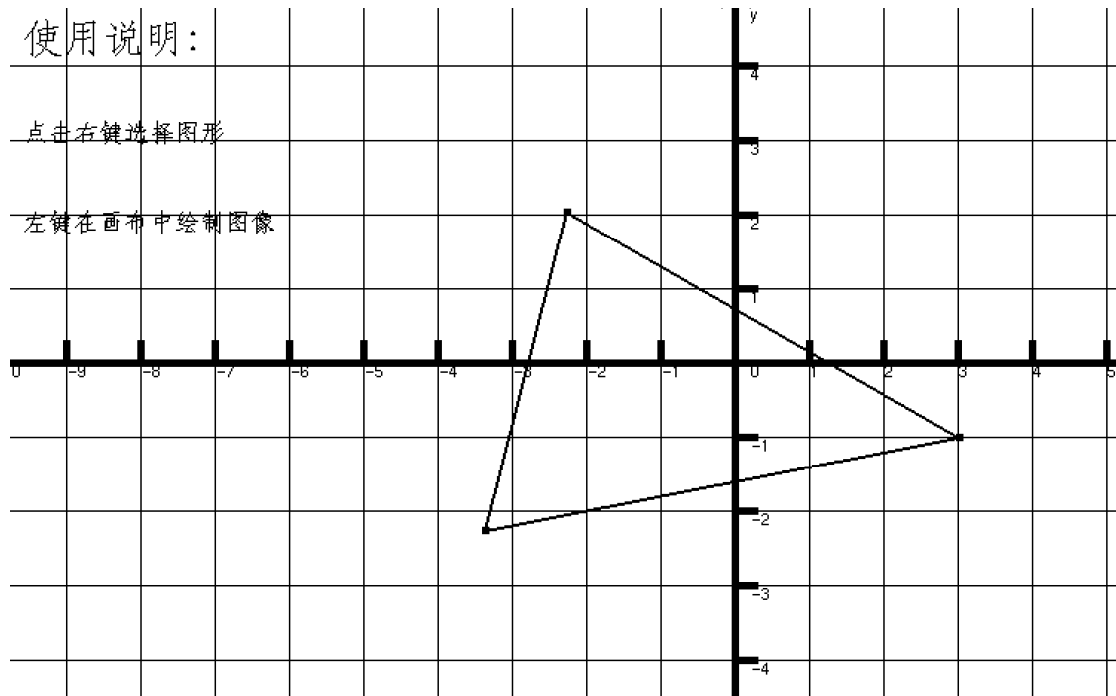
### 5.2.2、绘制三角形：

用鼠标左键点击确定三个顶点

使用说明：

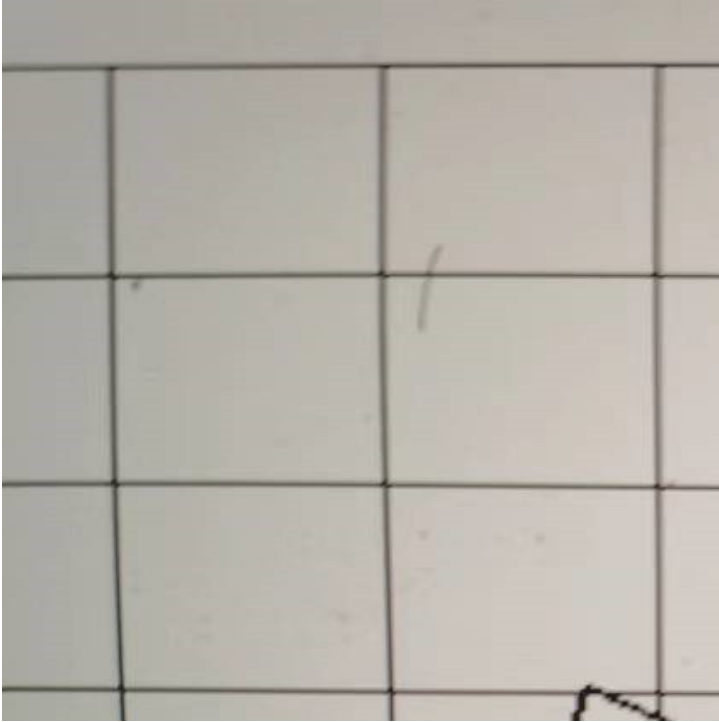
点击右键选择图形

左键在画布中绘制图像

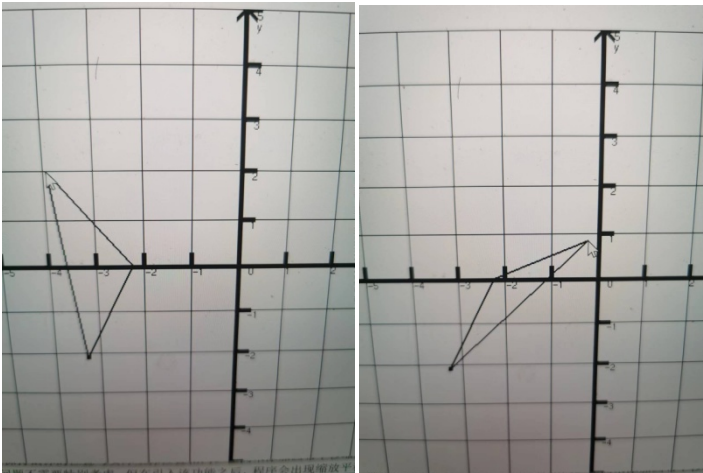


