

**面向对象程序设计**

**大 程 序 报 告**



大程名称： 基于OpenGL的几何画板

**姓名 ：**

**学号： 3210103379**

**电话：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 15167970568**

**指导老师： 李际军**

**2021~2022春夏学期 2022 年 12 月 28 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 4](#_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 5](#_Toc71541793)

[3.1 总体架构设计 5](#_Toc71541794)

[3.2 类体系设计 5](#_Toc71541795)

[3.3 主要类设计 5](#_Toc71541796)

[3.4 源代码文件组织设计 6](#_Toc71541797)

[3.5 关键功能类及函数设计描述 6](#_Toc71541798)

[4 新设计类功能说明 6](#_Toc71541799)

[4.1 总体架构设计 6](#_Toc71541800)

[4.2 类模块体系设计 7](#_Toc71541801)

[4.3 数据结构类设计 7](#_Toc71541802)

[4.4 源代码文件组织设计 7](#_Toc71541803)

[4.5 重点类及函数设计描述 7](#_Toc71541804)

[5 部署运行和使用说明 8](#_Toc71541805)

[5.1 编译安装 8](#_Toc71541806)

[5.2 运行测试 8](#_Toc71541807)

[5.3 使用操作 8](#_Toc71541808)

[5.4 收获感言 8](#_Toc71541809)

[6 参考文献资料 8](#_Toc71541810)

基于OpenGL的几何画板大程序设计

# 大程序简介

*<说明:描述大程序选题或开发背景，要实现的程序设计目标与要求、主要术语>*

## 选题背景及意义

选题背景：分析libigl - A simple C++ geometry processing library的代码，理清其类体系；在借鉴吸收的基础上设计一个基于visualstudio和OpenGL的几何建模功能的project。

意义：实现了二维画图（三角形、矩形、圆形等图形）的功能。

## 目标要求

实现交互功能（右键出现弹出式菜单）

实现二维绘图功能（三角形、矩形等图形）

实现保存、加载功能

实现清空画板的功能

## 术语说明

交互绘图功能：用户能够通过鼠标控制绘图的样式、通过键盘实现二维画图

保存、加载功能：保存可以将当前绘制的图形数据保存至特定的文本文件（myproject或Release文件夹下的1.txt文件）；加载：将1.txt文件中的图形数据加载到当前屏幕中。

# 需求分析

*<说明:分析你要开发的大程序的具体功能需求，包括哪些业务逻辑要求、需要哪些功能、哪些数据结构、哪些性能方面的要求>*

## 业务需求

*从业务上，描述实现的大程序是解决什么领域的什么问题，有哪些需求。*

该大程序注重于解决几何建模领域的交互绘制问题，有交互绘制、二维画图、加载保存数据的需求。

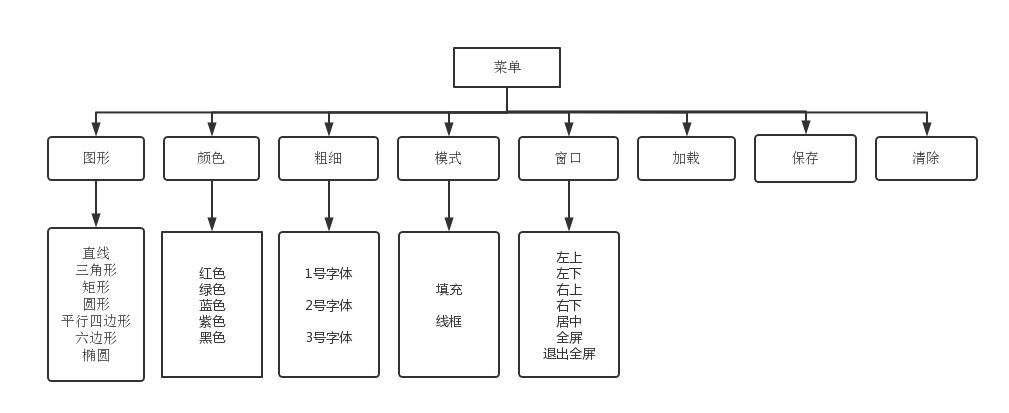
## 功能需求

*从功能上，描述实现的大程序应该有哪些功能。画一个功能模块架构图，然后逐一介绍每个功能。*

1. 菜单功能：提供图形、颜色、粗细、模式、窗口、加载、保存、清除功能
2. 图形选择：选择直线、三角形、矩形、平行四边形、圆形、六边形、椭圆七

种形状进行绘画

1. 颜色选择：选择红、绿、蓝、紫、黑五种颜色的画笔
2. 粗细选择：选择画笔的粗细
3. 模式选择：选择画填充/线框的图形
4. 窗口功能：调节窗口的位置
5. 加载、保存：将特定文件中的图形数据加载到画布上或者将图形数据保存到特定文件中
6. 清除：清空画布



## 数据需求

*从数据上，描述实现的大程序应该包含怎样的数据要求，包括数据、类结构，数据格式以及内容。*

1. 构造点类及其派生类（直线类、三角形类等）来记录绘制各种图形所需的坐标、颜色、填充等信息
2. 构造直线类、三角形类等图形类的对象数组来存储每个图形的数据
3. 创建类的静态成员函数来记录图形的个数
4. 构建相应的全局变量来控制画板的大小和位置

## 性能需求

*从性能上，分析要实现的大程序应该满足怎样的性能，包括各个关键算法的运行时间速度、数据量、吞吐量等方面要求。*

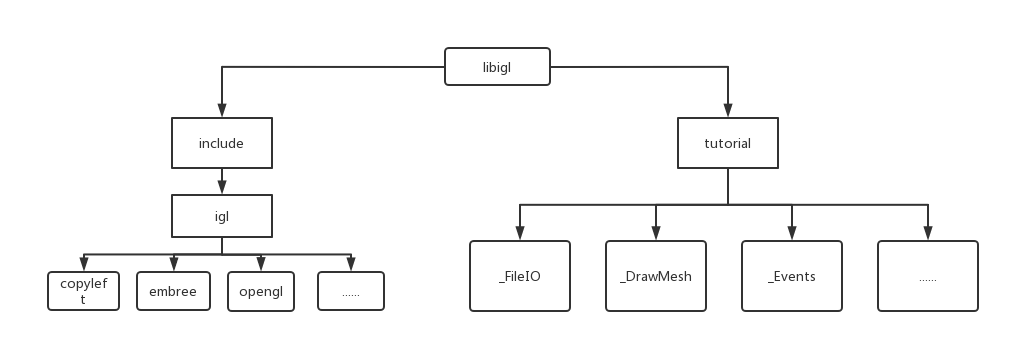
由于该程序是基于OpenGL的，所以把图形绘制在屏幕上需要经常性的调用mydisplay（）函数重新将所有图形绘制一遍，因此mydisplay（）中的算法要尽可能缩短时间复杂度。同时为了便于编程，限定了图形类对象数组的个数，所以运行端要有足够的内存空间来存储数据。

# 类库已有功能分析

*<说明：这部分详细libigl （A simple C++ geometry processing library），包括序总体架构设计、功能模块设计、数据结构设计、文件组织设计>*

## 总体架构设计

*<说明：这里给出一个总体架构图>*



## 类体系设计

Libigl的类体系很复杂，由几十个类组成，主要包括Mesh, MeshViewer, MeshDeformer, MeshUpdater, MeshProcessing, MeshTopology, MeshIO, MeshAnalysis, MeshStatistics, MeshVisualizer, MeshUtils, MeshTransforms和MeshUtilsEngines等。各个类之间存在多层级的关联和继承关系.

## 主要类设计

libigl的主要类设计包括Eigen类、igl::MeshIO类、igl::DrawMesh类、igl::Camera类、igl::Viewer类、igl::OpenGL类、igl::TriMesh类和igl::Scene类。Eigen类提供了基本的矩阵操作功能，igl::MeshIO类为Mesh文件的载入和存储提供接口；igl::DrawMesh类封装了一些绘制Mesh的方法；igl::Camera类提供了摄影机操作的接口；igl::Viewer类封装了OpenGL图形操作；igl::OpenGL类封装了OpenGL的背景和顶点绘制方法；igl::TriMesh类提供了对三角网格的操作和更新接口；igl::Scene类提供了场景内物体的操作和交互接口，包括三角网格和复杂形体。

