

实验1、OpenGL初步

姓 名：\_ 徐铭逸\_\_\_\_

学 号：\_22920202202931

学 院：\_\_信息学院\_\_\_\_

专 业：\_\_\_软件工程\_\_\_

年 级：\_\_\_\_2 \_\_\_ \_\_\_

二〇二二年 **三** 月 **九** 日

目录

[Task1：绘制一个实心圆盘 3](#_Toc97738618)

[操作流程： 3](#_Toc97738619)

[关键代码截图： 6](#_Toc97738620)

[运行结果截图： 9](#_Toc97738621)

[Task2：绘制一个奥运五环 10](#_Toc97738622)

[操作流程： 10](#_Toc97738623)

[关键代码截图： 11](#_Toc97738624)

[运行结果截图： 13](#_Toc97738625)

# Task1：绘制一个实心圆盘

## 操作流程：

1. 实验准备：
   1. 在Visual Studio 2022下开发，使用C++ 17标准
   2. 使用glfw作为窗口库，glad作为OpenGL函数库，glm作为矩阵运算库
   3. 采用OpenGL**可编程流水线**绘制
   4. 规划功能：在3D空间中绘制实心圆盘，并且可以自由移动视角来观察该圆盘
2. 实验过程
   1. 通用工具类、函数的实现
      1. **着色器类**  
         可编程流水线中必须使用GLSL着色器程序进行绘制，因此考虑使用一个着色器类封装相关过程，提高可重用性  
         在**shader.cpp和shader.h**中添加着色器类class Shader，实现以下功能：从文件中加载着色器，使用该着色器，向着色器传递uniform变量
      2. **矩阵变换系列函数**  
         可编程流水线中要实现空间变换无法使用OpenGL的glMatrix系列函数，因此考虑使用命名空间trans封装相关过程，提高可重用性  
         在**transform.cpp和transform.h**中添加命名空间namespace trans，再细分出projection、model、view三种变换的命名空间  
         对三种变换分别实现对应功能：各种变换函数、向着色器提交变换等  
         其中矩阵运算使用的是glm库，并且之后在程序中会大量使用glm库的mat4，vec3等类型
      3. **三角网格类**  
         可编程流水线中绘图需要使用VAO(Vertex Array Object，顶点数组对象)，因此考虑使用一个三角网格类封装相关过程，提高可重用性，之后绘图都基于三角网格进行  
         在**mesh.cpp和mesh.h**中添加三角网格类class Mesh，其中存储的是struct Triangle代表三角形的结构体类型，而Triangle中存储的是三个struct Vertex代表顶点数据的结构体类型，目前Vertex中只存储顶点的位置和颜色数据  
         为了实现可编程流水线绘制，大致的流程如下：
         1. 创建VAO, VBO（顶点数组缓冲对象）
         2. 根据着色器输入设置VAO对应的顶点属性
         3. 向VBO中提交对应的顶点数据
         4. 调用glUseProgram使用着色器，调用glDrawArrays绘制

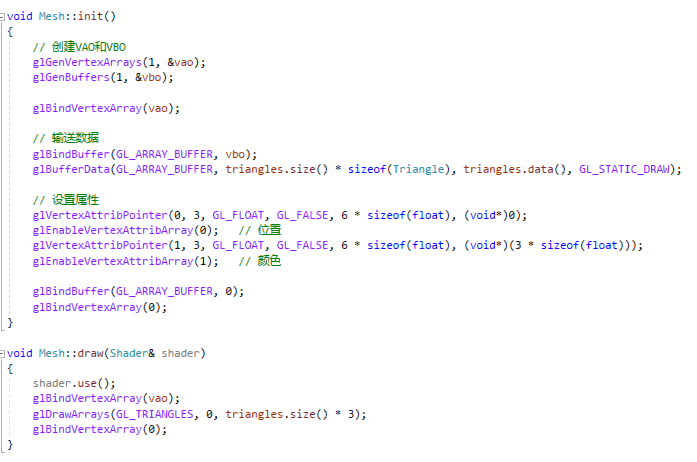
因此Mesh类中存储顶点数据、VAO ID、VBO ID，并且包含一个init()方法用于初始化渲染（1.2.3.步），一个draw()方法用于绘制，一个add()方法用于添加三角形数据等

