计算机图形学作业一

【专业】计算机科学与技术　　【姓名】谢宇桐　　【学号】22336259

一、环境搭建

由于我的疏漏没看到实验报告要包含环境搭建这一项，因此我没有中间的过程的截图，这里大部分只有文字叙述：

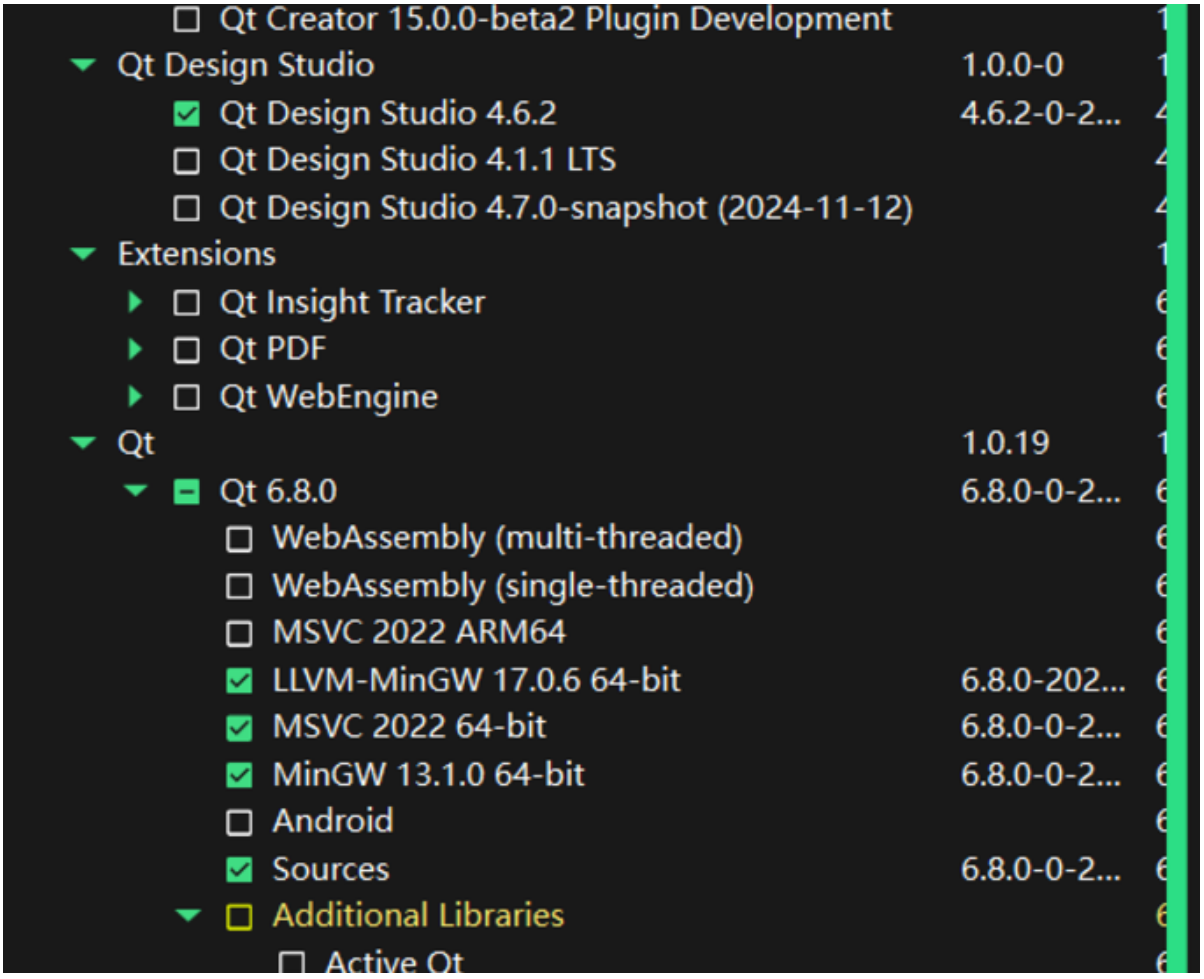
1. ### Visual Studio 2022 安装

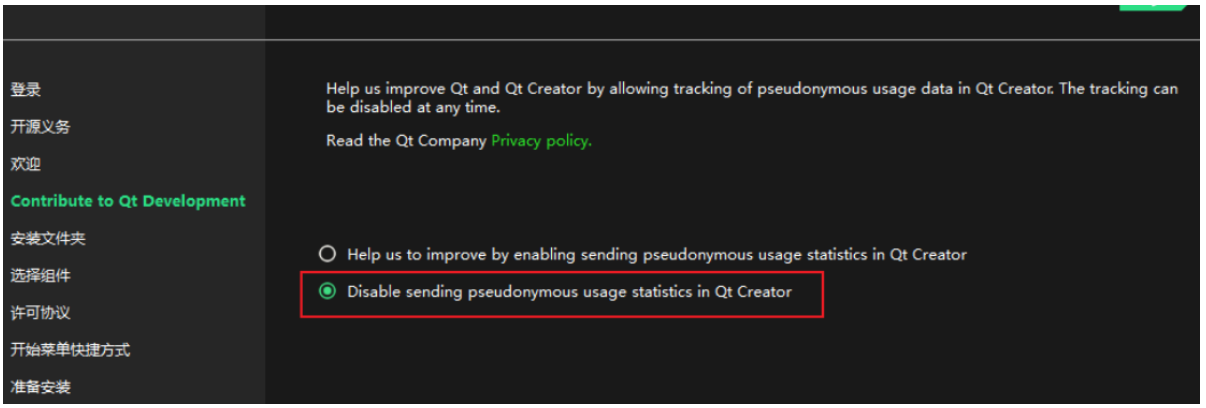
   按照教程网址 https://visualstudio.microsoft.com/zh-hans/vs/ 安装Visual Studio 2022 Community 版，默认配置就好。

2. ### Qt安装

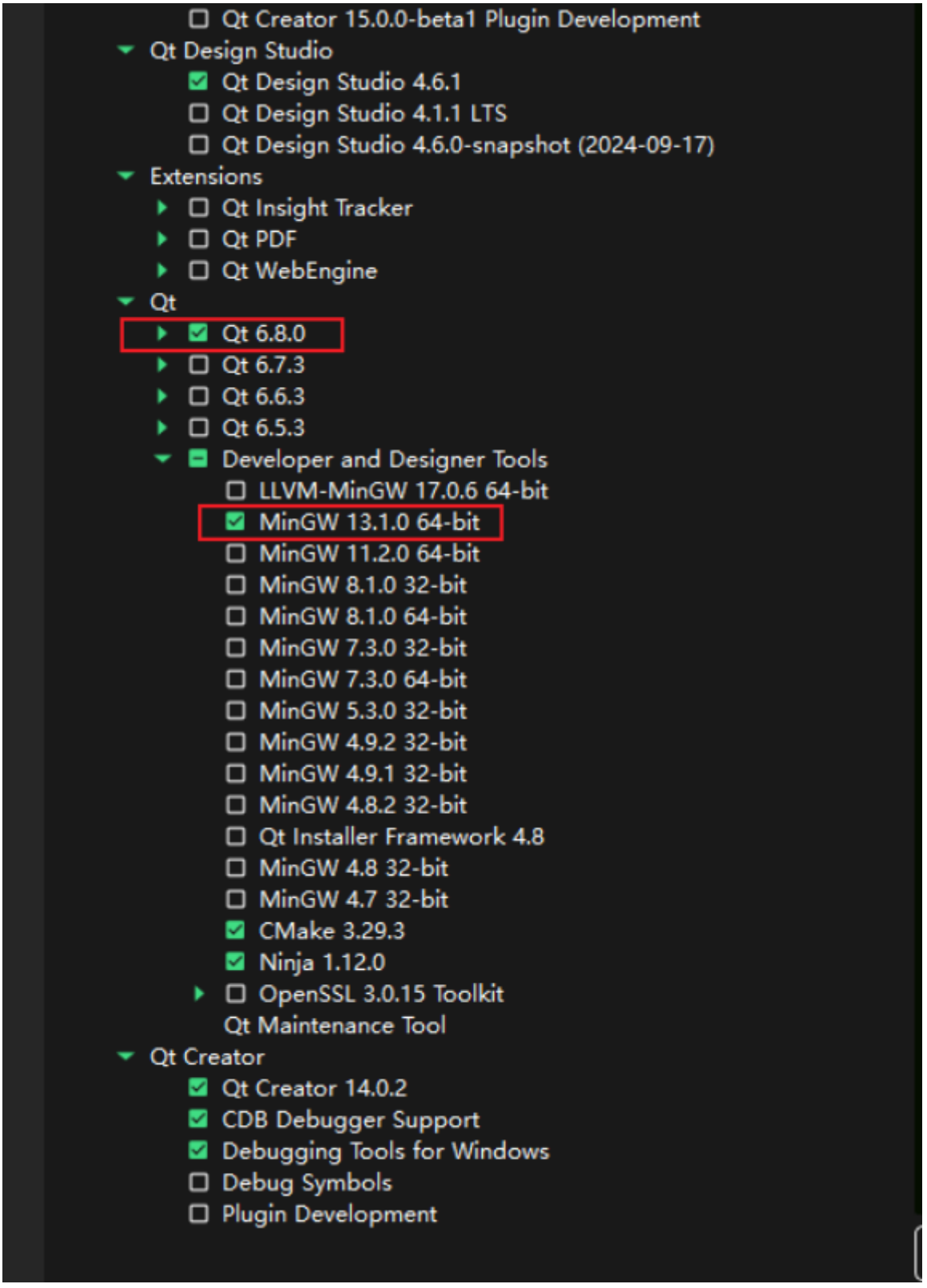
   在这里我遇到了很多问题，首先安装Qt找不到合适的源头进行安装，在清华镜像源找时找不到最新版本的Qt，上网查才得知Qt 5.15 后不提供离线安装包，需要付费才能安装。助教发的网址已失效，为了效率只能CSDN查询到清华镜像源的位置 https://mirrors.tuna.tsinghua.edu.cn/qt/official_releases/online_installers/