## 源代码文件组织设计

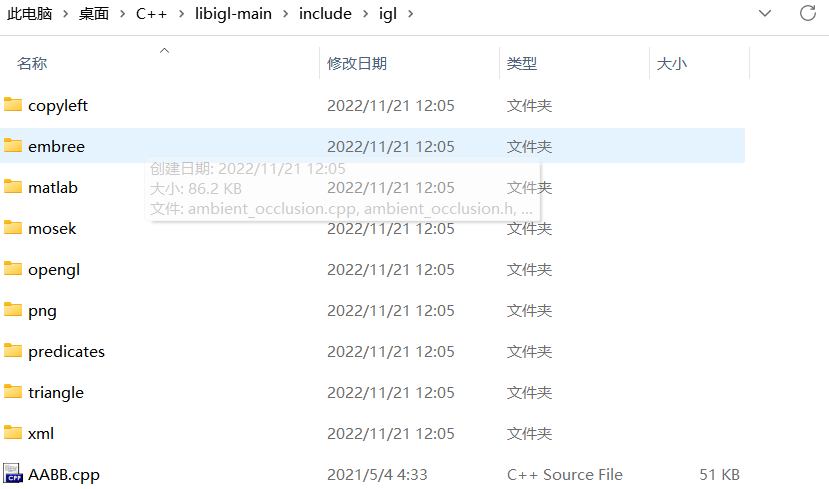
<文件目录结构>

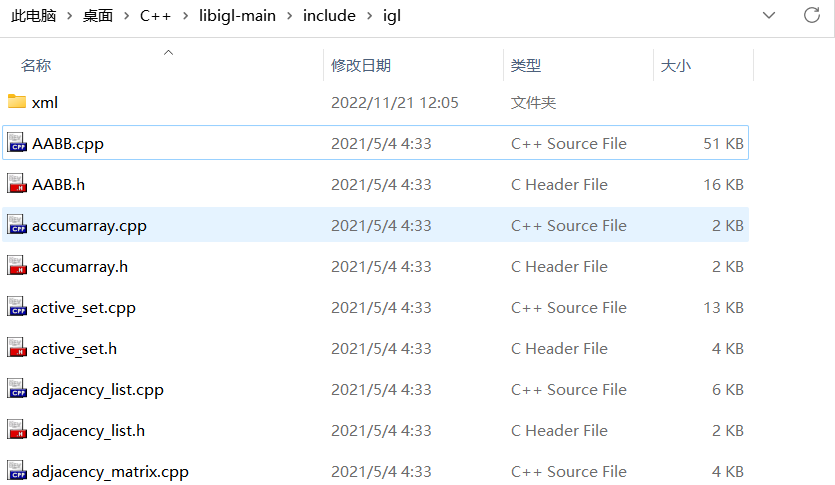
1）文件函数结构

（可通过图或文字描述程序分为几个文件（需写出文件全名，即\*.h，\*.cpp）,每个.cpp文件包含哪些函数定义, 每个.h包含哪些内容。）

主要可以分为两个子文件夹（include和tutorial）。这两个文件夹下包含了libigl最重要的文件，它们的部分截图如下



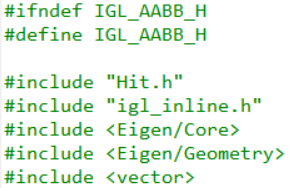


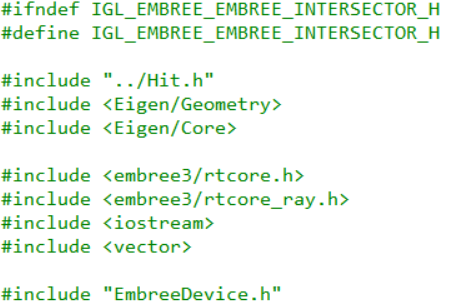


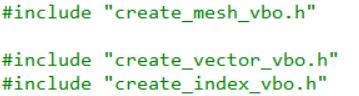
2）多文件构成机制

（说明分文件构成程序的实现机制，即如何具体采用文件包含、#define保护、外部变量或外部函数）

多文件构成机制大部分采用include子文件（.h文件）来实现，下面是部分实例的截图







## 关键功能类及函数设计描述

<说明：每个类按如下形式描述：

函数原型：

功能描述；

参数描述：

返回值描述：

重要局部变量定义：

重要局部变量用途描述：

函数算法描述：

>

igl提供了一组函数，用于开发和管理多边形处理和绘图应用程序，并且还提供了其他图形处理工具。其关键功能分为以下三大类：

1、多边形处理：主要包括自动处理、能编辑、计算和可视化多边形模型。

关键功能：

（一）igl :: readOBJ：可以从OBJ文件中读取三角形网格。

（二）igl :: readSTL：可以从STL文件中读取三角形网格。

（三）igl :: writeOBJ：可以将三角形网格写入OBJ文件。

（四）igl :: writeSTL：可以将三角形网格写入STL文件。

（五）igl :: Triangulate ：可以自动将曲面三角化成三角形网格

（六）igl :: computeEdgeAngles：计算每一边的角度

（七）igl :: computeLaplacian：计算拉普拉斯算子的梯度

（八）igl :: optimizeSharpEdges：有效地减少边角折迭效应

2、绘画：主要是使用opengl功能完成三维绘图，可以在三维空间中调整几何模型。

关键功能：

（一）igl :: Viewer ：创建一个用于在Opengl环境中绘图的Viewer

（二）igl :: Draw :: drawMesh：绘制三角形网格

（三）igl :: Draw :: drawLines：绘制线段

（四）igl :: Draw :: drawCurves：绘制曲线

（五）igl :: Draw :: drawPoints：绘制点

（六）igl :: Draw :: drawLabels：绘制标签

3、自动化：解决几何绘图领域的一些关键挑战，将自动处理算法应用到几何模型上。

关键功能：

（一）igl :: Remesh :: remesh：重新建模

其中一些重要函数的描述如下：

igl::point\_mesh\_squared\_distance

**函数原型：**

float igl::point\_mesh\_squared\_distance(const Eigen::MatrixXd &P, const Eigen::MatrixXd &V, const Eigen::MatrixXi &F, const int &cell, const Eigen::Vector3d &p, Eigen::Vector3d &q, Eigen::RowVector3i &f)

**参数描述：**

P：n×3的矩阵，表示点的位置 V：m×3的矩阵，表示模型顶点的位置 F：l×3的矩阵，表示模型的面 cell：要搜索的细胞索引 p：模型外部点的位置，3维向量 q：用于存储模型上最近点的结果 f：用于存储最近点所在面的索引

**功能描述：**

P：n×3的矩阵，表示点的位置 V：m×3的矩阵，表示模型顶点的位置 F：l×3的矩阵，表示模型的面 cell：要搜索的细胞索引 p：模型外部点的位置，3维向量 q：用于存储模型上最近点的结果 f：用于存储最近点所在面的索引

igl::opengl::glfw::Viewer::append\_mesh

**函数原型：**

void append\_mesh(const Eigen::MatrixXd &V, const Eigen::MatrixXi &F, bool rebuild\_cache = true);

**参数描述：**

V是顶点坐标矩阵，F是三角形面片数组  
**功能描述：**

将网格模型追加到Viewer实例中，如果参数rebuild\_cache为false则不重建缓存。

igl::readPLY

**函数原型：**

bool readPLY(const std::string &ply\_file, Eigen::MatrixXd &V, Eigen::MatrixXi &F, Eigen::MatrixXd &TC, Eigen::MatrixXi & FTC, Eigen::MatrixXd &N, Eigen::MatrixXi & FN);

**参数描述：**

ply\_file为待读取的ply文件的路径，V为顶点坐标矩阵，F为三角形面片数组，TC为顶点uv坐标矩阵，FTC为三角形uv面片数组，N为法向量矩阵，FN为每个面片的法向量索引数组

**功能描述：**

从指定的ply文件中读取网格模型，包括顶点、面片、uv坐标、法线信息。

igl::writePLY

**函数原型：**

bool writePLY(const std::string &ply\_file, const Eigen::MatrixXd &V, const Eigen::MatrixXi &F, const bool ascii = false);

**参数描述：**

ply\_file为待写入的ply文件的路径，V为顶点坐标矩阵，F为三角形面片数组，ascii表示文件格式是否为ascii

**功能描述：**

将指定的网格模型保存到ply文件中，可以指定文件格式为ascii或binary。

igl::readOBJ

**函数原型：**

igl::readOBJ(const std::string & obj\_file\_name, Eigen::MatrixXd & V, Eigen::MatrixXi & F, igl::readOBJParams & params):  
**参数描述：**

obj\_file\_name表示OBJ文件地址，V为输出顶点矩阵，F为输出面片索引矩阵，params为可选参数

**功能描述：**

读取OBJ文件，解析并存储其中的顶点信息与面片索引

igl::boundary\_facets

**函数原型：**

igl::boundary\_facets(const Eigen::MatrixXi & F, Eigen::VectorXi & b):  
**参数描述：**

F表示面片索引矩阵，b为输出边界面片索引矩阵

**功能描述：**

获取网格模型边界中的面片索引

igl::slice\_mask

**函数原型：**

igl::slice\_mask(const Eigen::MatrixXd & X, const Eigen::MatrixXi & R, const Eigen::MatrixXi & F, const Eigen::MatrixXi & C, Eigen::MatrixXd & U):  
**参数描述：**

X表示模型的顶点矩阵，R表示切片索引矩阵，F表示模型的面片索引矩阵，C表示掩码数据，U为输出数据

**功能描述：**

根据指定的切片索引、面片索引和掩码数据，提取模型的顶点信息

igl::MeshBooleanType

**函数原型：**

igl::MeshBooleanType(const Eigen::MatrixXd & Va, const Eigen::MatrixXi & Fa, const Eigen::MatrixXd & Vb, const Eigen::MatrixXi & Fb, igl::MeshBooleanType & type):  
**参数描述：**

Va表示模型A的顶点矩阵， Fa表示模型A的面片索引矩阵，Vb表示模型B的顶点矩阵，Fb表示模型B的面片索引矩阵，type为输出的枚举类型。

**功能描述：**

获取两个模型的boolean运算类型

igl::barycenter

**函数原型：**

igl::barycenter(const Eigen::MatrixXd & V, const Eigen::MatrixXi & F, Eigen::MatrixXd & bc):  
**参数描述：**

V表示模型的顶点矩阵，F表示模型的面片索引矩阵，bc为输出重心矩阵

**功能描述：**

计算模型各面片的重心

igl::writeSTL

**函数原型：**

igl::writeSTL(const std::string & stl\_file\_name, const Eigen::MatrixXd & V, const Eigen::MatrixXi & F, const bool binary = true):

**参数描述：**

stl\_file\_name表示输出文件地址，V表示模型的顶点矩阵，F表示模型的面片索引矩阵，binary为可选参数，表示是否采用二字节表示

**功能描述：**

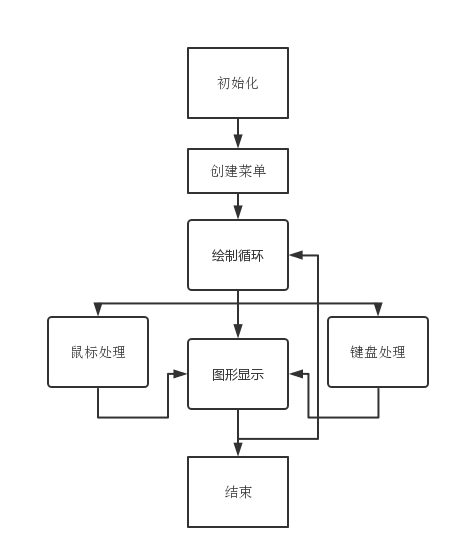
将模型信息保存为STL文件

# 新设计类功能说明

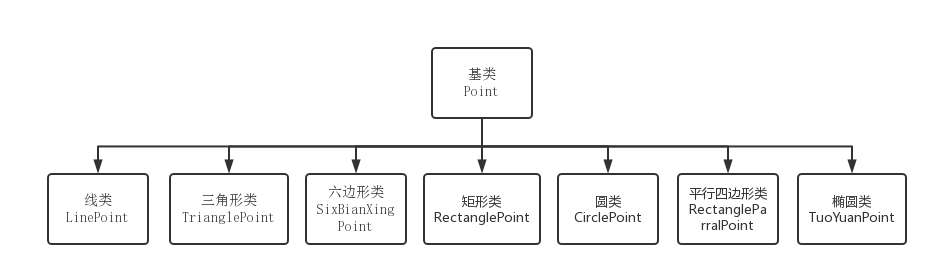
*<说明：这部分详细介绍大程序开发设计有关内容，包括大程序总体架构设计、功能模块设计、数据结构设计、文件组织设计>*