* 1. 程序基本框架
     1. 在main.cpp中分别实现以下初始化函数
        1. gl\_init()负责初始化glfw窗口和glad库
        2. callback\_init()负责初始化glfw的回调函数
        3. render\_init()负责初始化一些有关于渲染的状态(比如着色器加载、透视投影)
     2. 分别实现以下主要功能函数
        1. Update() 负责进行绘制无关的逻辑更新，比如摄像机坐标
        2. Draw() 负责进行绘制
     3. 在main函数中调用以上初始化函数  
        之后进入一个while**主循环**，退出条件是窗口关闭每次循环调用一次Update()和Draw()，并调用一次glfwPollEvents()处理glfw的回调事件，调用一次glfwSwapBuffers()刷新屏幕双缓冲
  2. Task1功能实现
     1. 扇形的绘制通过多个四边形（三角形）拼接组成 如下图  
        因为需要绘制多次扇形，并且Task2也有对应需求，因此将绘制扇形封装为单个函数  
        void load\_fan(Mesh& mesh, const vec3& center, const vec3& color, float r1, float r2, float s, float t, float step)  
        其中参数意义：mesh代表要加入该扇形的三角网格，center代表扇形的中心坐标，color代表扇形的颜色，r1、r2代表扇形的内、外半径，s、t为起始、结束圆心角（0~2pi），step为每个四边形的圆心角  
        函数内部为一个循环，每次计算出一个四边形（对应的两个三角形）加入到mesh中，计算方式是利用三角函数，具体见代码截图
     2. 之后新建一个函数task1\_init()和一个网格Mesh task1\_mesh  
        在task1\_init()中多次调用load\_fan向task1\_mesh中加入多个颜色和形状不同的扇形  
        之后在render\_init()中调用task1\_init()初始化，并在draw()中调用task1\_mesh的draw()函数完成绘制
  3. 其他琐碎功能
     1. 整个程序的空间变换三个部分的实现：
        1. Projection变换，在render\_init()中调用创建透视投影
        2. View变换，在draw()中调用，对应摄像机的位置和旋转
        3. Model变换，在draw()中对每个要画的东西进行变换，在本程序中我对圆盘使用了一个基于时间的旋转变换

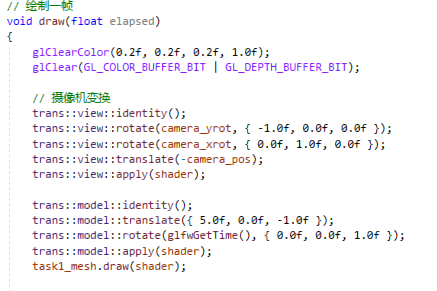
每次变换结束后要向着色器shader提交变换矩阵

* + 1. 着色器的实现  
       本程序使用的着色器较为简单，顶点着色器输入顶点坐标和颜色，向OpenGL输出坐标，向片段着色器输出颜色，片段着色器向OpenGL输出平滑插值的顶点颜色  
       着色器文件在可执行文件目录下的/res/shader/文件夹里

