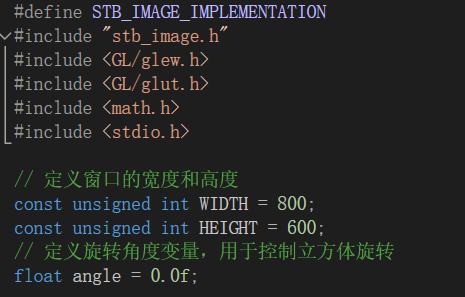
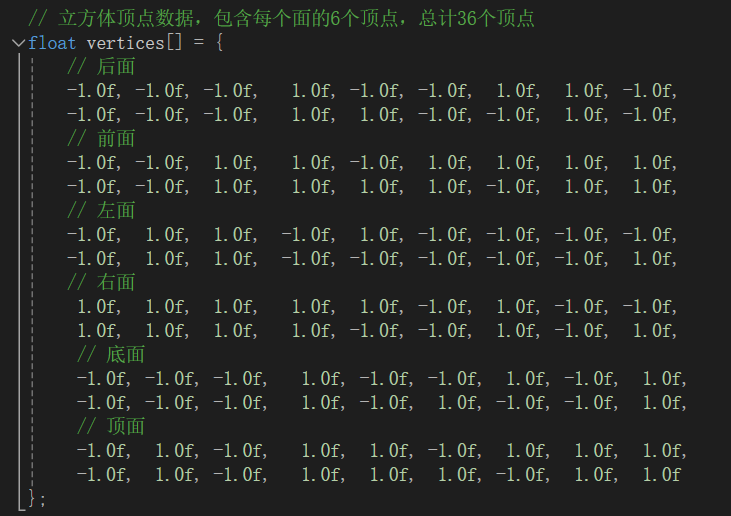
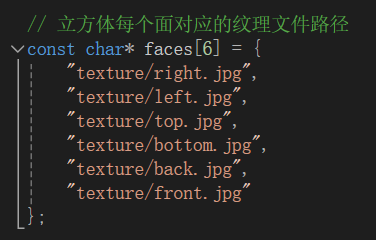
## 头文件与全局变量



引用了stb\_image.h文件以实现正方体贴图文件的加载。WIDTH与HEIGHT用于定义生成窗口的宽度与高度，angle为立方体的旋转角度，用于控制立方体的旋转操作。

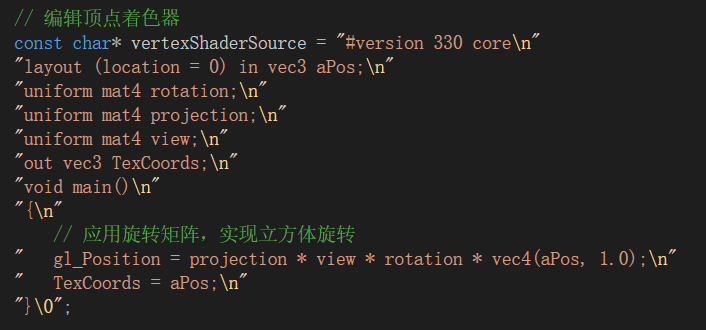


vertices数组用于存储立方体的顶点信息，立方体的每个面有六个顶点（两个三角形）。

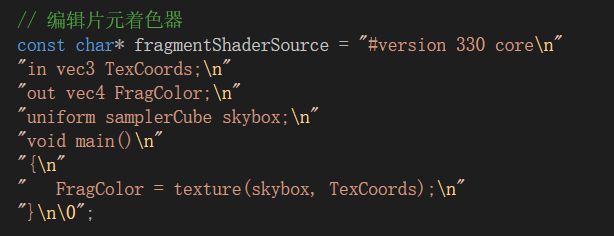


faces数组用于存储立方体每个面纹理文件的地址信息，便于后续导入。

## 修改着色器

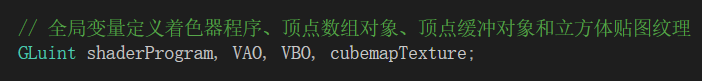


在顶点着色器代码中，使用layout (location = 0) 将顶点属性数据正确绑定到着色器，再通过uniform将旋转矩阵、投影矩阵、视图矩阵传入顶点着色器中，然后定义输出，将顶点的坐标传递给片元着色器。在顶点着色器的主函数中应用旋转矩阵，实现立方体的旋转，并生成正确的图像。