## 总体架构设计

*<说明：这里给出一个总体架构图>*

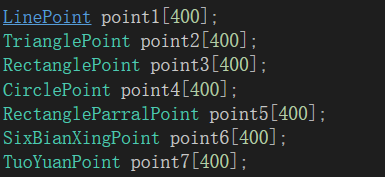


## 类模块体系设计



七大图形子类继承至基类Point，用来存储各种图形的位置、颜色、填充、画笔等信息。

## 数据结构类设计



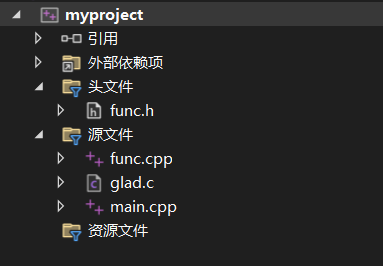
图形类的对象数组：用来存储图形对象的数据

## 源代码文件组织设计

<文件目录结构>

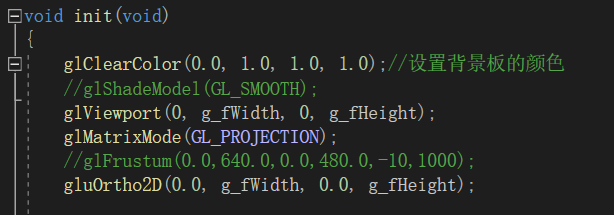
1）文件函数结构

（可通过图或文字描述程序分为几个文件（需写出文件全名，即\*.h，\*.cpp）,每个.cpp文件包含哪些函数定义, 每个.h包含哪些内容。）

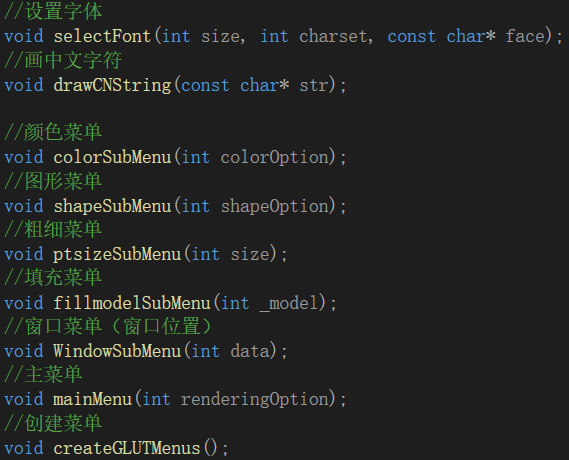


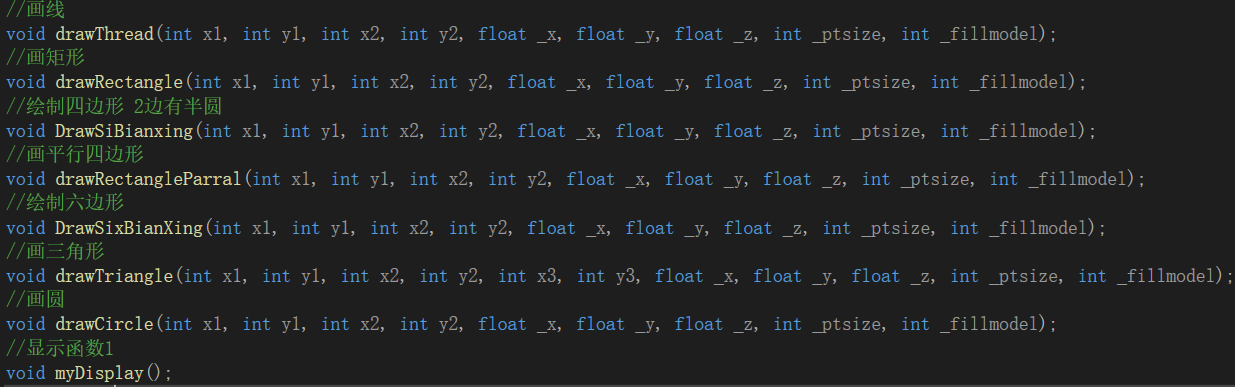
程序主要分为三个文件：main.cpp func.cpp func.h

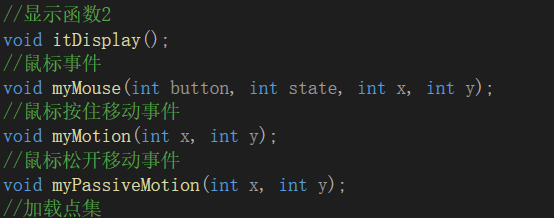
Main.cpp:



Func.h:

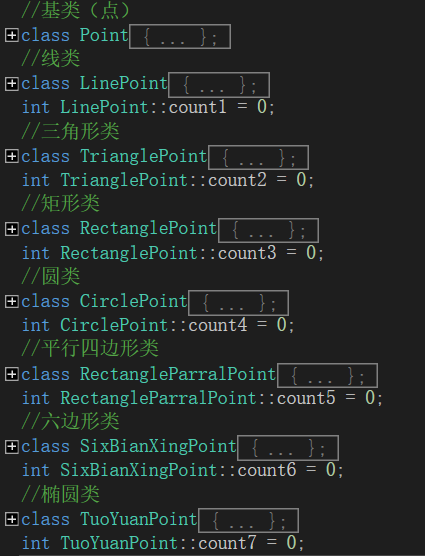


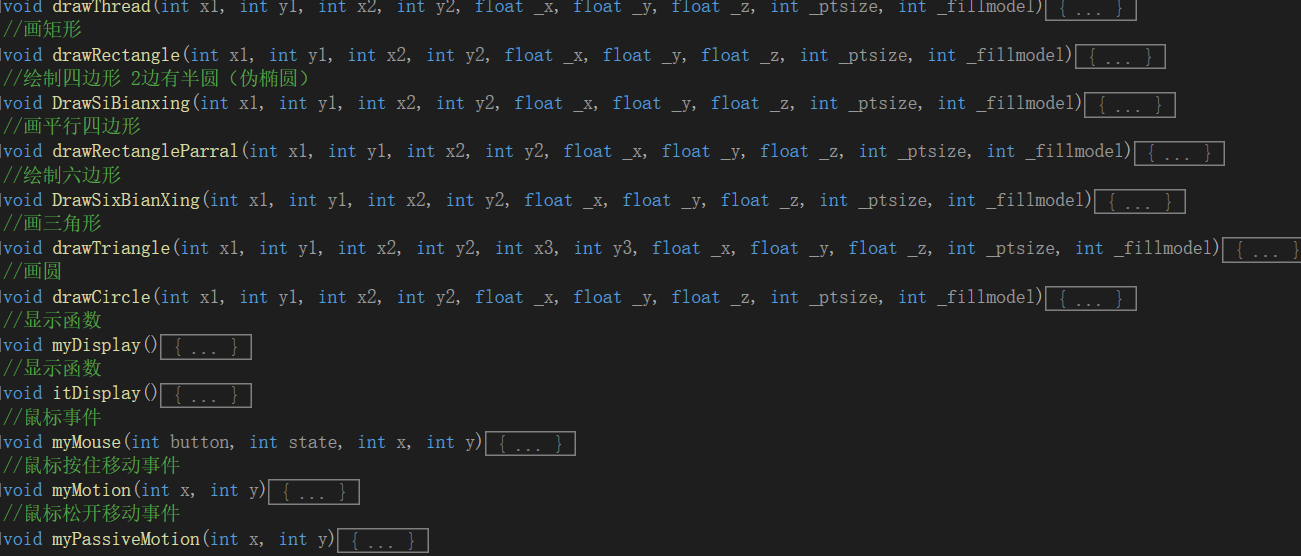
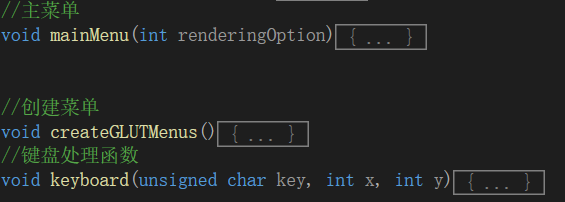
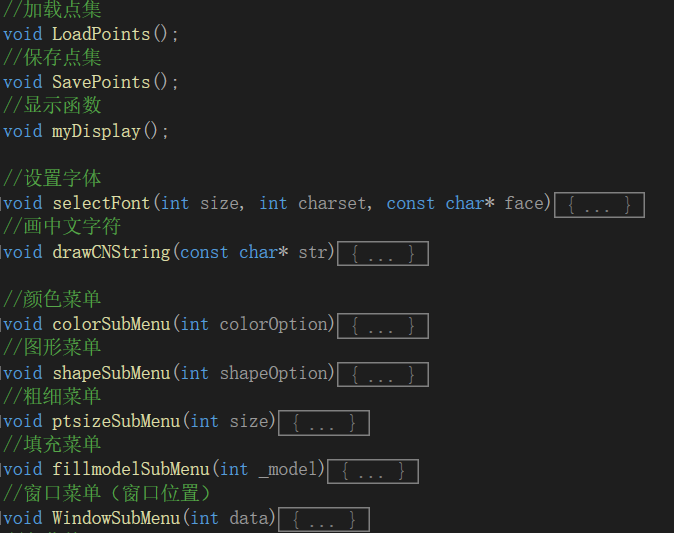




Func.cpp:

（基类Point和七大图形子类）



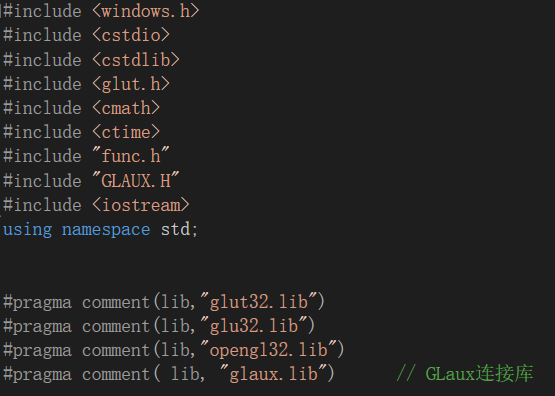


2）多文件构成机制

（说明分文件构成程序的实现机制，即如何具体采用文件包含、#define保护、外部变量或外部函数）

主要通过main.cpp include “func.h”以及OpenGL的其他库来实现多文件的构成，具体形式如下：

Main.cpp头部：



## 重点类及函数设计描述

<说明：每个函数按如下形式描述：

函数原型：

功能描述；

参数描述：

返回值描述：

重要局部变量定义：

重要局部变量用途描述：

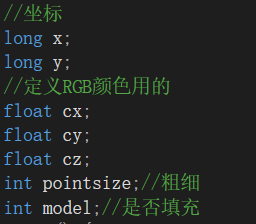
函数算法描述：

>

**重点类：**

该程序共设计了一个基类和七个子类（详见上面的贴图-->4.4中的func.cpp），其大致用途是用来存储各图形的位置、颜色、填充等信息。

基类Point中重要的成员变量：



**重点函数：**



功能：开始绘制前的初始化工作：如设置背景板的颜色、图像的视口和视角（OpenGL相关内容）等



参数：colorOption用来指定画笔的颜色

功能：指定画笔的颜色



参数：shapeOption用来指定将要绘制图形

功能：指定将要绘制的图形



参数：size用来指定画笔的粗细

功能：指定画笔的粗细



参数：\_model用来指定（填充/线框）

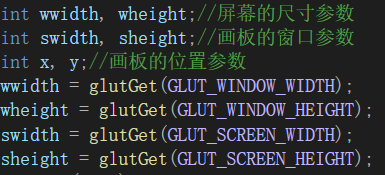
功能：指定画图的模式



参数：data用来判断哪个菜单被选中

重要局部变量：

（相关用途见下图的注释）



功能：实现窗口（画板）的位置改变



参数：renderingOption用来判断主菜单中的哪个选项被按下

功能：对主菜单中的每个功能按键进行处理，让其实现对应的功能



功能：创建菜单



参数：key:判断哪个键盘按键被按下 x,y:键盘按键触发时鼠标的位置

功能：处理键盘事件



参数：分别用来指定起点、终点的位置信息、颜色（RGB的值）、画笔的粗细和填充模式

功能：画直线



参数：分别用来指定对角顶点的位置信息、颜色（RGB的值）、画笔的粗细和填充模式

功能：画矩形



参数：分别用来指定对角顶点（矩形逼近的对角顶点）的位置信息、颜色（RGB的值）、画笔的粗细和填充模式

功能：画椭圆



参数：分别用来指定对角顶点的位置信息、颜色（RGB的值）、画笔的粗细和填充模式

功能：画平行四边形



参数：分别用来指定对角顶点（六边形的左上和右下两个对称点）的位置信息、颜色（RGB的值）、画笔的粗细和填充模式

功能：画六边形

算法描述：通过将六边形对称分为六个部分来画，每个部分对应于正三角形的底边，由此调用画直线的函数（进行六次）即可得到正六边形



参数：分别用来指定三个顶点的位置信息、颜色（RGB的值）、画笔的粗细和填充模式

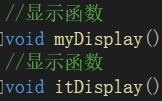
功能：画三角形



参数：分别用来指定椭圆直径对应两点的位置信息、颜色（RGB的值）、画笔的粗细和填充模式

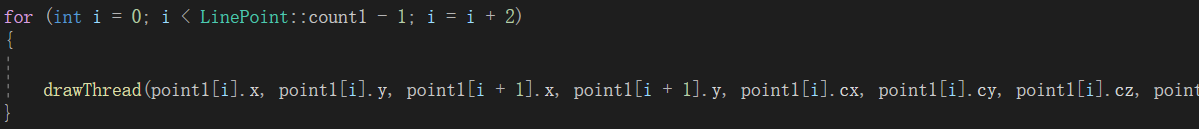
功能：画圆形

算法描述：通过微小角度连续画闭合直线来逼近得到近似圆形



功能：将所有图形对象绘制在画板上

算法描述：以画直线为例，通过一个for循环语句将所有记录在内的直线类对象的数据全部通过调用画直线函数呈现在画板上。





参数：button:判断鼠标的哪个按键被按下（左键/中键/右键） state:判断鼠标处于按下（DOWN）还是弹起（UP）状态 x,y:传入鼠标的位置信息

功能：处理鼠标事件，根据左键还是右键、弹起还是按下来记录对应图形对象的相关数据



参数：x,y:传入鼠标的位置信息

功能：动态记录图形对象终点数据、实时绘制相应的图形



参数：x,y:传入鼠标的位置信息

功能：上一个函数输入的是图形对象起点的相关数据，这个函数记录的是图形对象终点的相关数据（若是三角形则需记录两个终点数据）



功能：将特定文件夹（myproject或Release）下的文本文件“1.txt”中存储的图形数据赋值给当前的图形对象数组（之后调用显示函数即可将相应的图形绘制在画板上），相当于文件的打开功能



功能：将当前绘制图形对象数组中的全部数据保存到特定文件夹（myproject或Release）下的文本文件“1.txt”中，相当于文件的保存功能

# 部署运行和使用说明

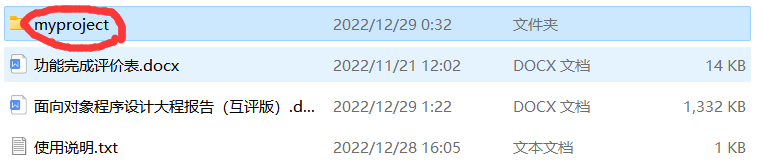
*<此部分介绍如何由提交的源代码包，进行存放、编译生成.exe文件的过程说明，以及运行.exe后的用户使用手册>。*

## 编译安装

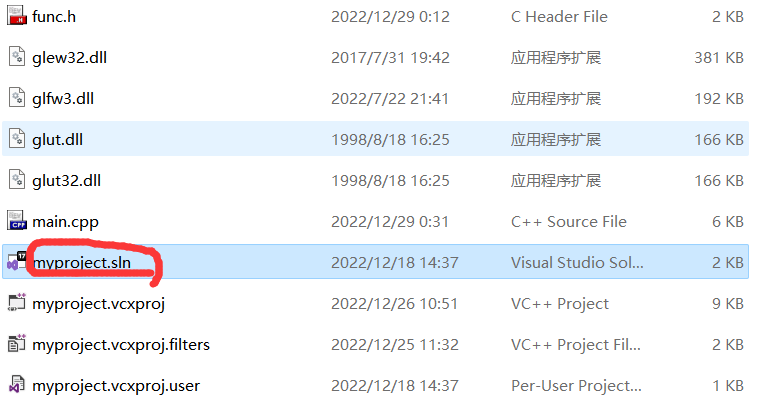
*<说明：如何从源代码开始编译生成可执行文件，如何运行的说明>*

*为了保证能够编译成功，需要准备Visual Studio，我电脑上的版本是VS2022*

首先，打开压缩包后将看到如下画面：



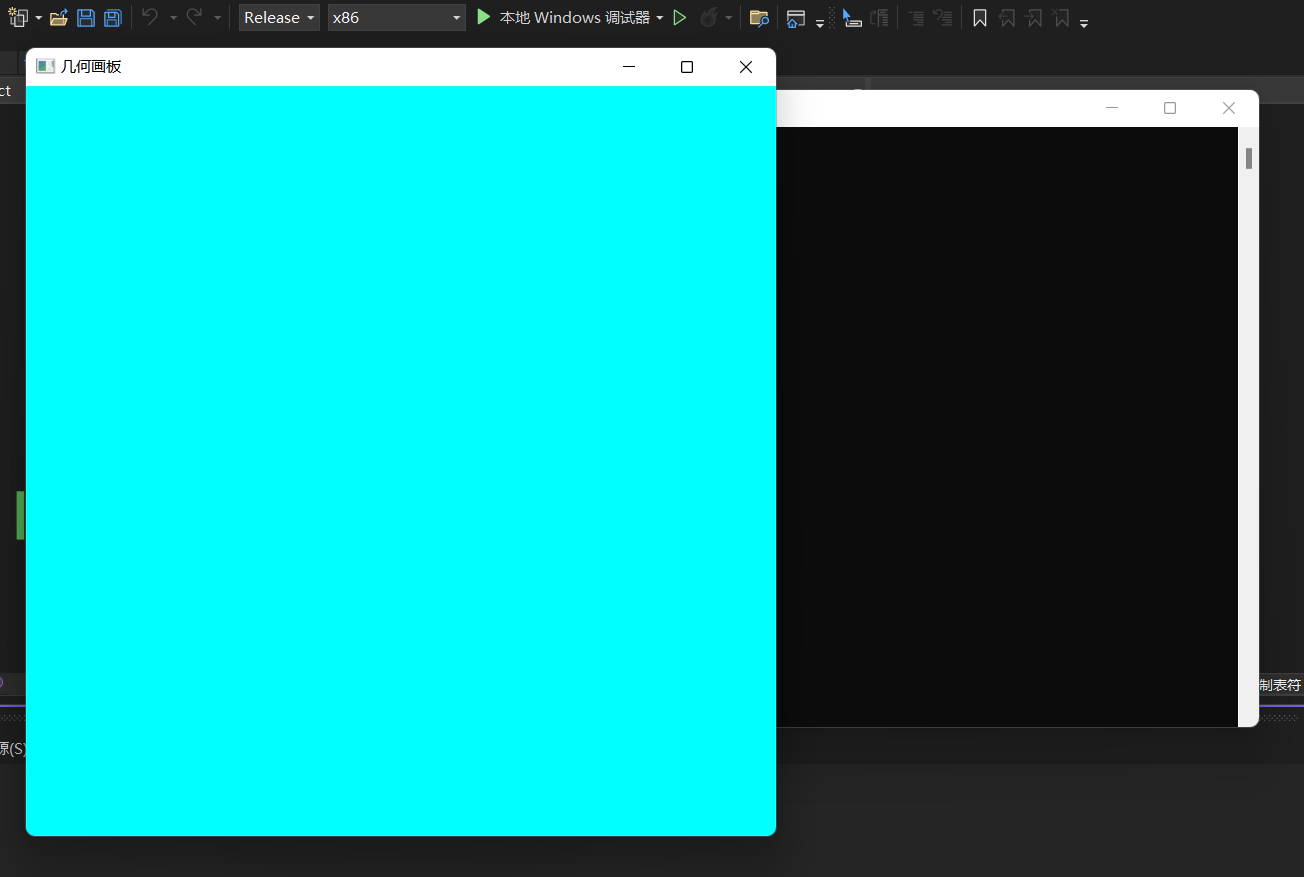
点击myproject-->在下一文件夹中找到myproject.sln，点击打开：