安装后注册账号，按照教程来进行安装，组件选择如下（截图源自教程）：





组件选择如下（截图源自教程）：

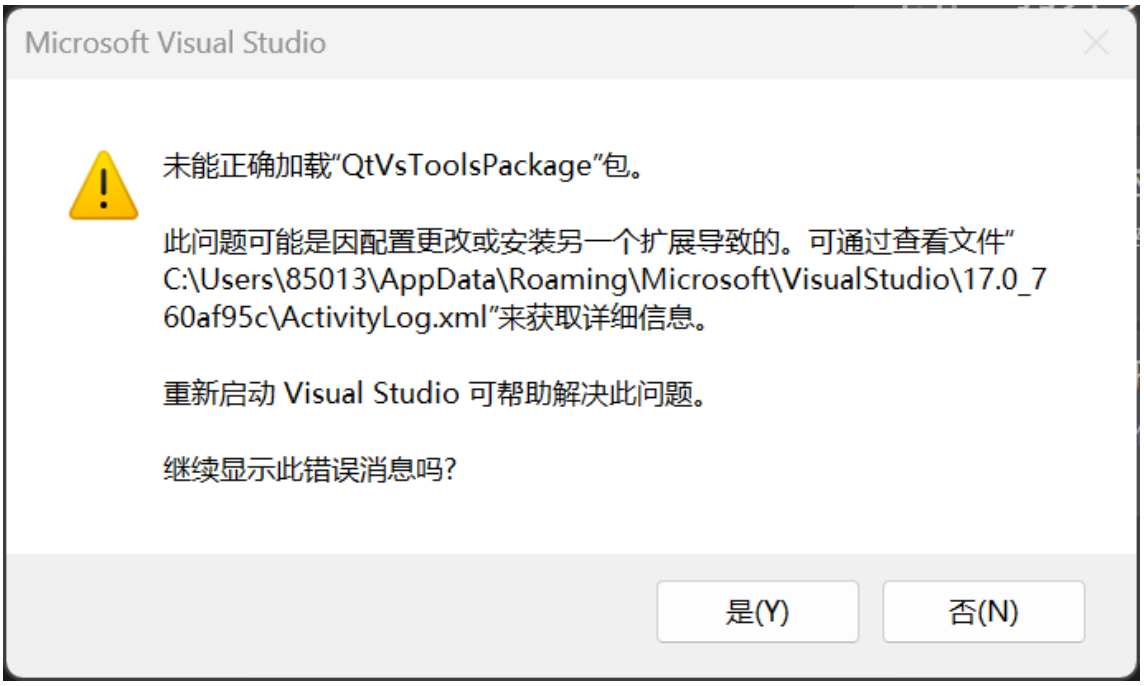


在安装过程中，由于校园网的不稳定，导致一直下载一半就失败要重试，在经历一下午的磋磨后选择用热点下载，非常顺畅。最终下载完成。

3. Qt VS Tools 扩展安装

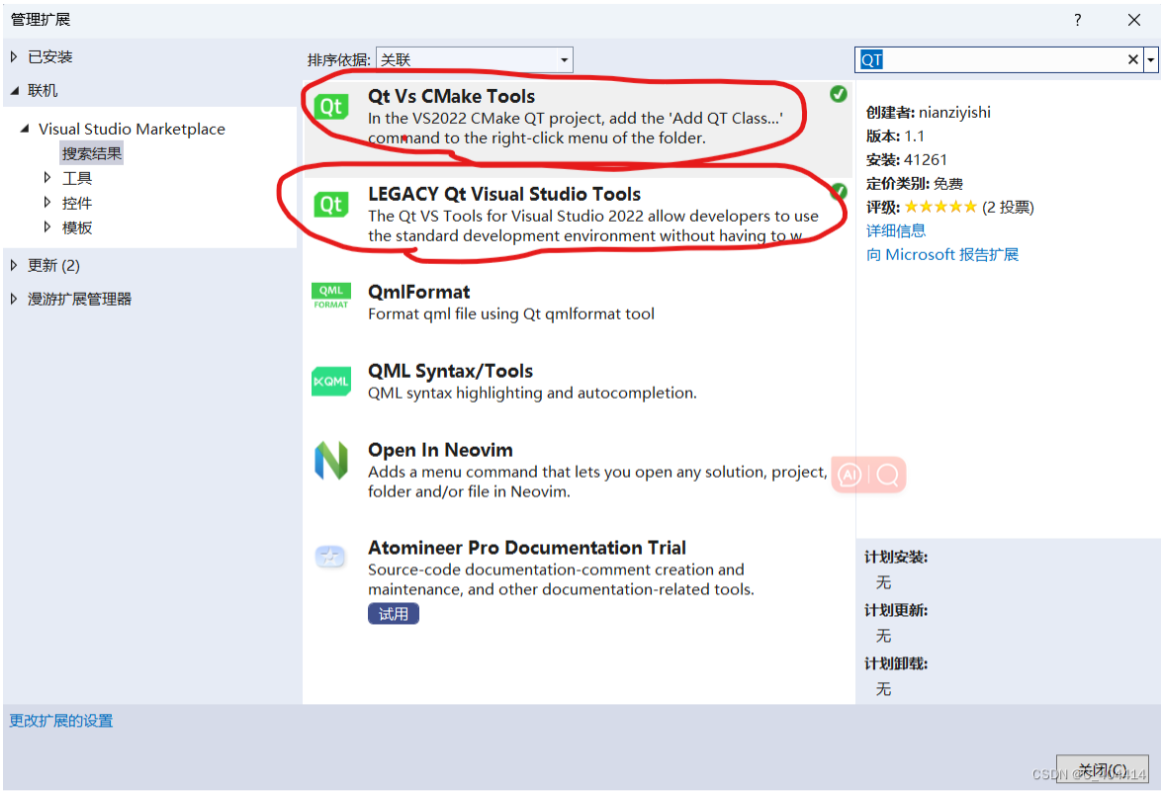
启动 Visual Studio 2022，选择“继续但无需代码”启动。选择“扩展”—>"管理扩展"，搜索Qt Visual Studio Tools。

我是一上来能搜索到此选项，但下载后无法正常启动，总是会出现如下报错：

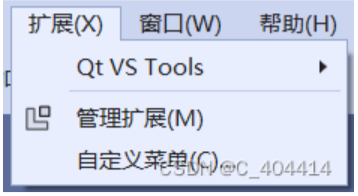


接着下一步启动 VS 进入主界面，准备进入扩展中的 Qt VS Tools 点 开 Qt Options，但发现点开扩展的 Qt VS Tools里无内容。即下载失败。

不太清楚原因，上网查找到要多下载几个组件，于是下载对应两个：



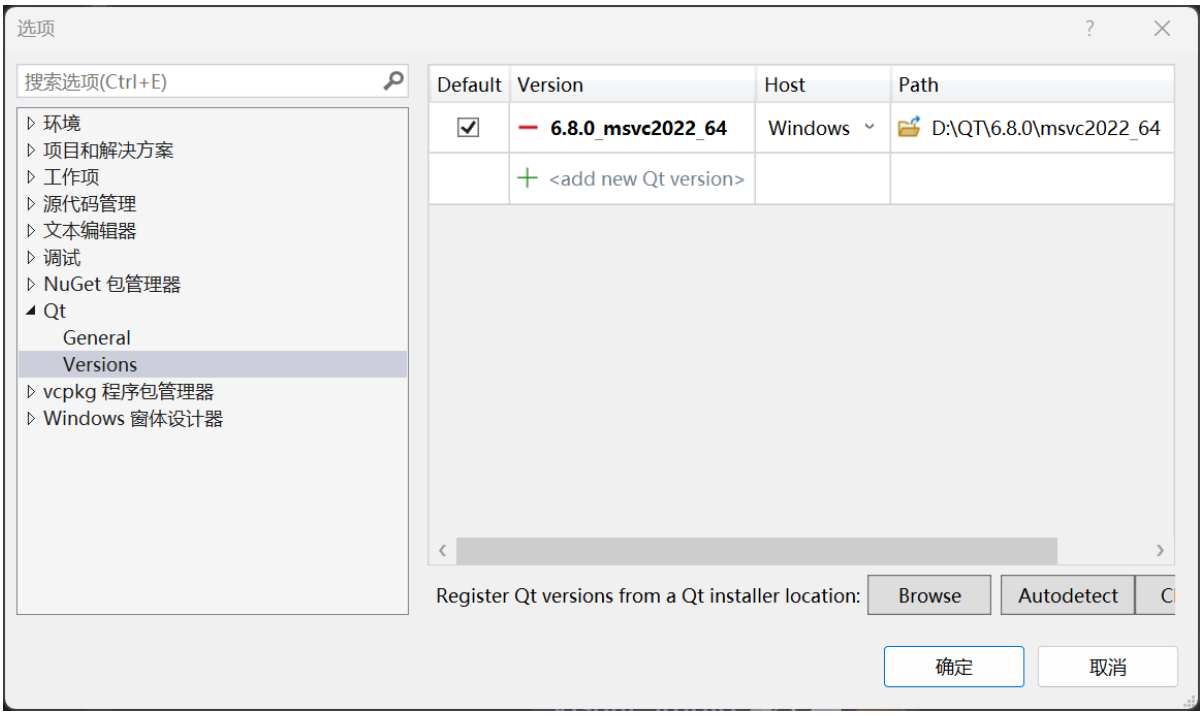
上图两个拓展都要下载，很多教程只说下第一个，就没办法显示了



第二个也下载之后就可以正常显示了，好像vs2017，2019都没有这样的问题

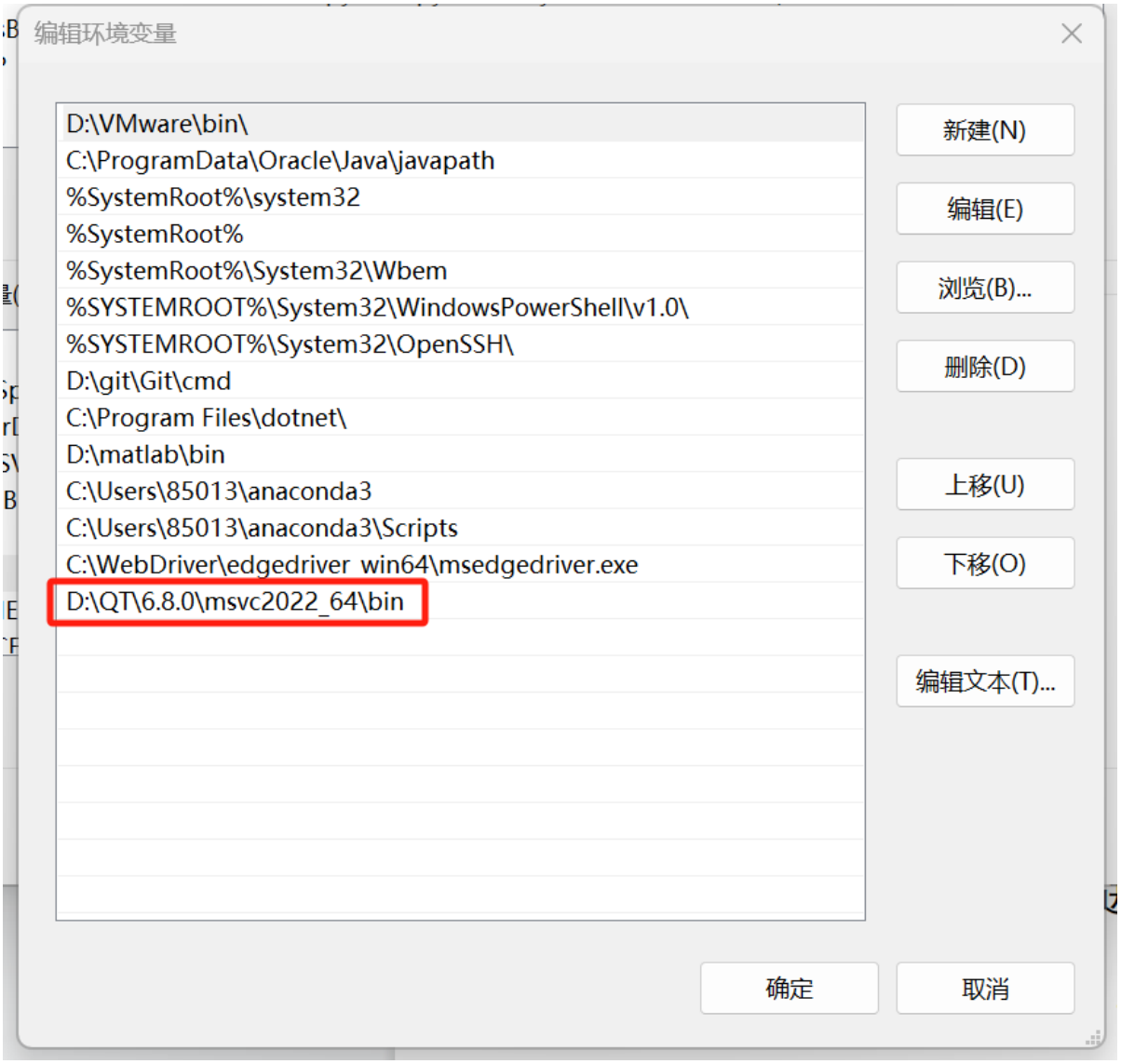
但仍然报错，且两个一起报错，因此只能按照教程下载对应文件https://pan.baidu.com/s/1HXf3ju75VSuR2yQBmCpxhw?pwd=ah31到对应文件夹，之后再安装就能正常使用了。

接着下一步启动 VS 进入主界面，准备进入扩展中的 Qt VS Tools 点开 Qt Options，进入如下页面。刚开始我的无内容，需要点击下方的“Autodetect”，可以检测到有三个版本，选择图中版本。这一步就完成了。