### 5.2.3、清除：

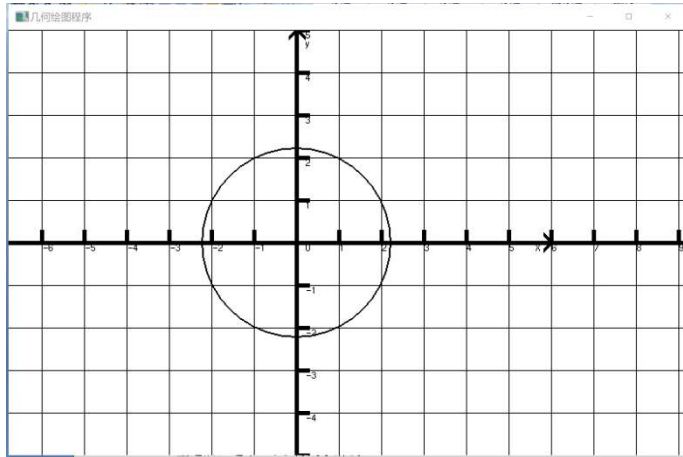
点击菜单栏的 initial，清除屏幕中的几何图形



5.2.4: 鼠标拖动:  
长按左键实现拖动。



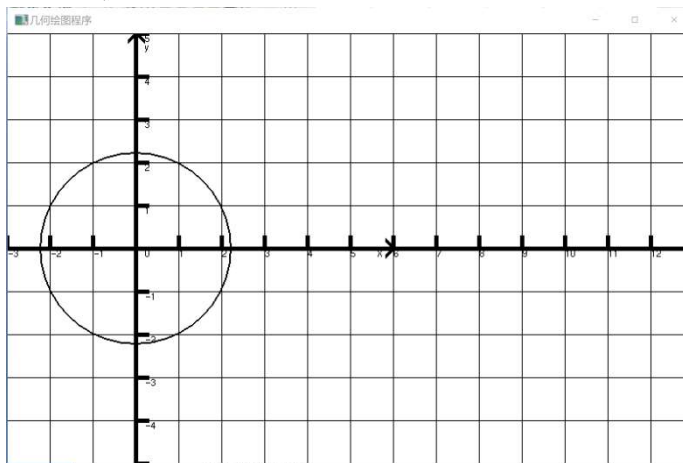
键盘事件:



以这张图为对比

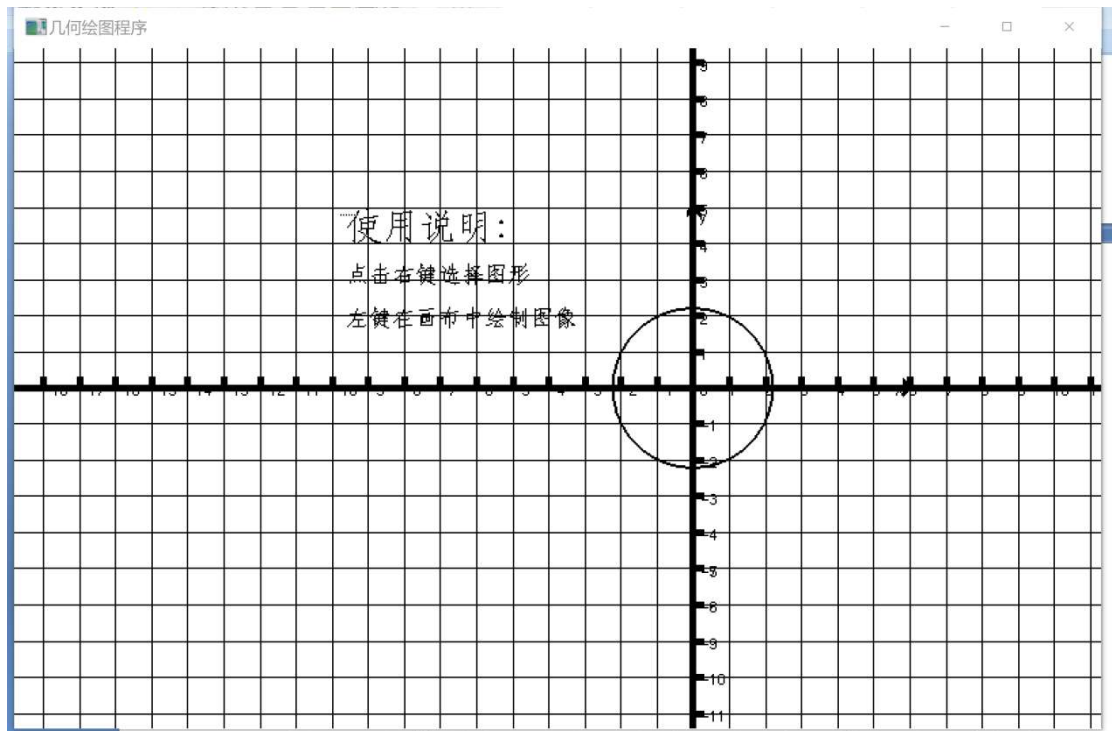
5.2.5: 平移:

例如: 按 W 右移

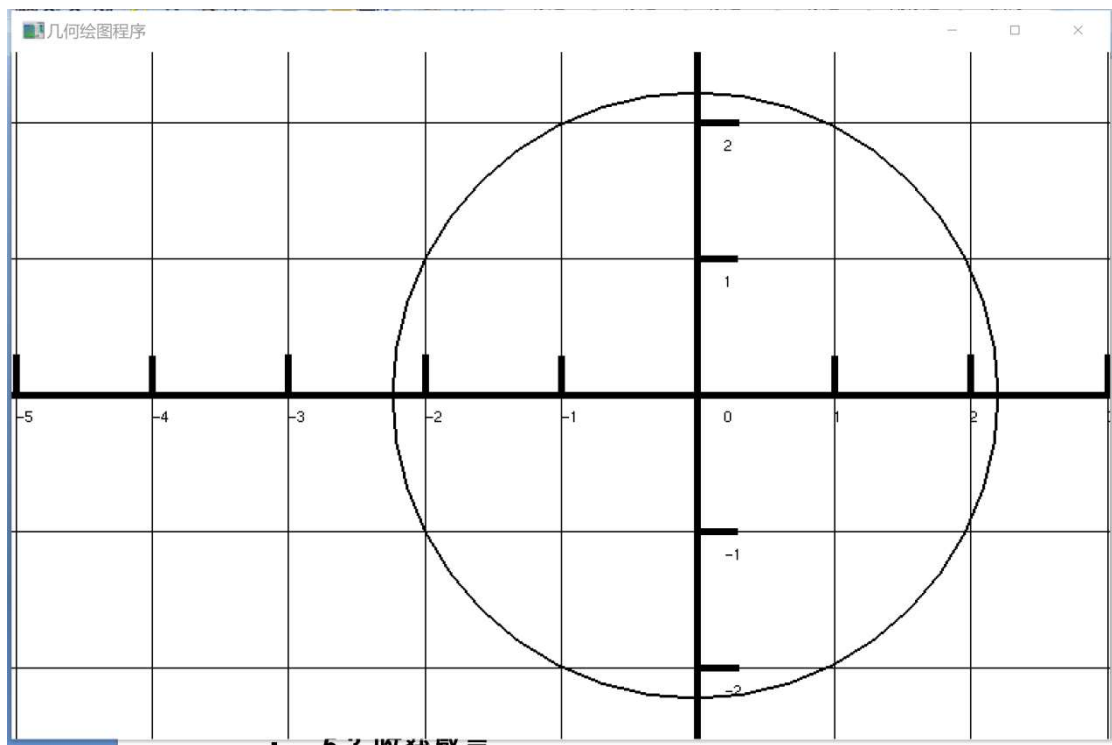


5.2.6、缩放:

按 X 缩小



按 Z 放大

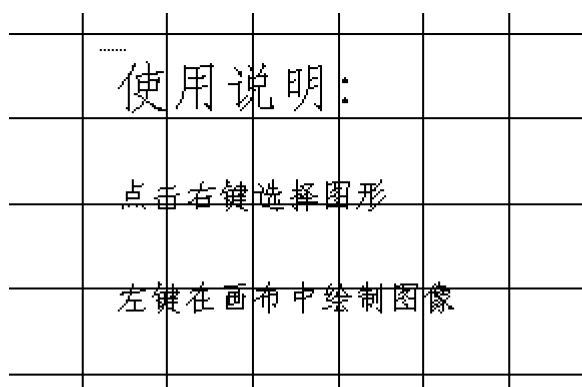


5.2.7、选择颜色:



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

### 5.3 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 OPENGGL 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。

## 6 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](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](https://blog.51cto.com/u_12136715/2953725)