打开后将看到如下画面（VS2022截图）：

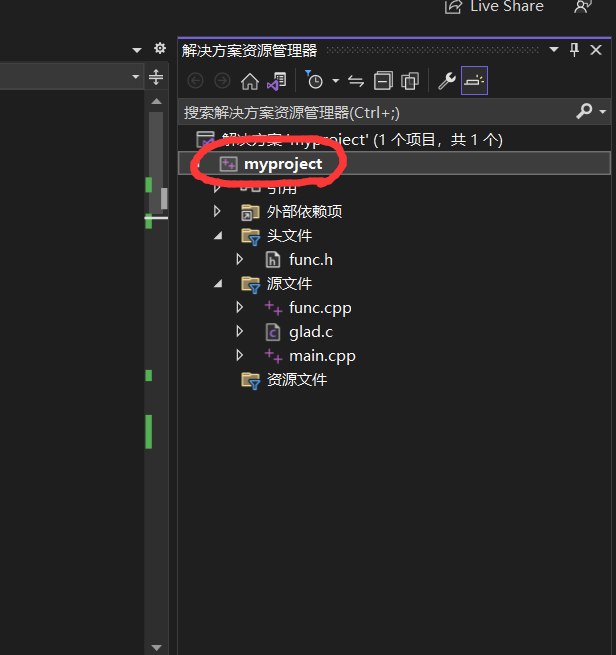


之后点击绿色三角形即可编译成功并开始使用，成功后将看到如下画面：

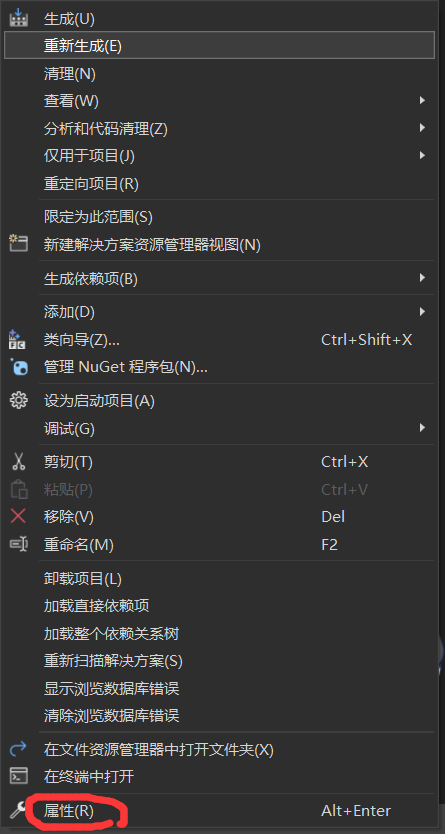


之后在画板中点击鼠标右键即可查看菜单，选择相应的功能。

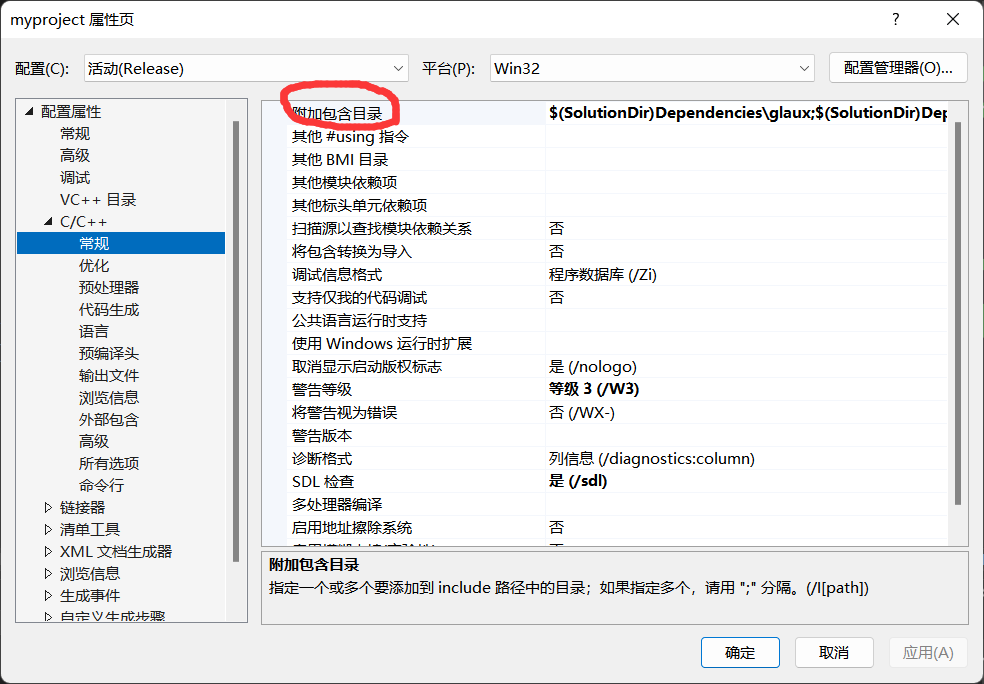
*如果编译失败，请查看如下这三个地方的路径是否与你当前文件夹（解压缩后的文件夹）的路径相一致：*