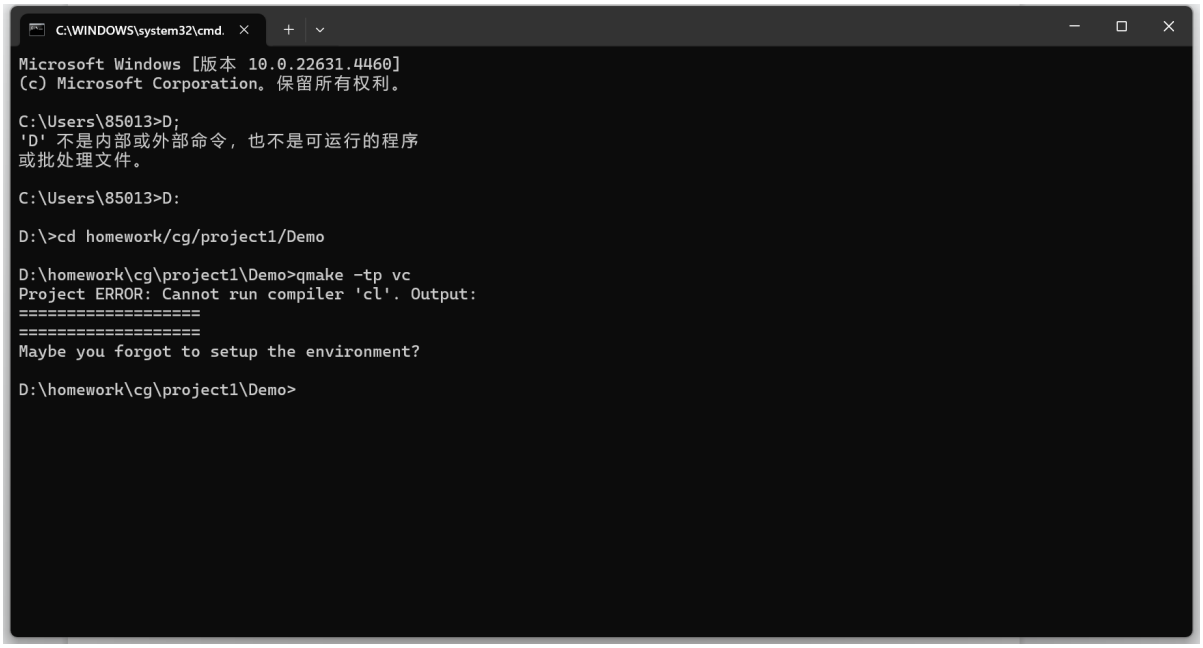


4. 环境配置

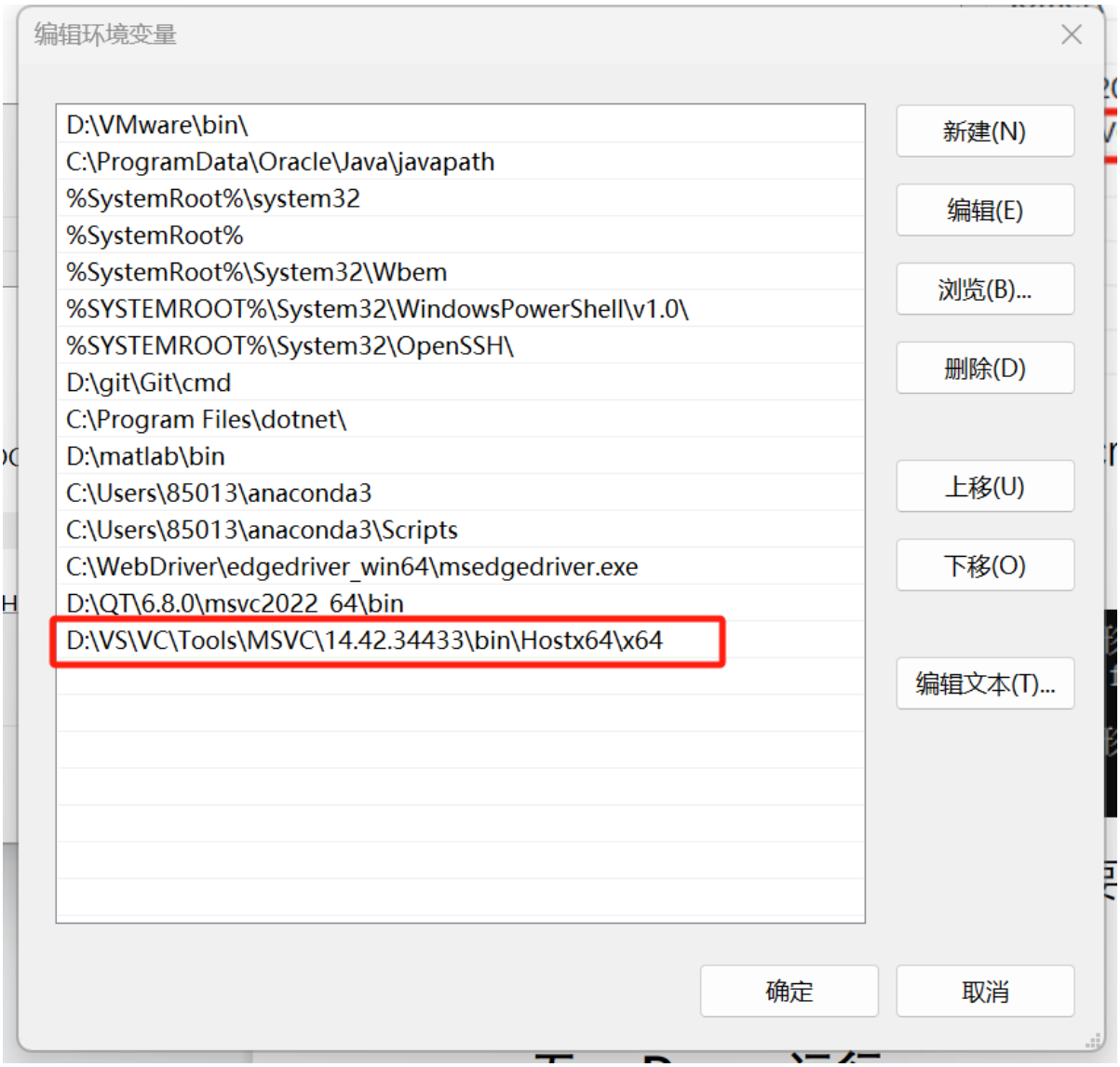
qmake 使用：在系统变量的Path中新建系统环境变量



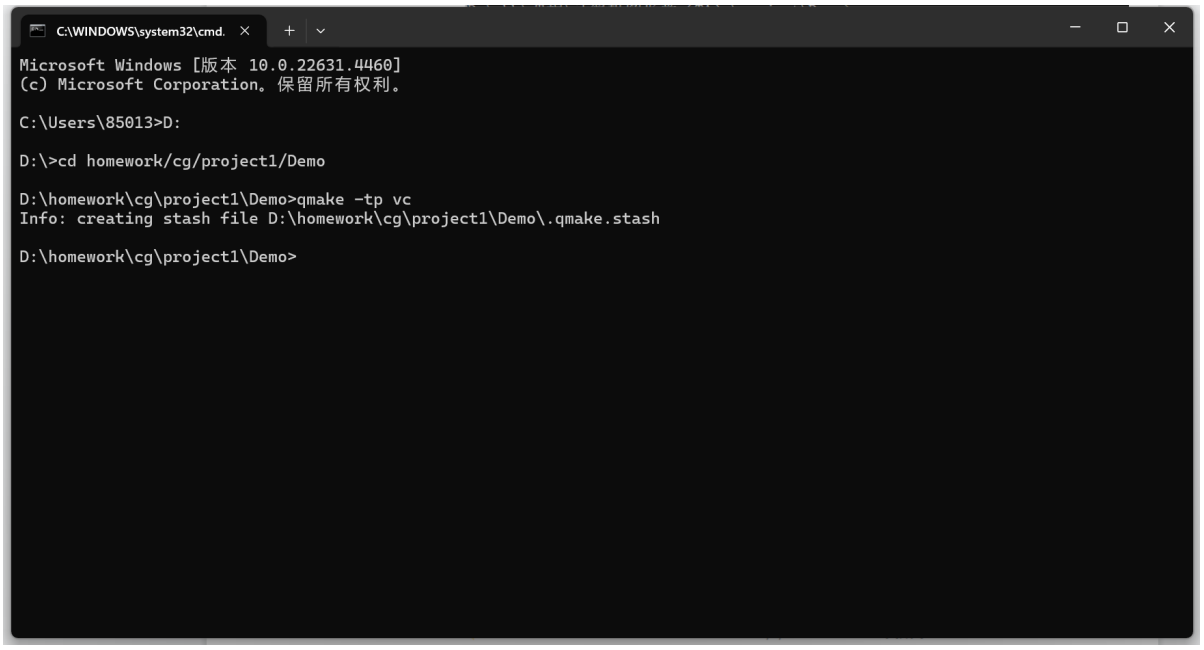
创建Demo文件夹，将课程发的CGTemplate压缩包除了作业要求以外的文件都放入里面。用cmd进入Demo，执行命令 `qmake -tp vc` 生成VS工程文件，会报错：



需要将cl.exe的路径加入系统环境变量：



完成后需要重新打开终端，再次执行命令 `qmake -tp vc`，此时Demo文件夹中会生成CGTemplate.vcxproj 等文件：



名称	修改日期	类型	大小
.vs	2024/11/23 1:07	文件夹	
debug	2024/11/23 2:54	文件夹	
release	2024/11/23 0:37	文件夹	
.qmake.stash	2024/11/23 0:37	STASH 文件	1 KB
CGTemplate	2024/11/23 2:50	Qt Project file	1 KB
CGTemplate.vcxproj	2024/11/23 2:54	VC++ Project	16 KB
CGTemplate.vcxproj.filters	2024/11/23 1:31	VC++ Project Filter...	3 KB
CGTemplate.vcxproj.user	2024/11/23 1:07	Per-User Project O...	1 KB
main.cpp	2024/11/22 21:53	C++ Source	1 KB
myglwidget.cpp	2024/11/22 21:53	C++ Source	2 KB
myglwidget.h	2024/11/23 2:03	C/C++ Header	1 KB

5. Demo运行

下载GLEW：http://glew.sourceforge.net/ 我下载的是2.2.0版本。接着配置OpenGL：打开CGTemplate.pro，加入 `INCLUDEPATH += "path/to/your/glew-2.2.0/include"`

```
QT += core gui opengl

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

CONFIG += console qt c++11

DEFINES += QT_DEPRECATED_WARNINGS

INCLUDEPATH += "D:\homework\cg\glew-2.2.0-win32\glew-2.2.0\include"

LIBS += \
        Glu32.lib \
        OpenGL32.lib
LIBS += glew32.lib

SOURCES += \
    main.cpp \
```

重新`qmake -tp vc`生成项目，用VS打开CGTemplate.vcxproj运行main.cpp文件。然后出现以下两个问题：

1. 无法打开 QOpenGLWidget

查看教程，发现是Qt版本不匹配。实验教程演示的Qt版本为5.13.0，而我安装的是Qt6.8.0。所以教程给的代码有些问题：

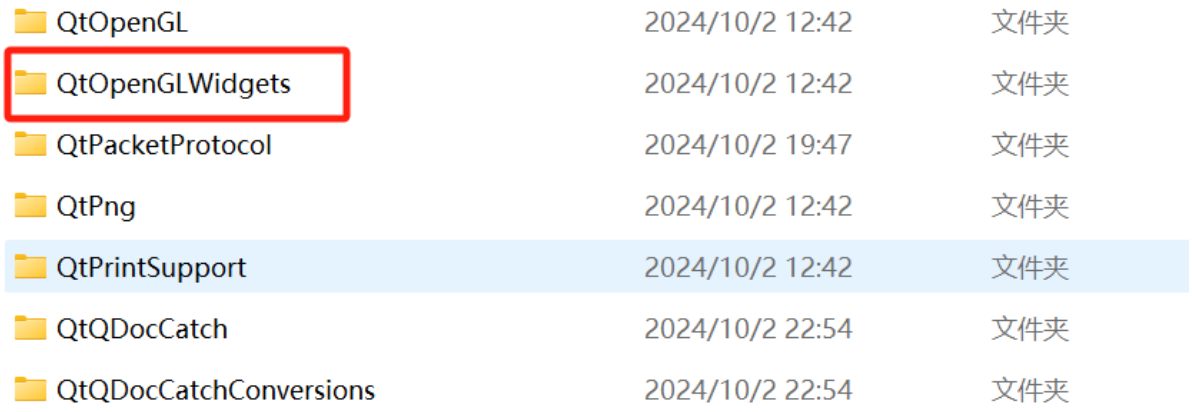
```
QT += core gui opengl
greaterThan(QT_MAJOR_VERSION, 4): QT += widgets
```

将其改为：

```
QT += core gui opengl openglwidgets
```

注意，这里修改后一定要重新`qmake -tp vc`生成项目！！！！再用VS打开CGTemplate.vcxproj运行main.cpp文件。否则就会出现仍然匹配不到的情况。

第二种方法我猜想是因为文件夹名称不符合的原因，而include里是没有s的



```
#include <QtGui>
#include <QOpenGLWidget>
#include <QOpenGLFunctions>

class MyGLWidget : public QOpenGLWidget{
    Q_OBJECT

public:
```