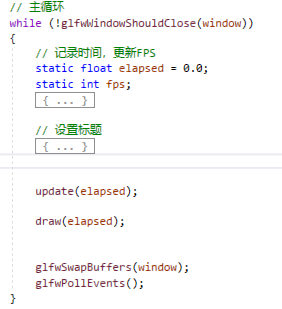
## 关键代码截图：



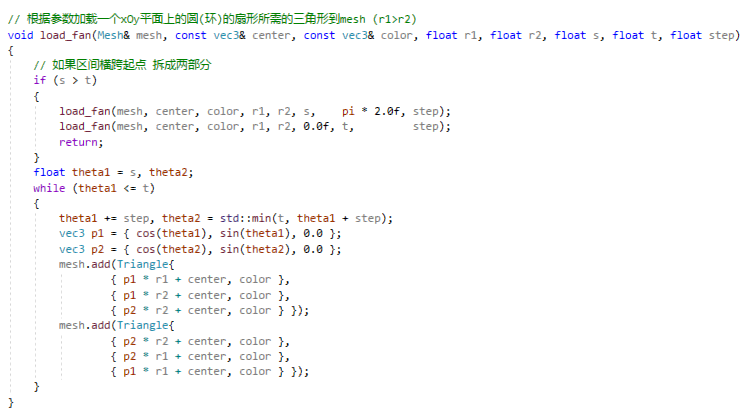
三角网格的初始化和绘制，程序中所有绘制基于三角网格类



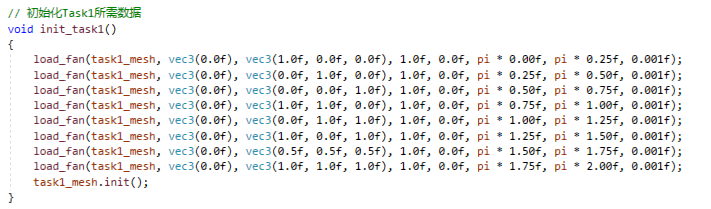
绘制一帧的函数，可以看到trans命名空间里的空间变换函数的使用方法



函数的主循环，其中更新Fps和标题的部分代码折叠

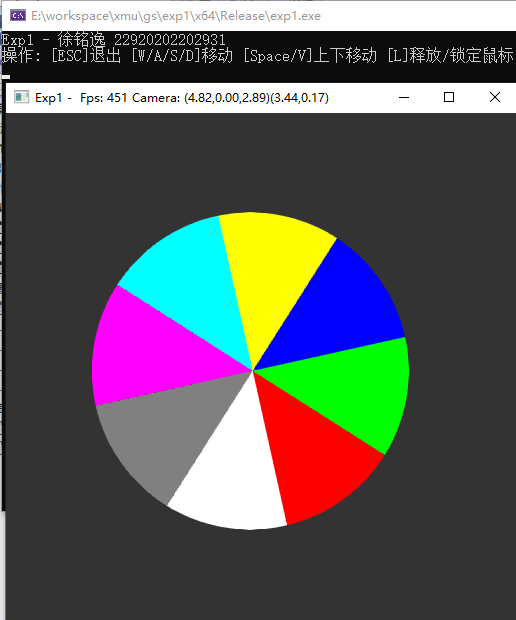


向Mesh中加入扇形的函数，可以看到向Mesh添加三角形的方法

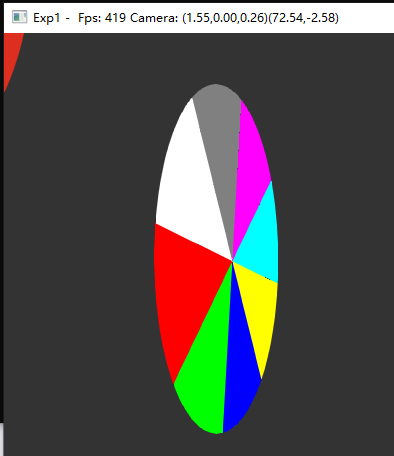


Task 1中扇形的初始化

## 运行结果截图：



正面观察，可以看到在终端中有附带程序的操作指南，在窗口的标题有Fps和摄像机位置、朝向的显示



在侧面观察

# Task2：绘制一个奥运五环

## 操作流程：

1. 实验准备：

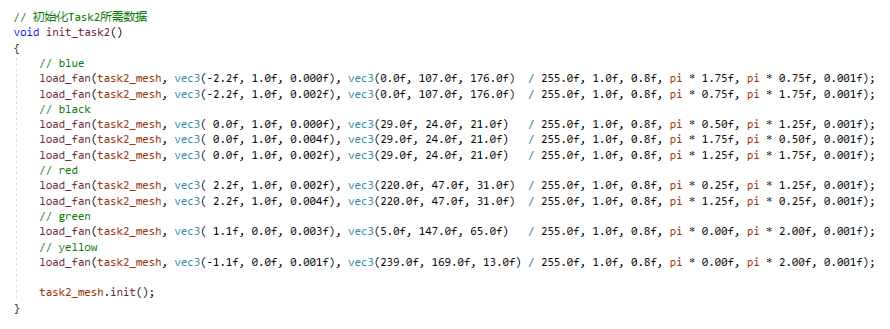
在Task1的程序中实现，使用Task1中绘制扇形的函数

1. 具体实现
   1. 五个圆环位置和颜色不同，只需修改load\_fan中的位置和颜色参数
   2. 五个圆环之间互相穿过，无法直接绘制，给出以下解决方法：