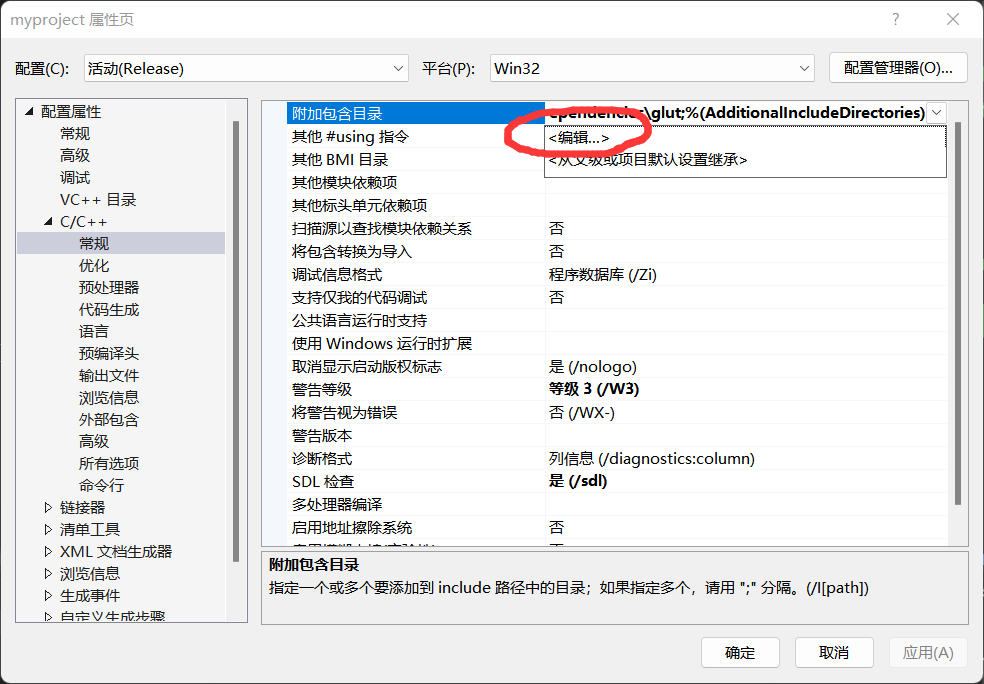


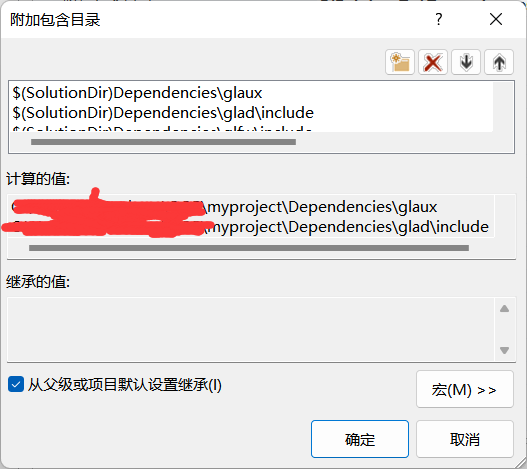
在VS右部找到myproject，右键出现菜单，点击属性



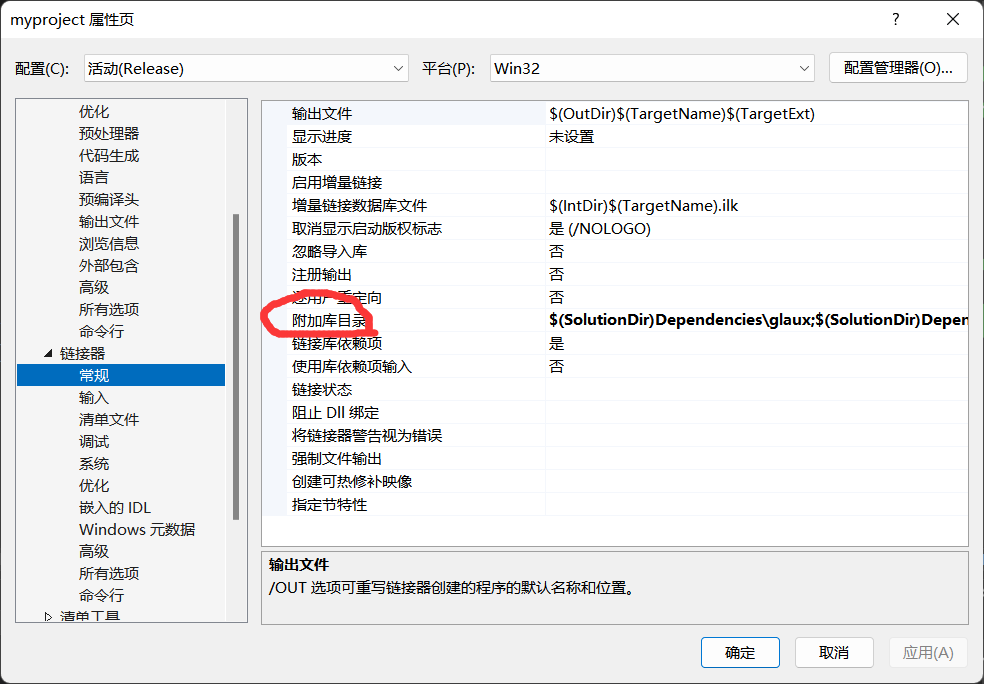
先找到C/C++ 常规选项下的附加库目录，查看其对应路径是否正确

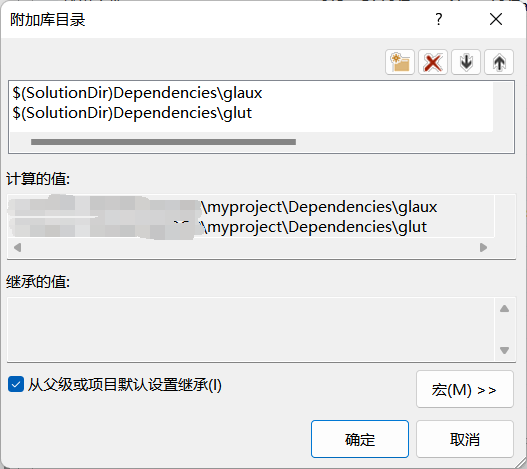
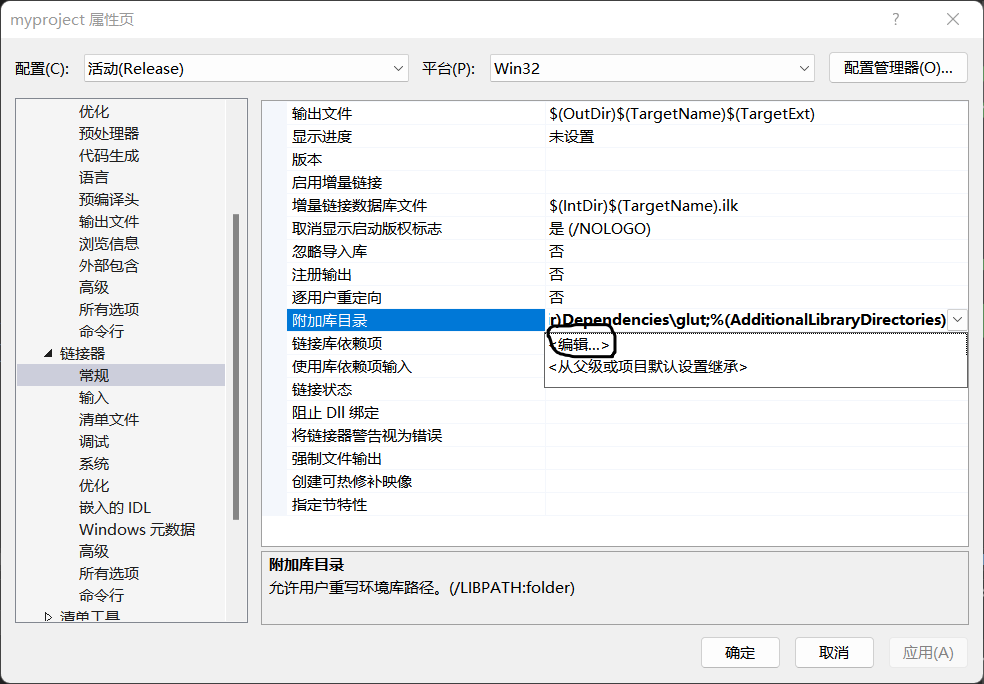