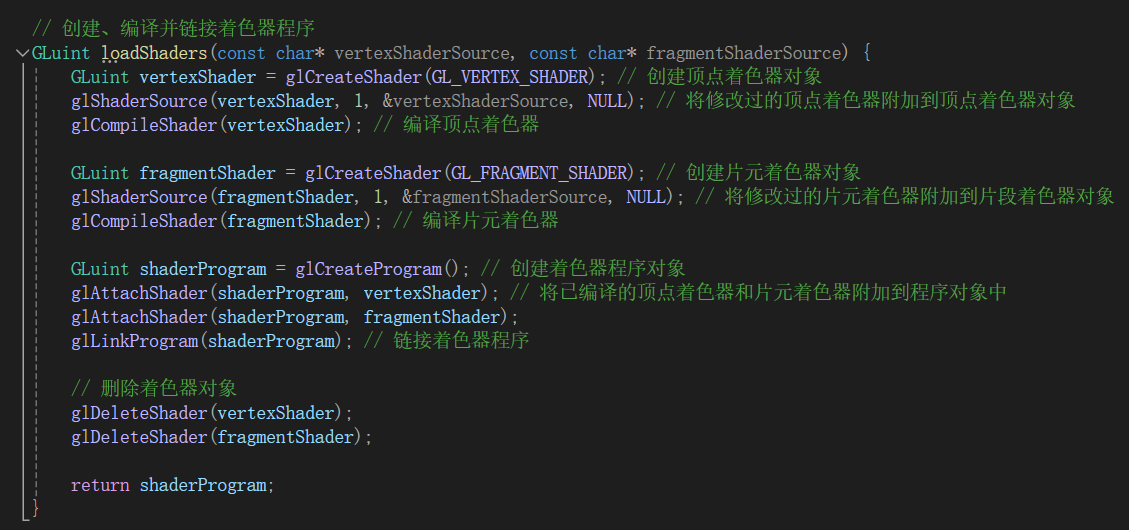


在片元着色器中，定义输入为顶点着色器中的顶点坐标，定义输出为添加纹理后的片元颜色。通过uniform传入经过samplerCub采样后的立方体纹理。在片元着色器的主函数中，通过纹理采样函数texture获取skybox立方体纹理中的颜色，最终作为片元着色器的输出。

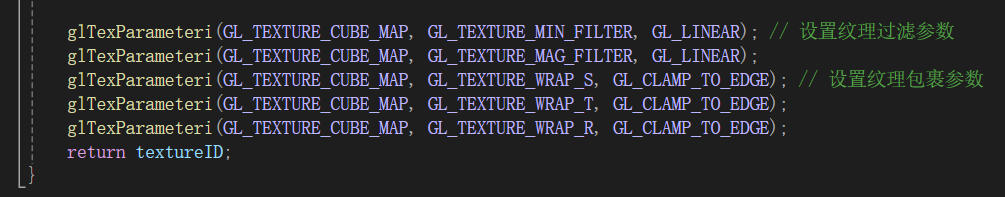
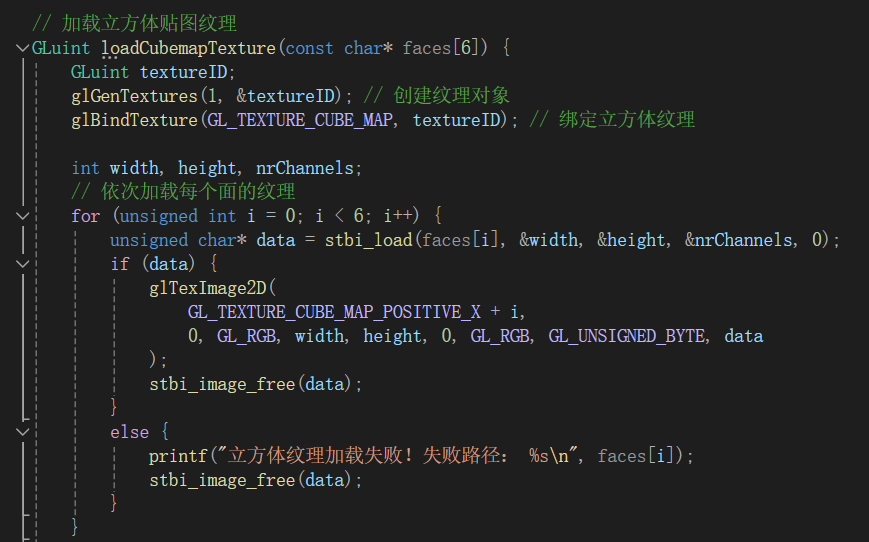
## 创建、编译并链接着色器程序