      1. 在2D绘图下可以将圆环拆成几个部分，按正确的顺序绘制
      2. 在3D下关闭深度测试，也可以按照2D的方法绘制
      3. 在3D下开启深度测试，**可以将圆环拆成几个部分后对它们在z轴上进行不同的微小偏移**，使得前后顺序正确
      4. 在3D下开启深度测试，可以将圆环进行小幅度旋转，使其真的在三维空间里环环相扣

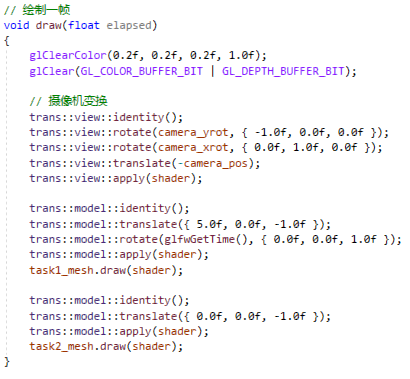
**本程序使用方法iii**具体做法是将左上和右上的环拆成两部分，中上的环拆成三部分，并施加不同的z轴深度绘制，详细参数可参考代码截图

* 1. 和Task1类似，将五环的三角形数据加入task2\_mesh，在task2\_init()中初始化，在draw()中绘制，使用同一个着色器  
     为了防止和task1的圆碰撞，对Model变换矩阵进行平移操作

## 关键代码截图：



Task2中共9个扇形部分的创建

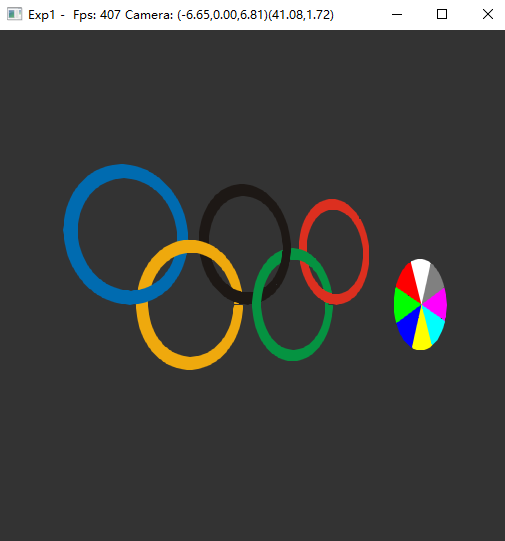


添加Task2后的绘制函数

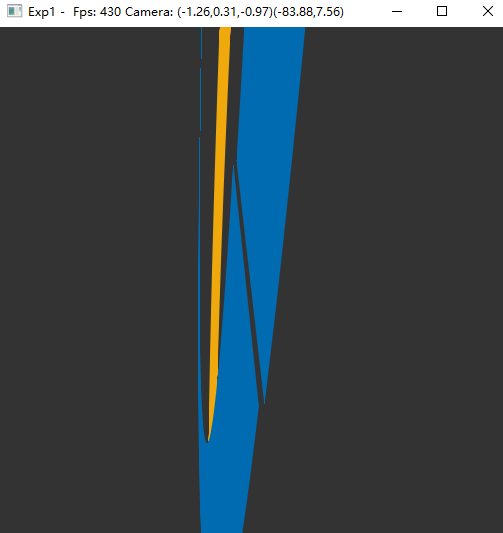
## 运行结果截图：



正面观察



在侧面和Task1的圆盘一起



因为使用偏移z轴的方法而产生的微小接缝