然后找到链接器 常规选项下的附加库目录相应的路径是否正确





然后找到链接器 输入选项下的附加依赖项，确保其中填写了如下内容：

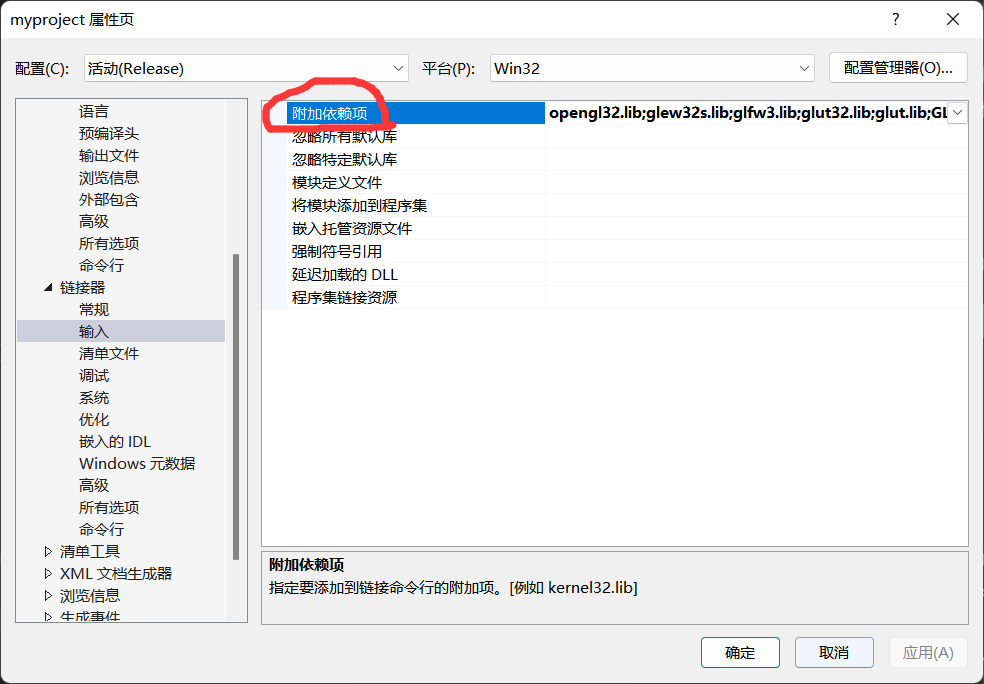
opengl32.lib

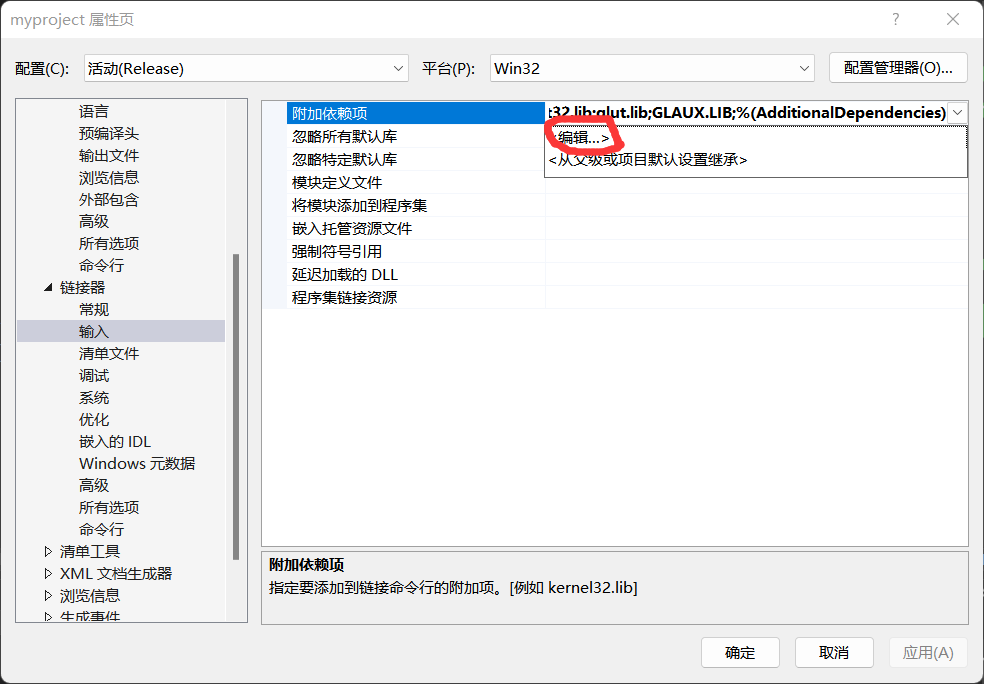
glut32.lib

glut.lib

GLAUX.LIB

（每个一行）



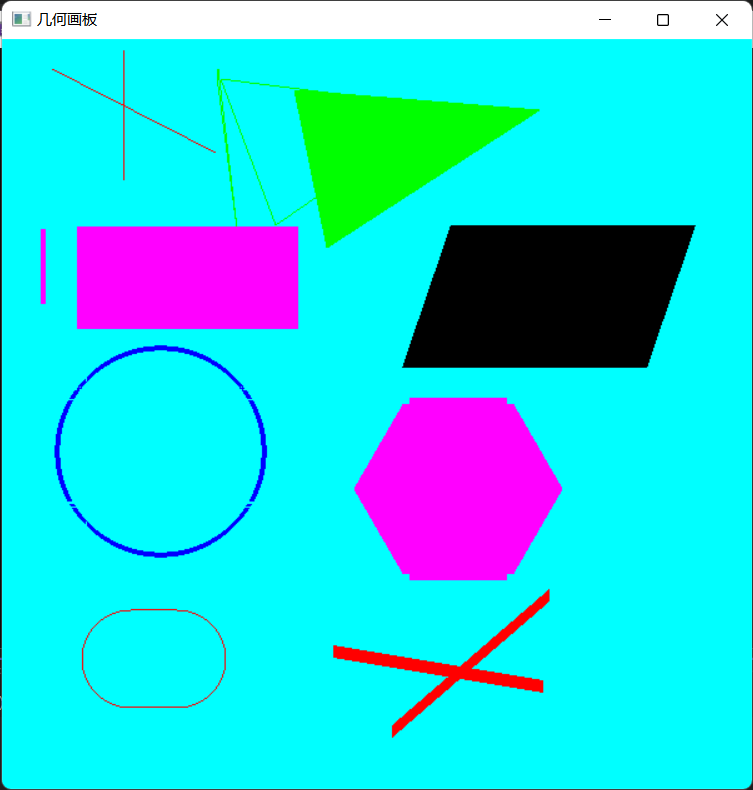


## 运行测试

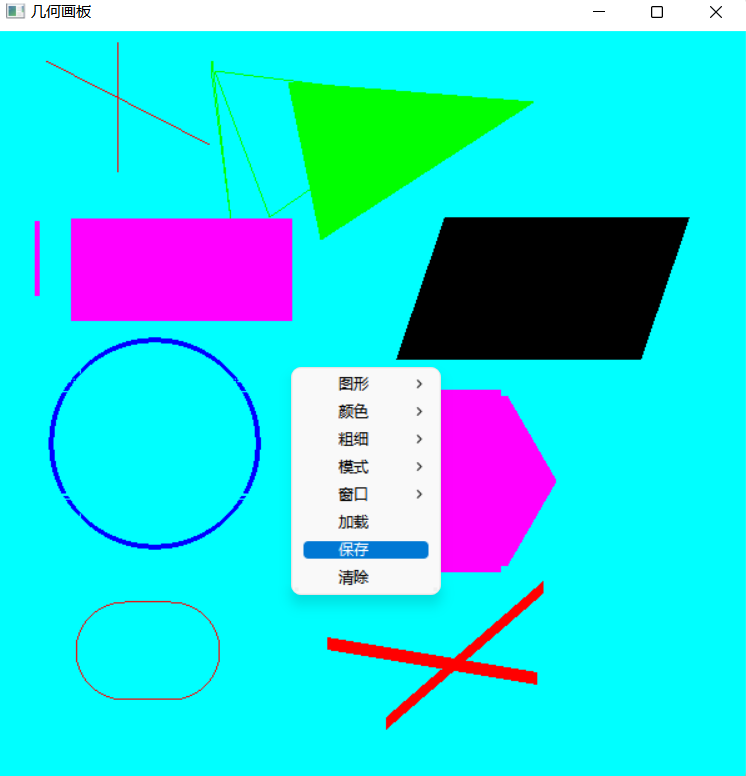
*<选取测试阶段典型的案例，说明如何设计测试数据，发现和定位错误的，测试结果可以含有屏幕截图。>*