首先定义需要使用到的全局变量：着色器程序shaderProgram、顶点数组对象VAO、顶点缓冲对象VBO和立方体贴图纹理cubemapTexture。

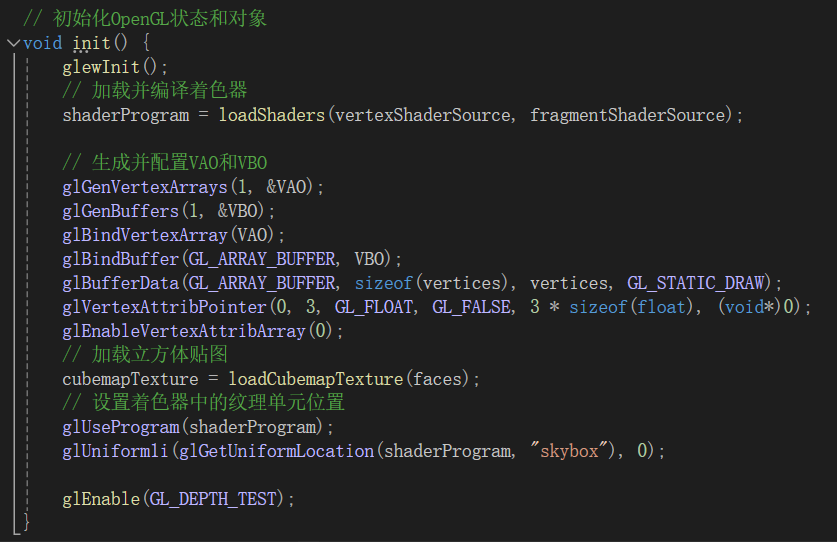


在loadShaders函数中分别创建顶点着色器对象与片元着色器对象，再分别将修改过的顶点着色器与片元着色器附加到对应着色器对象上，然后编译。着色器程序同理，最后链接着色器程序。



在loadCubemapTexture函数中加载立方体贴图纹理，首先创建一个纹理对象并将其与立方体纹理进行绑定。然后使用stb\_image.h文件中的相关函数依次加载每个面的纹理。最后设置纹理过滤参数与纹理包裹参数。

## 初始化OpenGL



在init函数中首先加载并编译先前修改好的着色器，生成并配置VAO和VBO，将faces数组中的立方体贴图加载至全局变量cubemapTexture中备用，最后设置着色器中的纹理单元位置并开启深度测试。

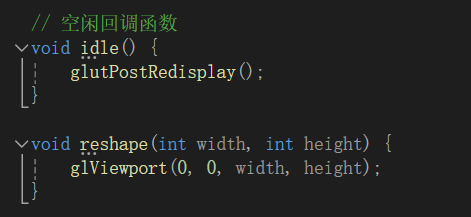
## display函数





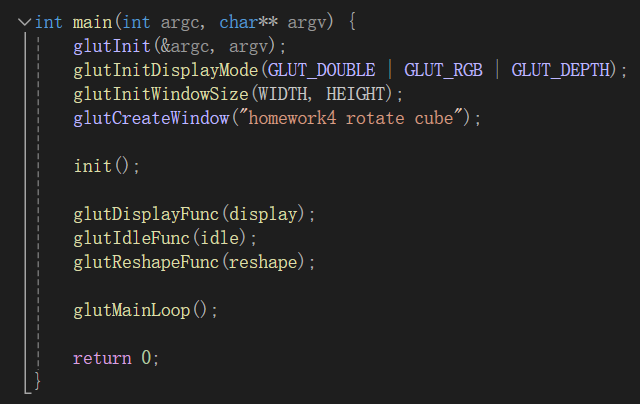
在display函数中计算旋转矩阵、视图矩阵、投影矩阵并分别将其传给顶点着色器。然后激活纹理单元并绑定立方体贴图，通过VAO中的信息绘制立方体。

## 空闲回调函数与reshape函数



空闲回调函数idle调用glutPostRedisplay()以便在空闲时重新绘制窗口内容。reshape函数使用glViewport更新OpenGL渲染视口，确保渲染的内容适应新的窗口大小。