所以把这个改为：

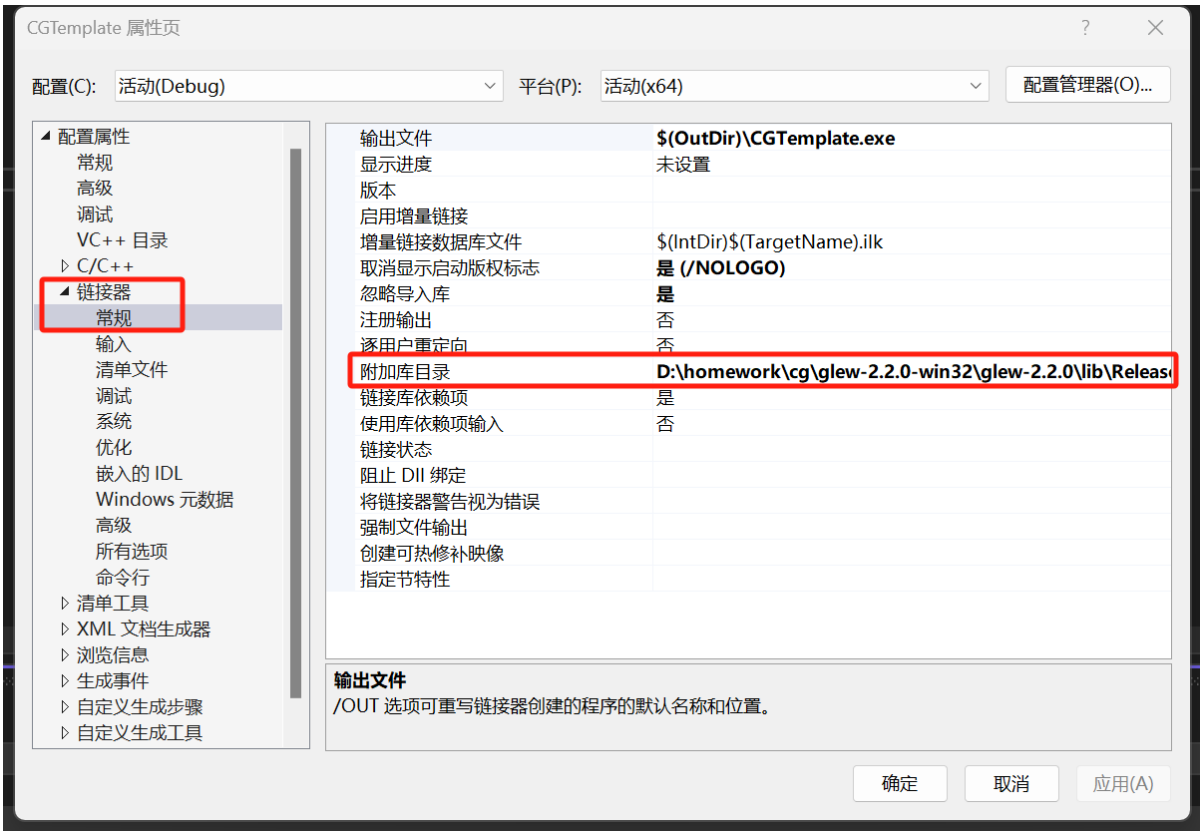
```
#include <QtGui>
#include <QtOpenGLWidgets/QOpenGLWidget>
#include <QOpenGLFunctions>

class MyGLWidget : public QOpenGLWidget{
    Q_OBJECT

public:
```

也可以正常运行。

接下来就会遇到第二个问题。

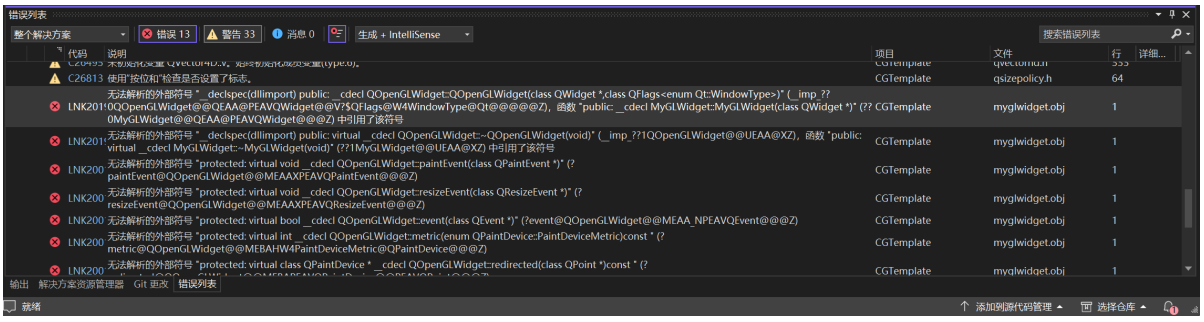
2. 找不到glew32.lib

解决方案资源管理器→CGTemplate→属性→配置属性→链接器→常规→附加库目录，将之前下载的glew32中包含glew32.lib的文件夹的路径添加进去。

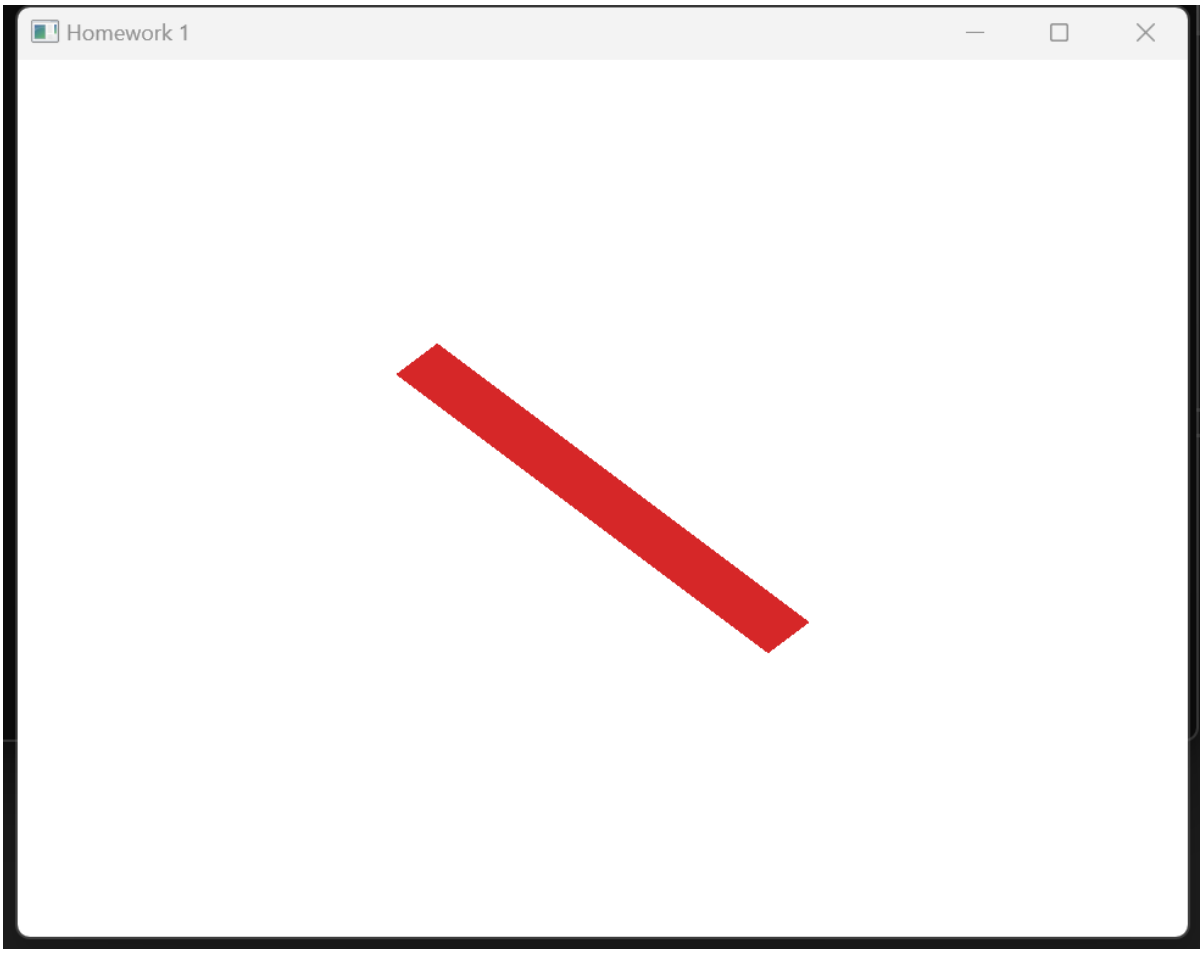


在全部配置完成后，理应是可以完成的，但我不知为何仍然运行不了，折磨了好久：

会出现以下报错：



上网查询说出现这种报错都是因为lib文件找不到。我按照网上的方法都试了一遍，仍然不行。最后重新把这两个问题重新做了一遍，发现自己漏了**修改完pro文件后的重新** `qmake -tp vc` **生成项目**，最后终于可以运行，虽然不知道两者有什么关系，但也算是完成了：