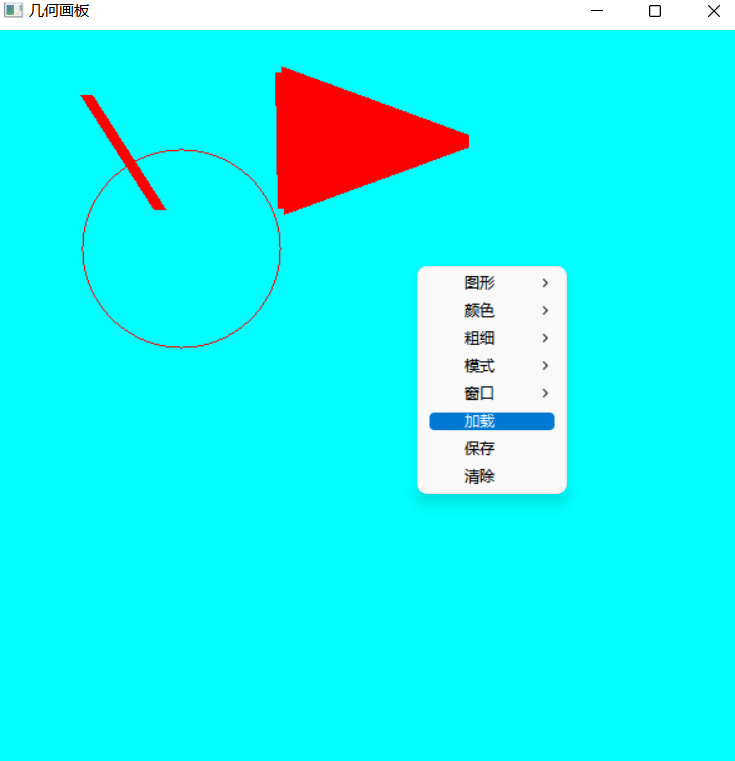
**测试案例——检测加载、保存功能的实现：**

首先，打开程序在画板上画下一些图形，尽可能应用全部的功能，画下的图形如下：

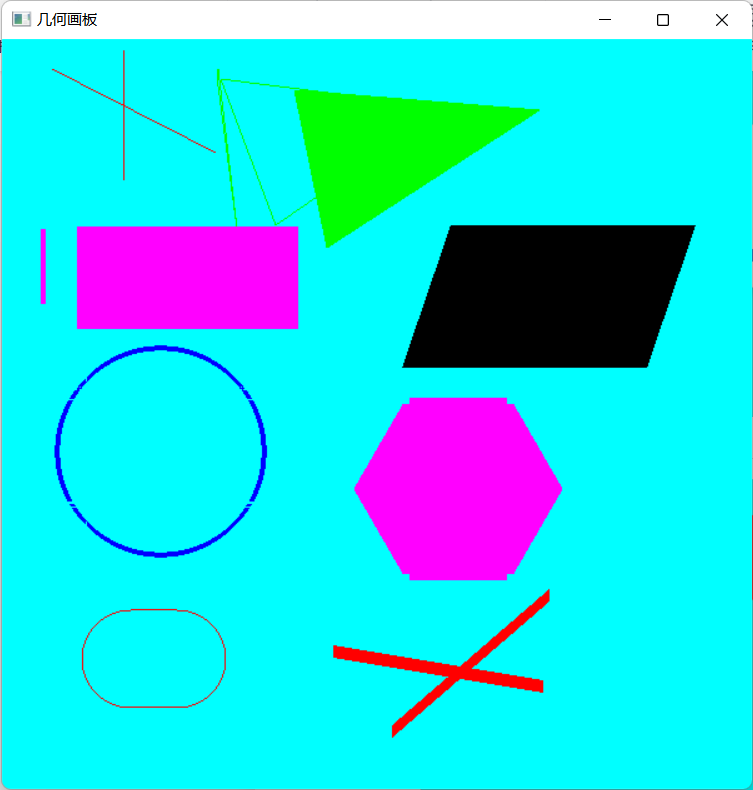


然后，点击鼠标右键在出现的菜单中选择保存功能，保存完好点击清除，并重新画上一些图形。

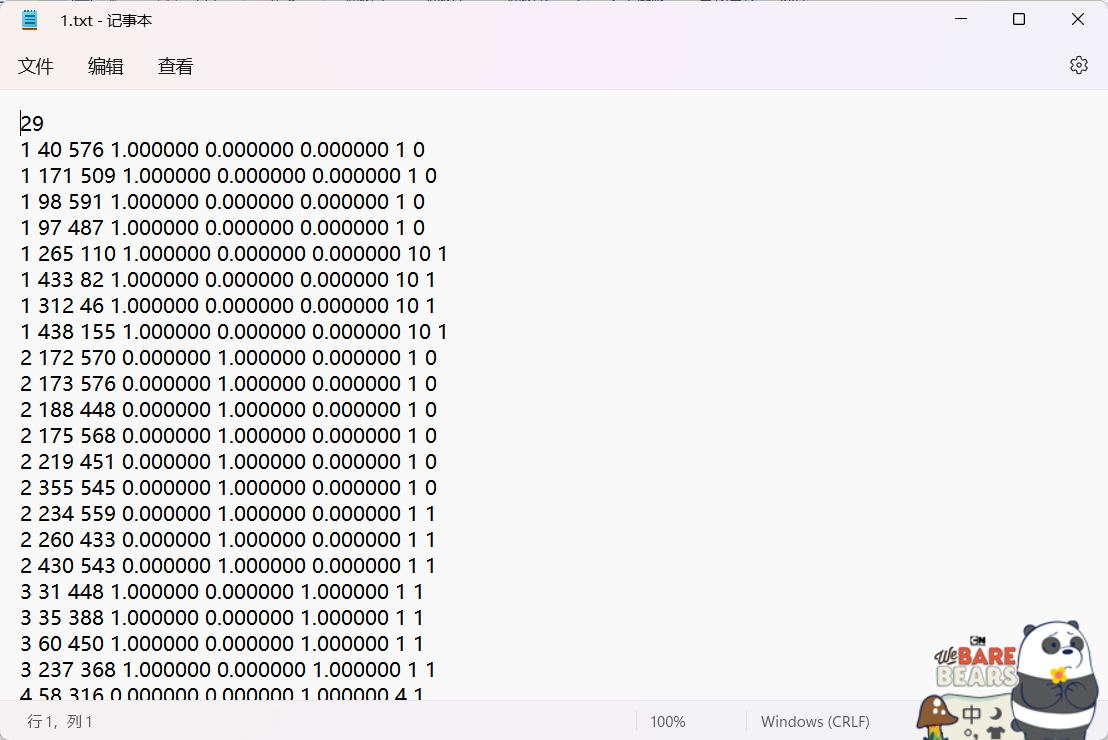




点击加载功能后屏幕上重新出现了上一次画的图案：



然后再去myproject或Release文件夹下找到存储图形数据的“1.txt”，检验其中记录的数据：



检验无误，说明加载、保存功能测试成功！

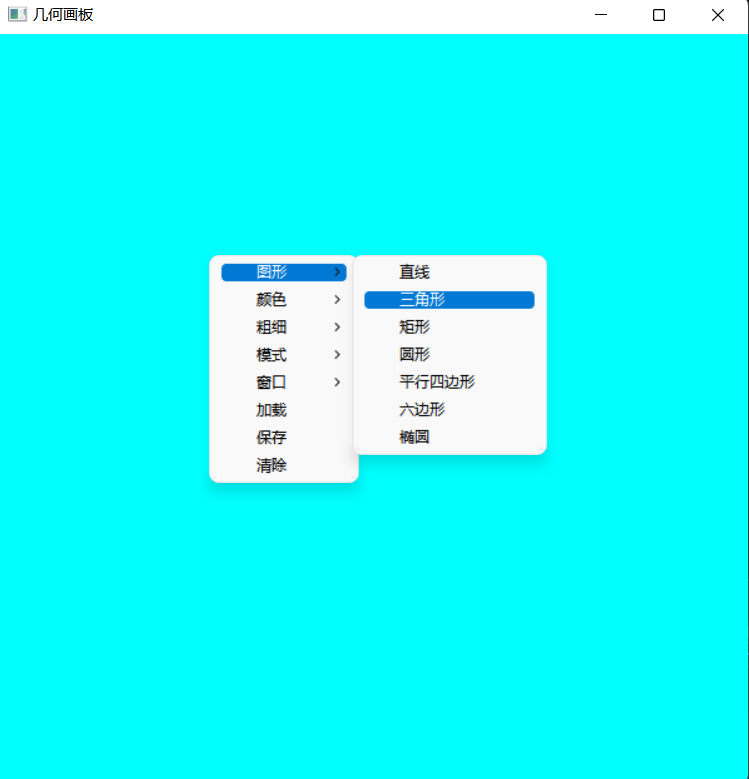
## 使用操作

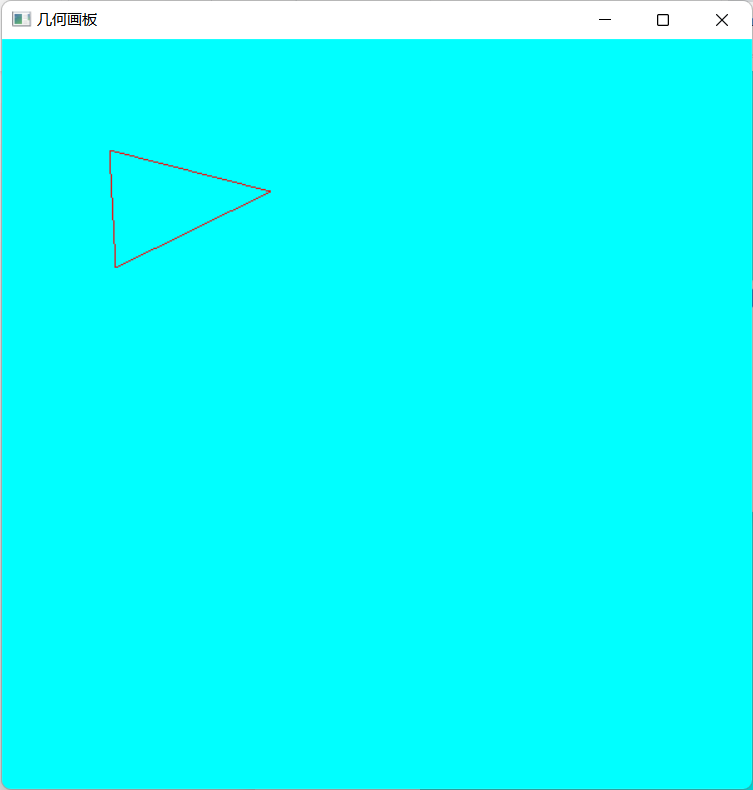
*<此部分介绍如何使用操作大程序，包括各个功能如何操作，结合一些截图>。*

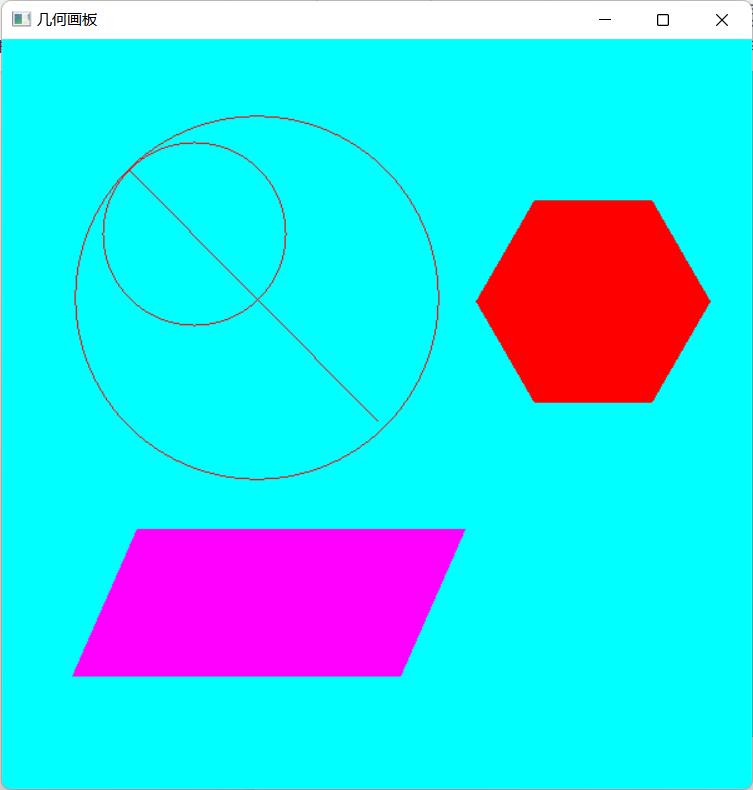
菜单：点击鼠标右键弹出菜单，选择相应的功能之后用鼠标左键即可进行绘图。

保存、加载：保存功能可以将上次绘制的图形数据存储到myproject或Release文件夹下的“1.txt”文件里（哪个文件夹取决于如何生成该画图程序，通过编译生成—>myproject文件夹下，直接点击exe执行文件生成—>Release文件夹下）；加载功能可以将“1.txt”文件里的图形数据画到几何面板上

清除：点击清除功能后有时需要在点击一下屏幕才能实现全屏清除







## 收获感言

*<说明：在项目过程中的体会和心得，经验和教训，自我评价。*

这次大作业最大的收获是学会了OpenGL绘图的一些方法，也自己去网上摸索了一些额外功能的开发（如交互式菜单如何实现等）。但毕竟OpenGL发展时间已久，其中涉及的函数和知识繁多，所以在自己完成项目的时候常常会出现达不到预期结果的情况，然后就需要去网上找经验贴，学习借鉴别人的代码来理清思路。这其中就会面临取舍，有时候一些代码块可能前期需要，但在项目慢慢推进的过程中它们往往失去了用处，这个时候就要及时的吸收新思路、摒弃旧思路，将原有的框架和新的API整合，来实现新的效果。这些经历都锻炼了我应用开发的能力，虽然过程很煎熬、也碰了不少壁，但收获也不小。总的来说，我自认为自己的程序写的还算满意，虽然由于时间和精力的问题，我实现的程序可能过于简单，但大部分还是满足了作业的要求，希望自己以后能够再接再厉。

# 参考文献资料

*<列出参考的书籍、论文、网站的信息和地址等>。*

**关于OpenGl的配置问题：**

CSDN “openGL 在Visual Studio2022 环境下的配置”

<https://blog.csdn.net/Accelerator12138/article/details/126858648>

**OpenGl的学习与使用：**

<https://learnopengl-cn.github.io/>

CSDN （五）OpenGL菜单管理\_遥望星河的博客-CSDN博客\_opengl菜单 https://blog.csdn.net/weixin\_44326002/article/details/99329837?ops\_request\_misc=%257B%2522request%255Fid%2522%253A%2522167188212216782425181092%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request\_id=167188212216782425181092&biz\_id=0&utm\_medium=distribute.pc\_search\_result.none-task-blog-2~all~top\_positive~default-1-99329837-null-null.142^v68^control,201^v4^add\_ask,213^v2^t3\_control2&utm\_term=opengl%E8%8F%9C%E5%8D%95%E5%8A%9F%E8%83%BD&spm=1018.2226.3001.4187

CSDN gluOrtho2D作用

<https://blog.csdn.net/u012861978/article/details/86737990>

CSDN OpenGL常用函数\_weixin\_30469895的博客-CSDN博客

https://blog.csdn.net/weixin\_30469895/article/details/987...=1018.2226.3001.4187

CSDN OpenGL 满屏的两种方式：借助glutFullScreen ，借助glutEnterGameMo

[https://blog.csdn.net/deyangliu/article/details/5947415](https://blog.csdn.net/deyangliu/article/details/5947415" \t "_blank)

CSDN OpenGL的键盘交互绘制+（详细代码）\_BBbila的博客-CSDN博客\_opengl添加菜单键盘

https://blog.csdn.net/qq\_40610760/article/details/8006422...=1018.2226.3001.4187

CSDN 计算机图形学OpenGL学习实验三——交互操作\_临风浅吟的博客-CSDN博客

https://blog.csdn.net/qq\_37996608/article/details/105720878?spm=1001.2101.3001.6650.5&utm\_medium=distribute.pc\_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7ERate-5-105720878-blog-84188837.pc\_relevant\_3mothn\_strategy\_recovery&depth\_1-utm\_source=distribute.pc\_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7ERate-5-105720878-blog-84188837.pc\_relevant\_3mothn\_strategy\_recovery&utm\_relevant\_index=10

CSDN 实验三 实验四：OpenGL的交互绘制\_m0\_46336221的博客-CSDN博客\_opengl在窗

https://blog.csdn.net/m0\_46336221/article/details/125201077?ops\_request\_misc=%257B%2522request%255Fid%2522%253A%2522167187330716800225575332%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request\_id=167187330716800225575332&biz\_id=0&utm\_medium=distribute.pc\_search\_result.none-task-blog-2~all~baidu\_landing\_v2~default-4-125201077-null-null.142^v68^control,201^v4^add\_ask,213^v2^t3\_control2&utm\_term=opengl%20%E7%BB%98%E5%88%B6%E6%8C%89%E9%92%AE&spm=1018.2226.3001.4187

CSDN网站等其他文章：

https://blog.csdn.net