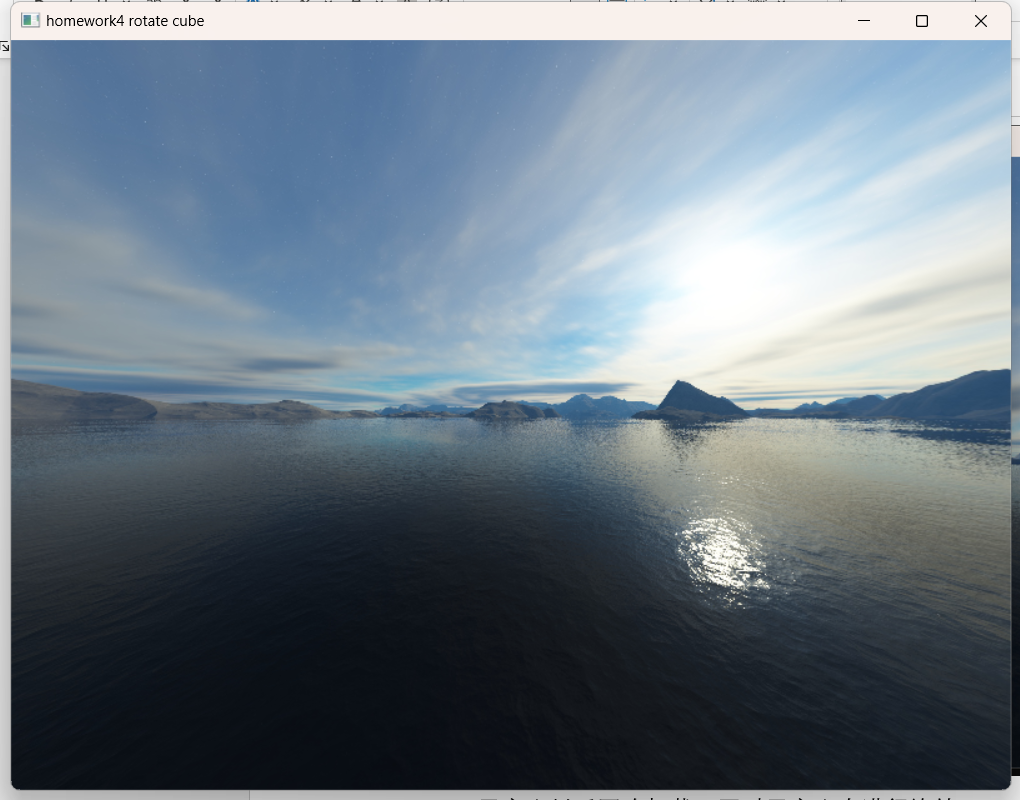
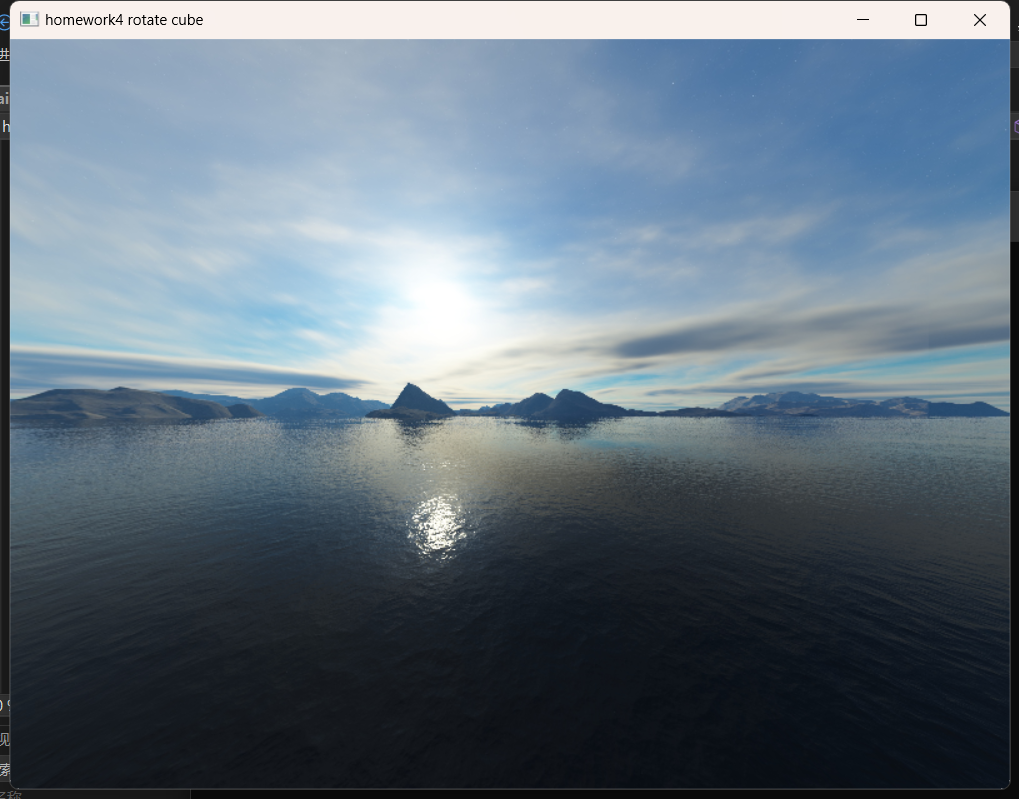
## 主函数



初始化glut、设置显示模式、创建窗口、初始化OpenGL、注册display函数、注册空闲回调函数、注册reshape函数、开始glut主循环。

## 程序运行截图

运行程序，生成窗口效果如下图：



天空盒材质正确加载，同时天空盒在进行旋转。