终于能开始实验了！！

二、实验内容

1. **在二维画布上，使用基本图元，以原点为绘制中心，绘制自己姓名首字母。**

   1.1 并比较 GL_TRIANGLES, GL_TRIANGLE_STRIP, GL_QUAD_STRIP 的绘制开销（需要的 glVertex 调用次数）；

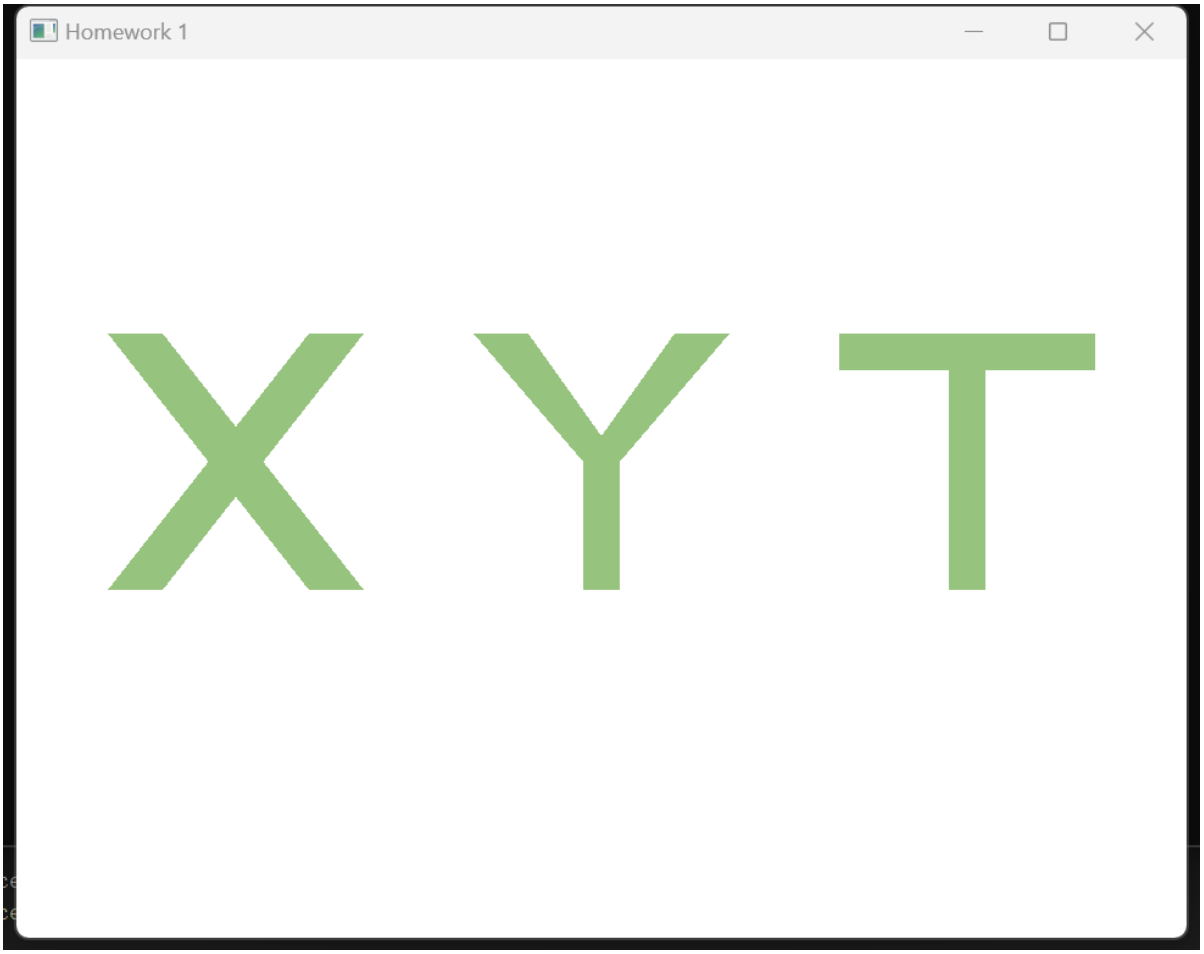
   GL_TRIANGLES：绘制三角形，根据顶点生成三角形

   调用次数12+18+12=42次

```
void MyGLWidget::scene_1()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(0.0f, width(), 0.0f, height(), -1000.0f, 1000.0f);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glTranslatef(0.5 * width(), 0.5 * height(), 0.0f);

    glPushMatrix();

    // 绘制字母 X
```

```
    glColor3f(0.588f, 0.765f, 0.49f); // 绿色
    glTranslatef(-200.0f, -50.0f, 0.0f);
    glBegin(GL_TRIANGLES);
    // 左上到右下对角线
    glVertex2f(70.0f, 0.0f); glVertex2f(-40.0f, 140.0f); glVertex2f(-70.0f,
140.0f);
    glVertex2f(70.0f, 0.0f); glVertex2f(40.0f, 0.0f); glVertex2f(-70.0f, 140.0f);

    // 右上到左下对角线
    glVertex2f(-70.0f, 0.0f); glVertex2f(40.0f, 140.0f); glVertex2f(70.0f,
140.0f);
    glVertex2f(-70.0f, 0.0f); glVertex2f(-40.0f, 0.0f); glVertex2f(70.0f,
140.0f);
    glEnd();
    glPopMatrix();

    // 绘制字母 Y
    glPushMatrix();
    glTranslatef(0.0f, -50.0f, 0.0f);
    glBegin(GL_TRIANGLES);
    // 上部
    // 左上到右下对角线
    glVertex2f(-70.0f, 140.0f); glVertex2f(-40.0f, 140.0f); glVertex2f(10.0f,
70.0f);
    glVertex2f(-10.0f, 70.0f); glVertex2f(10.0f, 70.0f); glVertex2f(-70.0f,
140.0f);
    // 右上到左下对角线
    glVertex2f(70.0f, 140.0f); glVertex2f(40.0f, 140.0f); glVertex2f(-10.0f,
70.0f);
    glVertex2f(10.0f, 70.0f); glVertex2f(-10.0f, 70.0f); glVertex2f(70.0f,
140.0f);

    // 竖
    glVertex2f(-10.0f, 70.0f); glVertex2f(10.0f, 70.0f); glVertex2f(10.0f, 0.0f);
    glVertex2f(-10.0f, 70.0f); glVertex2f(-10.0f, 0.0f); glVertex2f(10.0f, 0.0f);
    glEnd();
    glPopMatrix();

    // 绘制字母 T
    glPushMatrix();
    glTranslatef(200.0f, -50.0f, 0.0f);
    glBegin(GL_TRIANGLES);
    // 顶部横线
    glVertex2f(-70.0f, 140.0f); glVertex2f(70.0f, 140.0f); glVertex2f(-70.0f,
120.0f);
    glVertex2f(70.0f, 140.0f); glVertex2f(70.0f, 120.0f); glVertex2f(-70.0f,
120.0f);

    // 竖线
    glVertex2f(-10.0f, 0.0f); glVertex2f(-10.0f, 140.0f); glVertex2f(10.0f,
0.0f);
    glVertex2f(10.0f, 140.0f); glVertex2f(-10.0f, 140.0f); glVertex2f(10.0f,
0.0f);
    glEnd();
    glPopMatrix();
```

```
    glPopMatrix();
}
```



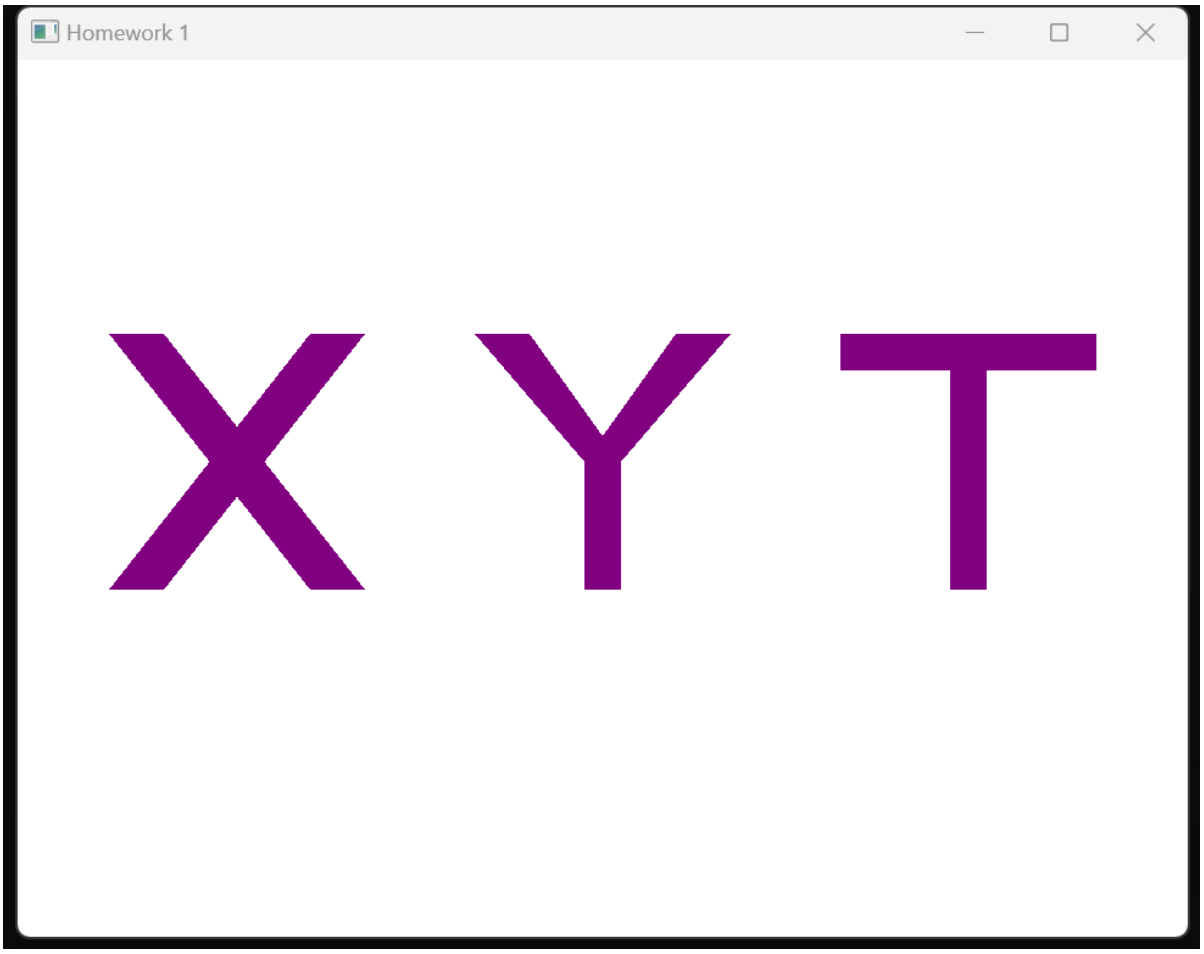
GL_TRIANGLE_STRIP：使用共享顶点，减少重复顶点的定义。每组相邻顶点自动生成一个三角形。开销更少

调用次数：8+10+8=26次

```
void MyGLWidget::scene_2()// 注意：由于我是直接加了一个函数scene_2，所以对应的paintGL、KeyPressEvent、以及myglwidget.h文件中的初始化都要加。按规律加即可，我就不再赘述，下面同理。
{
    glClear(GL_COLOR_BUFFER_BIT);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(0.0f, width(), 0.0f, height(), -1000.0f, 1000.0f);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glTranslatef(0.5 * width(), 0.5 * height(), 0.0f);

    glPushMatrix();

    // 绘制字母 X
    glColor3f(0.5f, 0.0f, 0.5f);// 紫色
    glPushMatrix();
    glTranslatef(-200.0f, -50.0f, 0.0f);
    glBegin(GL_TRIANGLE_STRIP);
    // 左上到右下对角线
```

```
    glVertex2f(-40.0f, 140.0f); // 1
    glVertex2f(-70.0f, 140.0f); // 2
    glVertex2f(70.0f, 0.0f);   // 3
    glVertex2f(40.0f, 0.0f);    // 第2个三角形顶点4（共享顶点2和3）
    glEnd();
    glBegin(GL_TRIANGLE_STRIP);
    // 右上到左下对角线
    glVertex2f(40.0f, 140.0f); // 1
    glVertex2f(70.0f, 140.0f);// 2
    glVertex2f(-70.0f, 0.0f);  // 3
    glVertex2f(-40.0f, 0.0f);    // 第2个三角形顶点4（共享顶点2和3）
    glEnd();
    glPopMatrix();


    // 绘制字母 Y
    glColor3f(0.5f, 0.0f, 0.5f);
    glPushMatrix();
    glTranslatef(0.0f, -50.0f, 0.0f);
    glBegin(GL_TRIANGLE_STRIP);
    // 上部左侧对角线
    glVertex2f(-70.0f, 140.0f);
    glVertex2f(-40.0f, 140.0f);
    glVertex2f(-10.0f, 70.0f);// 两个对角线共用这两条
    glVertex2f(10.0f, 70.0f);// 两个对角线共用这两条

    glVertex2f(40.0f, 140.0f);
    glVertex2f(70.0f, 140.0f);
    glEnd();

    glBegin(GL_TRIANGLE_STRIP);
    // 竖线
    glVertex2f(10.0f, 70.0f);
    glVertex2f(10.0f, 0.0f);
    glVertex2f(-10.0f, 70.0f);
    glVertex2f(-10.0f, 0.0f);
    glEnd();
    glPopMatrix();


    // 绘制字母 T
    glColor3f(0.5f, 0.0f, 0.5f);
    glPushMatrix();
    glTranslatef(200.0f, -50.0f, 0.0f);
    glBegin(GL_TRIANGLE_STRIP);
    // 顶部横线
    glVertex2f(-70.0f, 140.0f);
    glVertex2f(-70.0f, 120.0f);
    glVertex2f(70.0f, 140.0f);
    glVertex2f(70.0f, 120.0f);
    glEnd();
    glBegin(GL_TRIANGLE_STRIP);
    // 竖线
    glVertex2f(-10.0f, 0.0f);
    glVertex2f(-10.0f, 140.0f);
    glVertex2f(10.0f, 0.0f);
```

```
    glVertex2f(10.0f, 140.0f);

    glEnd();
    glPopMatrix();

    glPopMatrix();
}
```



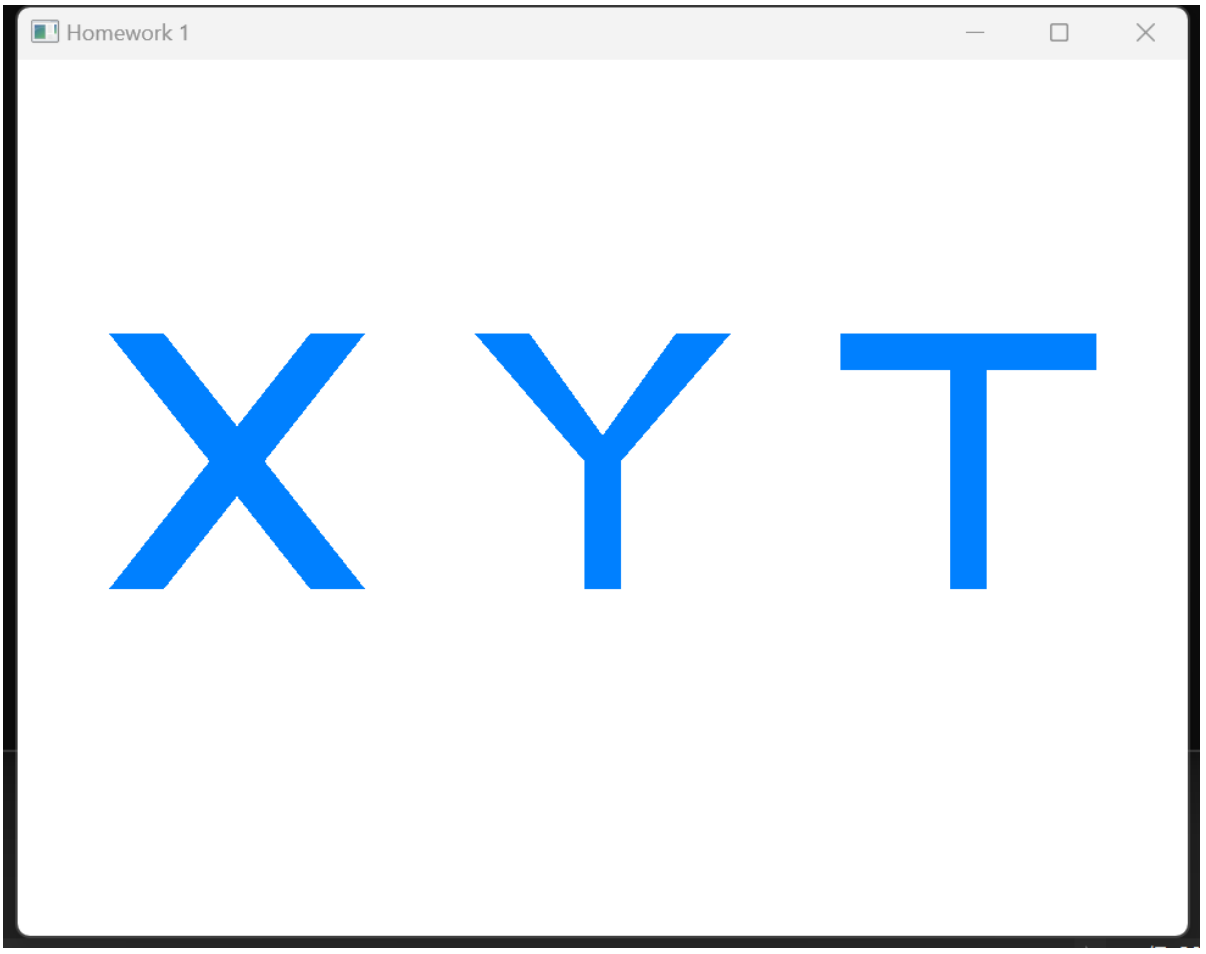
GL_QUAD_STRIP：这是四边形的使用共享顶点，也许是字母原因，与上面GL_TRIANGLE_STRIP的代码逻辑都是一致的

调用次数：8+10+8=26次

```
void MyGLWidget::scene_3()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(0.0f, width(), 0.0f, height(), -1000.0f, 1000.0f);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glTranslatef(0.5 * width(), 0.5 * height(), 0.0f);

    glPushMatrix();

    // 绘制字母 X
    glColor3f(0.0f, 0.5f, 1.0f);
```

```
    glPushMatrix();
    glTranslatef(-200.0f, -50.0f, 0.0f);

    // 左竖和左撇
    glBegin(GL_QUAD_STRIP);
    // 左上到右下对角线
    glVertex2f(-40.0f, 140.0f);
    glVertex2f(-70.0f, 140.0f);
    glVertex2f(70.0f, 0.0f);
    glVertex2f(40.0f, 0.0f);
    glEnd();
    glBegin(GL_QUAD_STRIP);
    // 右上到左下对角线
    glVertex2f(40.0f, 140.0f);
    glVertex2f(70.0f, 140.0f);
    glVertex2f(-70.0f, 0.0f);
    glVertex2f(-40.0f, 0.0f);
    glEnd();

    glPopMatrix();


    // 绘制字母 Y
    glColor3f(0.0f, 0.5f, 1.0f);
    glPushMatrix();
    glTranslatef(0.0f, -50.0f, 0.0f);
    glBegin(GL_QUAD_STRIP);
    // 上部左侧对角线
    glVertex2f(-70.0f, 140.0f);
    glVertex2f(-40.0f, 140.0f);
    glVertex2f(-10.0f, 70.0f);// 两个对角线对角线共用这两个顶点
    glVertex2f(10.0f, 70.0f);// 两个对角线对角线共用这两个顶点

    glVertex2f(40.0f, 140.0f);
    glVertex2f(70.0f, 140.0f);

    glEnd();

    glBegin(GL_QUAD_STRIP);
    // 竖线
    glVertex2f(10.0f, 70.0f);
    glVertex2f(10.0f, 0.0f);
    glVertex2f(-10.0f, 70.0f);
    glVertex2f(-10.0f, 0.0f);
    glEnd();
    glPopMatrix();


    // 绘制字母 T
    glColor3f(0.0f, 0.5f, 1.0f);
    glPushMatrix();
    glTranslatef(200.0f, -50.0f, 0.0f);
    glBegin(GL_QUAD_STRIP);
    // 顶部横线
    glVertex2f(-70.0f, 140.0f);
    glVertex2f(-70.0f, 120.0f);
```

```
    glVertex2f(70.0f, 140.0f);
    glVertex2f(70.0f, 120.0f);
    glEnd();
    glBegin(GL_QUAD_STRIP);
    // 竖线
    glVertex2f(-10.0f, 0.0f);
    glVertex2f(-10.0f, 140.0f);
    glVertex2f(10.0f, 0.0f);
    glVertex2f(10.0f, 140.0f);

    glEnd();

    glPopMatrix();
    glPopMatrix();
}
```



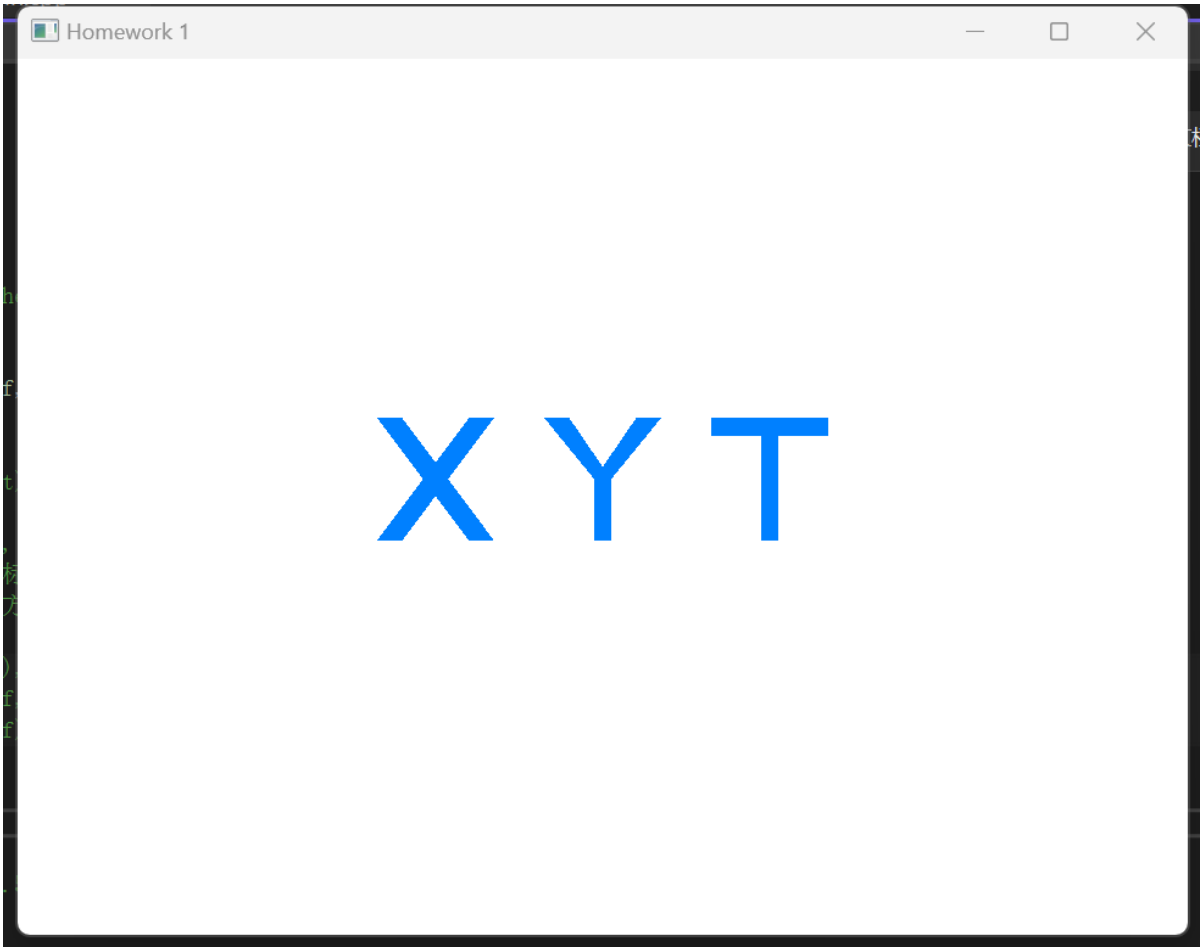
1.2 比较以下两个视角下, Orthogonal及Perspective投影方式产生的图像

由于窗口问题，我这里改一下窗口视角：

```
    glClear(GL_COLOR_BUFFER_BIT);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

    //glOrtho(0.0f, width(), 0.0f, height(), -1000.0f, 1000.0f);
    // 正交投影
    glOrtho(-700.0f, 700.0f, -500.0f, 500.0f, -1000.0f, 1000.0f);// 即由上面代码改为这样，并将下面的translatef删掉
```

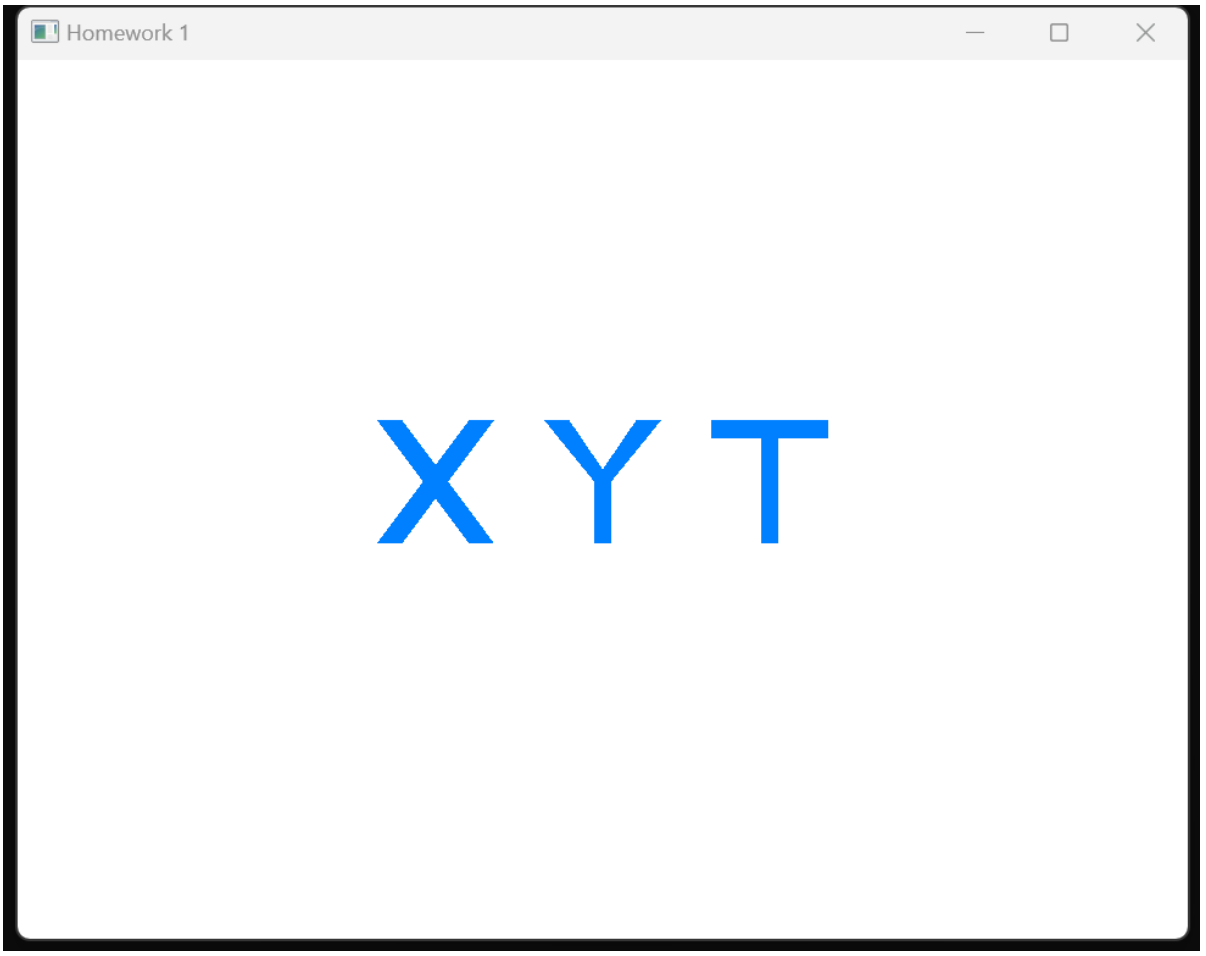
```
// 透视投影
// gluPerspective(45.0, (GLfloat)width() / (GLfloat)height(), 0.1, 100.0);

// 从(0,0,d)看向原点(0,0,0)
/*gluLookAt(0.0f, 0.0f, 1000.0f,  // 观察点位置
    0.0f, 0.0f, 0.0f,     // 目标点位置
    0.0f, 1.0f, 0.0f);    // 上方向*/

//从（0,0.5*d ,d）看向原点(0,0,0)
/*gluLookAt(0.0f, 500.0f, 1000.0f,  // 观察点位置
      0.0f, 0.0f          , 0.0f,     // 目标点位置
      0.0f, 1.0f          , 0.0f);    // 上方向*/


glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
//glTranslatef(0.5 * width(), 0.5 * height(), 0.0f);

glPushMatrix();
```

初始状态:

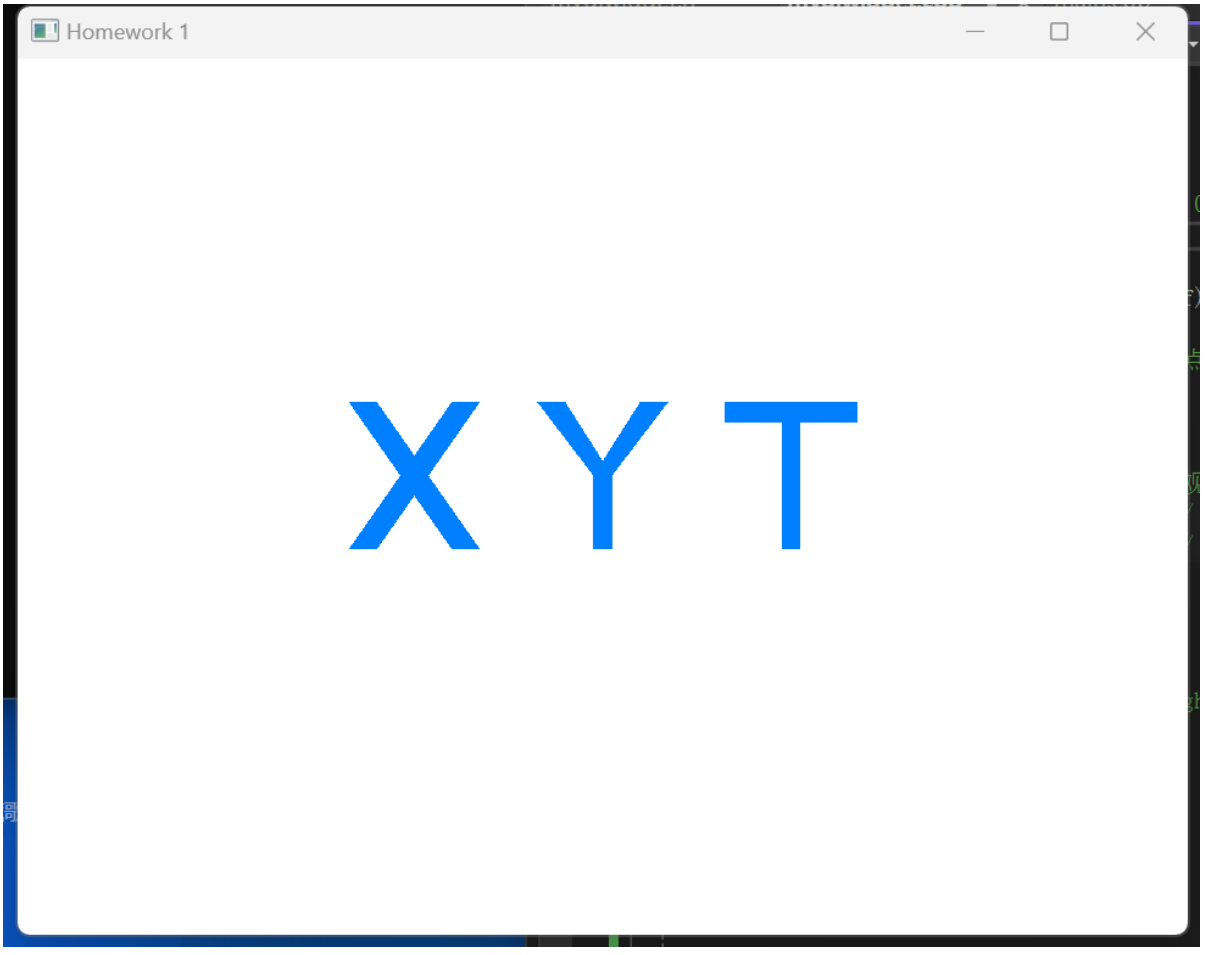


从(0,0,d)看向原点(0,0,0):

正交:



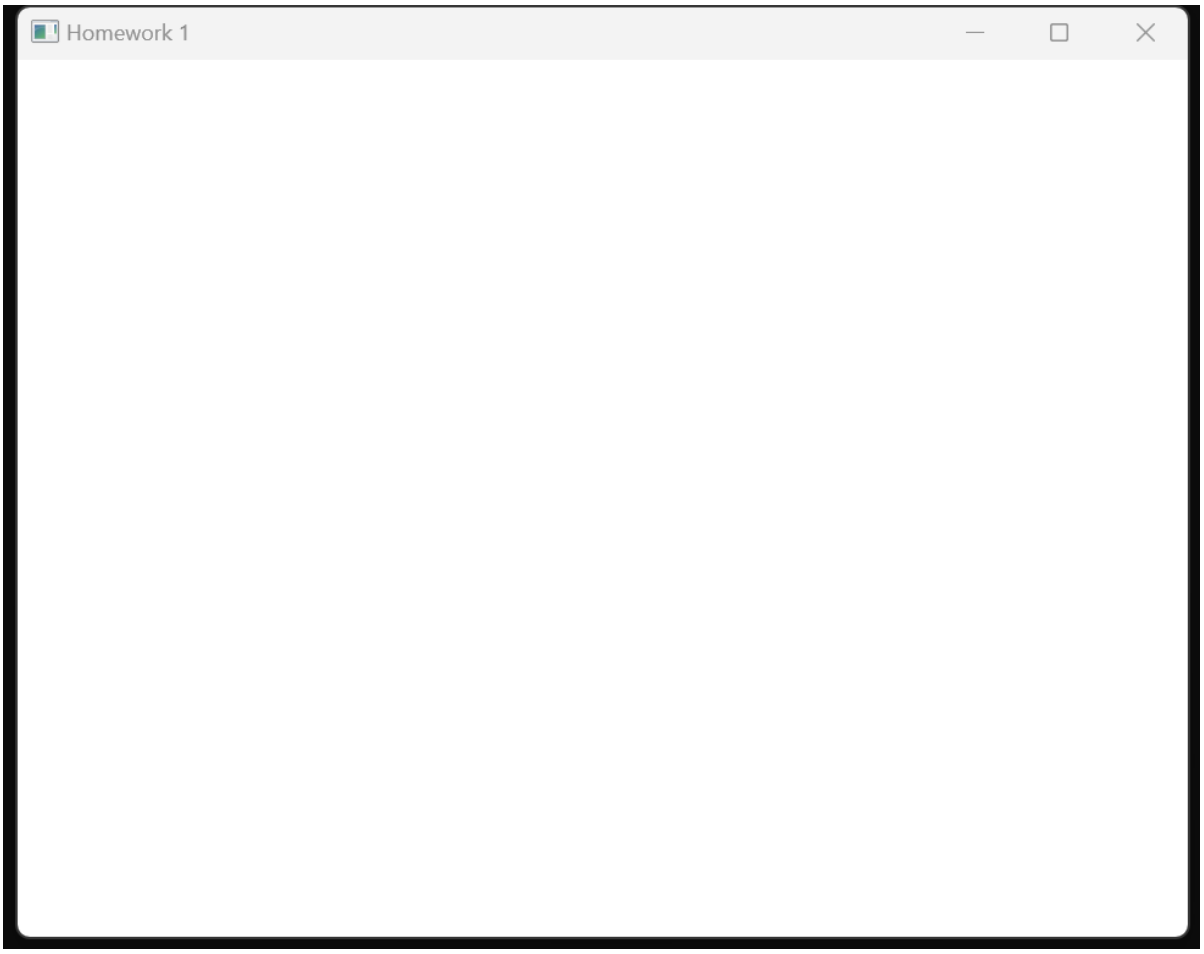
透视:



从（0,0.5*d ,d）看向原点(0,0,0):

正交:

透视:

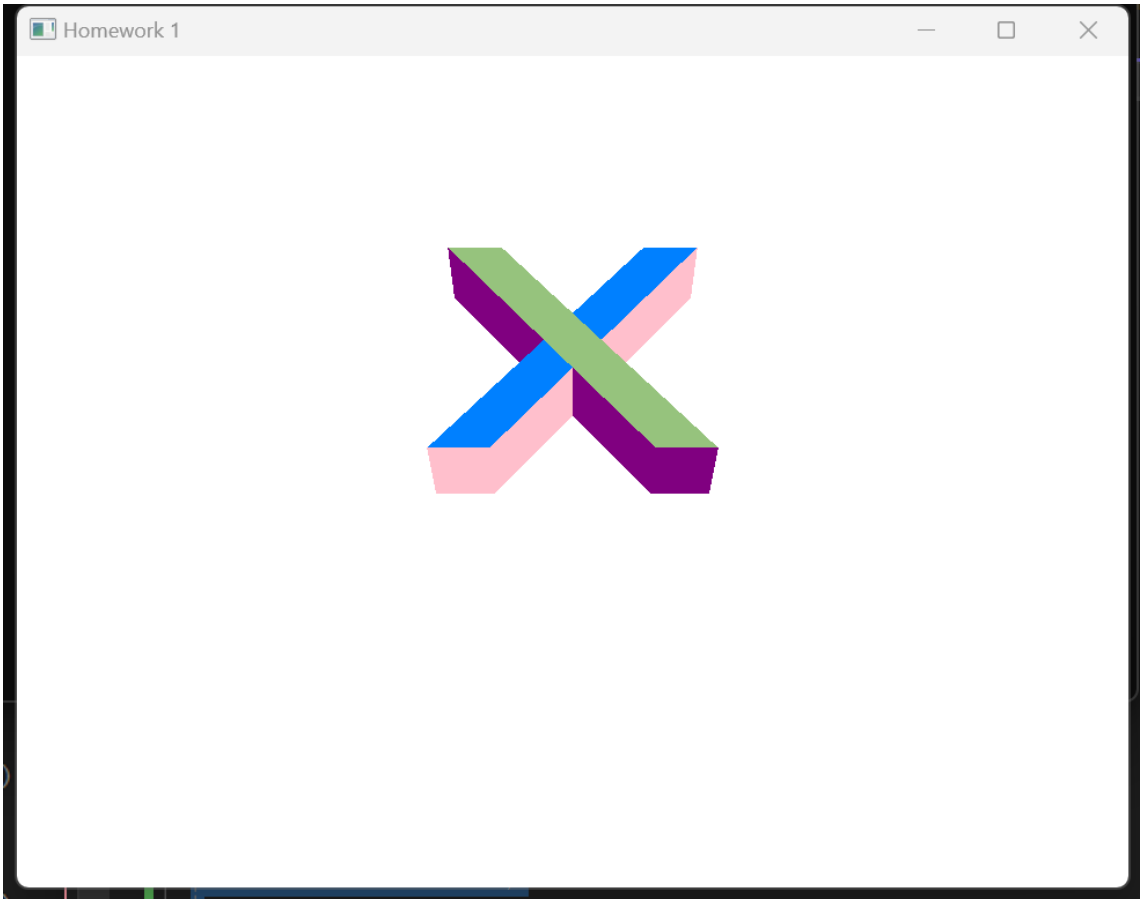


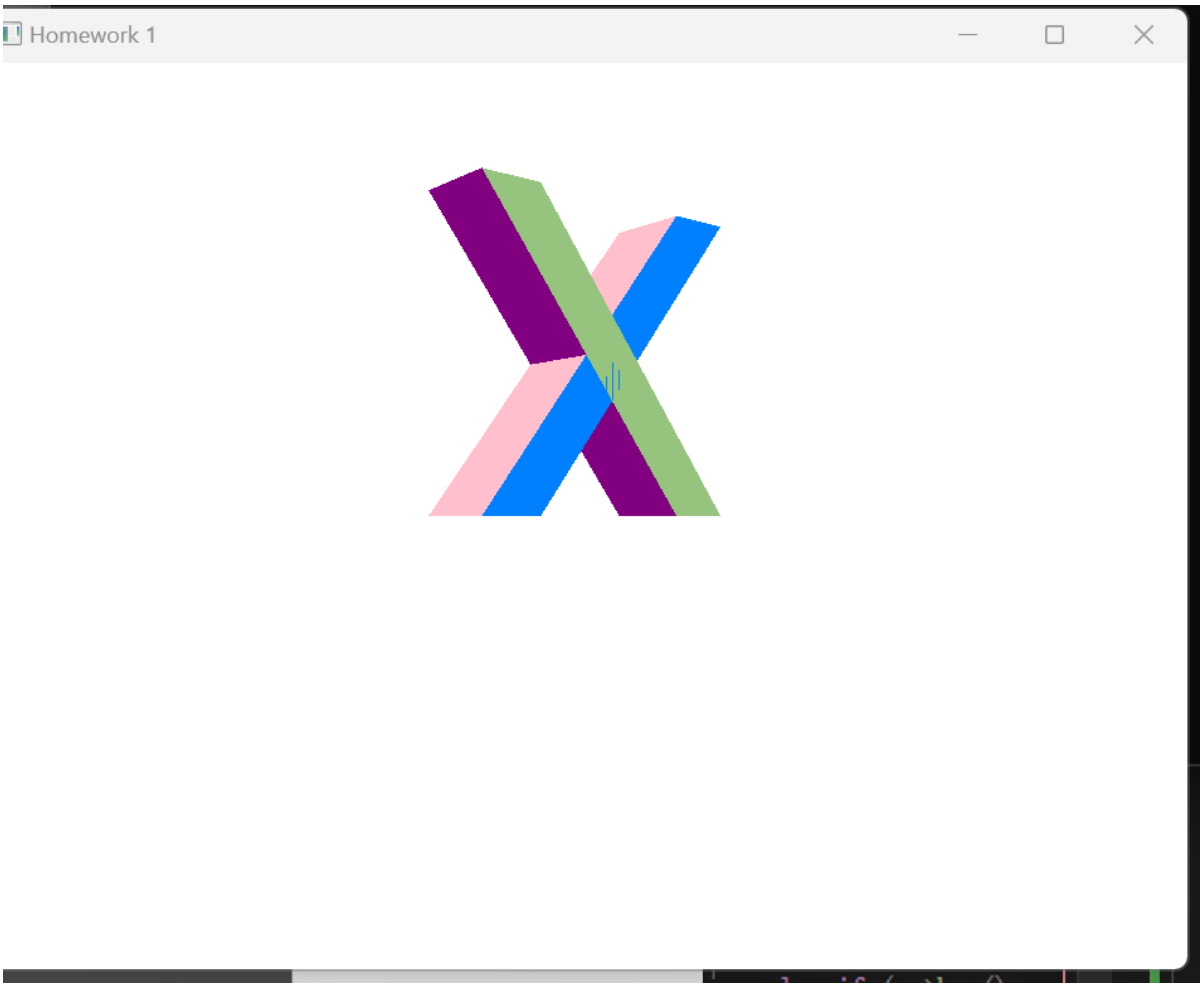
2. 绘制立体姓氏首字母

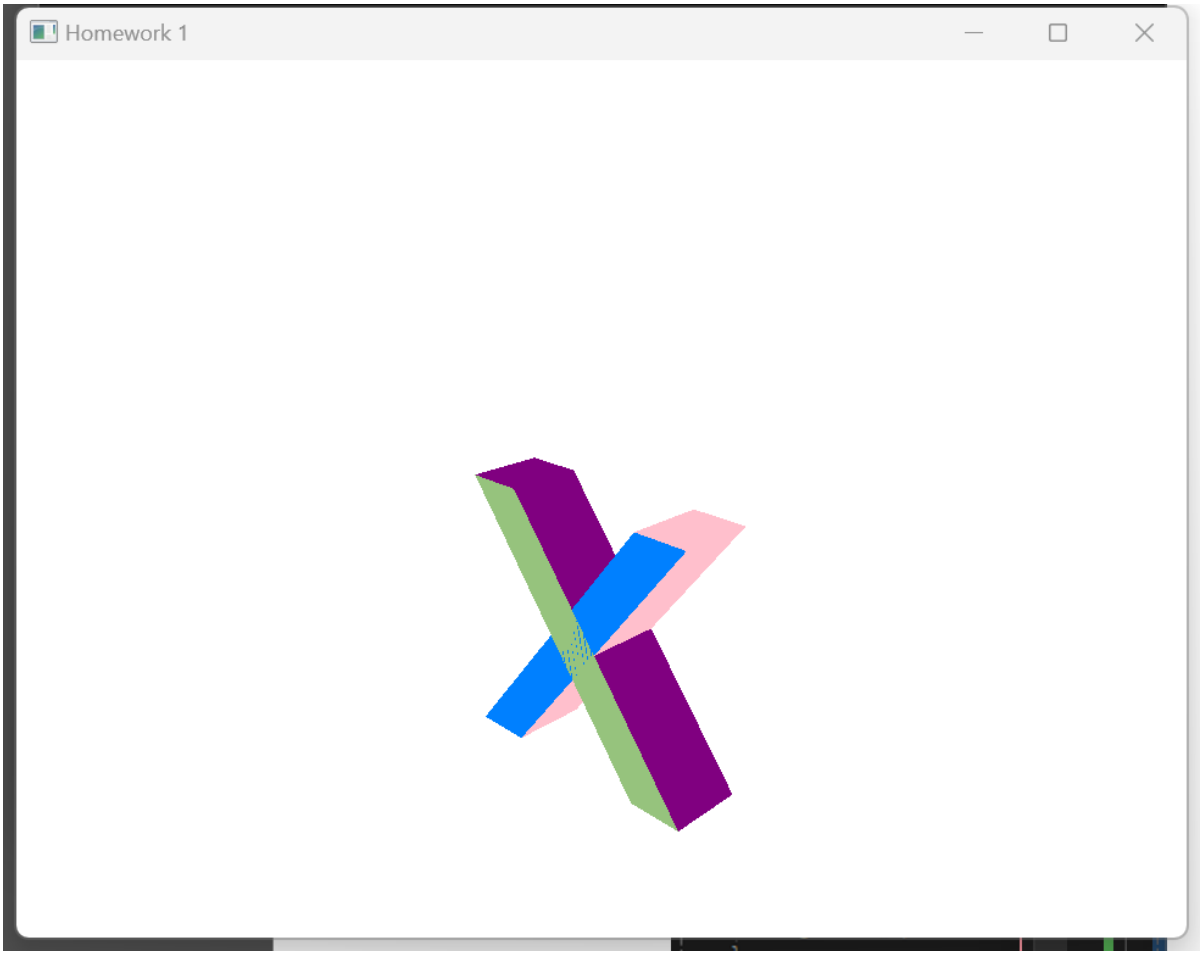
```
void MyGLWidget::scene_4() {
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glEnable(GL_DEPTH_TEST);

    // 设置投影矩阵
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45.0f, (GLfloat)width() / (GLfloat)height(), 1.0f,
2000.0f);

    // 设置模型视图矩阵
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(0.0f, 0.0f, 500.0f, // 眼睛位置
        0.0f, 0.0f, 0.0f,  // 看向点
        0.0f, 1.0f, 0.0f); // 上向量

    // 应用旋转
    glRotatef(rotationX, 1.0f, 0.0f, 0.0f); // 绕X轴旋转
    glRotatef(rotationY, 0.0f, 1.0f, 0.0f); // 绕Y轴旋转
    glRotatef(rotationZ, 0.0f, 0.0f, 1.0f); // 绕Z轴旋转

    // 设置字母 "X" 的厚度
    float thickness = 40.0f;
```

```
// 绘制字母 "X"


// 左上到右下对角线
glBegin(GL_QUAD_STRIP);
glColor3f(0.588f, 0.765f, 0.49f); // 绿色
glVertex3f(-40.0f, 140.0f, thickness / 2);
glVertex3f(-70.0f, 140.0f, thickness / 2);
glVertex3f(70.0f, 0.0f, thickness / 2);
glVertex3f(40.0f, 0.0f, thickness / 2);
glEnd();

glBegin(GL_QUAD_STRIP);
glColor3f(0.588f, 0.765f, 0.49f); // 绿色
glVertex3f(-40.0f, 140.0f, -thickness / 2);
glVertex3f(-70.0f, 140.0f, -thickness / 2);
glVertex3f(70.0f, 0.0f, -thickness / 2);
glVertex3f(40.0f, 0.0f, -thickness / 2);
glEnd();

// 右上到左下对角线
glBegin(GL_QUAD_STRIP);
glColor3f(0.0f, 0.5f, 1.0f); // 蓝色
glVertex3f(40.0f, 140.0f, thickness / 2);
glVertex3f(70.0f, 140.0f, thickness / 2);
glVertex3f(-70.0f, 0.0f, thickness / 2);
glVertex3f(-40.0f, 0.0f, thickness / 2);
glEnd();

glBegin(GL_QUAD_STRIP);
glColor3f(0.0f, 0.5f, 1.0f); // 蓝色
glVertex3f(40.0f, 140.0f, -thickness / 2);
glVertex3f(70.0f, 140.0f, -thickness / 2);
glVertex3f(-70.0f, 0.0f, -thickness / 2);
glVertex3f(-40.0f, 0.0f, -thickness / 2);
glEnd();

//connect
// 左上到右下对角线
glBegin(GL_QUAD_STRIP);
glColor3f(0.5f, 0.0f, 0.5f);
glVertex3f(-70.0f, 140.0f, thickness / 2);
glVertex3f(-70.0f, 140.0f, -thickness / 2);
glVertex3f(-40.0f, 140.0f, thickness / 2);
glVertex3f(-40.0f, 140.0f, -thickness / 2);
glVertex3f(70.0f, 0.0f, thickness / 2);
glVertex3f(70.0f, 0.0f, -thickness / 2);
glVertex3f(40.0f, 0.0f, thickness / 2);
glVertex3f(40.0f, 0.0f, -thickness / 2);
glVertex3f(-70.0f, 140.0f, thickness / 2);
glVertex3f(-70.0f, 140.0f, -thickness / 2);

glEnd();

// 右上到左下对角线
glBegin(GL_QUAD_STRIP);
```

```
    glColor3f(1.0f, 0.75f, 0.8f);
    glVertex3f(70.0f, 140.0f, thickness / 2);
    glVertex3f(70.0f, 140.0f, -thickness / 2);
    glVertex3f(40.0f, 140.0f, thickness / 2);
    glVertex3f(40.0f, 140.0f, -thickness / 2);
    glVertex3f(-70.0f, 0.0f, thickness / 2);
    glVertex3f(-70.0f, 0.0f, -thickness / 2);
    glVertex3f(-40.0f, 0.0f, thickness / 2);
    glVertex3f(-40.0f, 0.0f, -thickness / 2);
    glVertex3f(70.0f, 140.0f, thickness / 2);
    glVertex3f(70.0f, 140.0f, -thickness / 2);
    glEnd();


    glPopMatrix();
    glDisable(GL_DEPTH_TEST);
}
```







根据色彩可以看到X的不同区域以及建模方式。

至此，实验完成。