

浙江大学

面向对象程序设计

大 程 序 报 告



大程名称： GeoKurumi——基于 OpenGL 的模型处理程序

姓名： 姚欣岳

学号： 3210102379

电话： 19857162117

指导老师： 李际军

2022~2023 春夏学期 2023 年 5 月 24 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>5</b>
2.1	业务需求 .....	5
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>5</b>
3.1	总体架构设计 .....	5
3.2	类体系设计 .....	5
3.3	主要类设计 .....	5
3.4	源代码文件组织设计 .....	6
3.5	关键功能类及函数设计描述 .....	6
<b>4</b>	<b>新设计类功能说明 .....</b>	<b>7</b>
4.1	总体架构设计 .....	7
4.2	类模块体系设计 .....	8
4.3	数据结构类设计 .....	9
4.4	源代码文件组织设计 .....	11
4.5	重点类及函数设计描述 .....	13
<b>5</b>	<b>部署运行和使用说明 .....</b>	<b>13</b>
5.1	编译安装 .....	13
5.2	运行测试 .....	14
5.3	使用操作 .....	15
5.4	收获感言 .....	16
<b>6</b>	<b>参考文献资料 .....</b>	<b>17</b>

# GeoKurumi 模型处理大程序设计

## 1 大程序简介

### 1.1 选题背景及意义

随着科技的不断发展，数字化设计在各个领域中得到了广泛应用。其中，立体模型的设计在建筑、工程、医学等前沿领域中具有重要的地位。

而在此过程中设计师和工程师经常需要将三维模型的部分性质可视化，以便能更好地理解 and 调整设计方案。本程序处于对此类需求的分析，设计了与模型设计和可视化相关的功能。

### 1.2 目标要求

- **三维模型导入：**  
允许用户导入常见的三维模型格式，例如 obj、STL、pmx 等文件格式以提高程序适用范围。
- **模型显示与用户导航：**  
三维视图，用户可以通过鼠标键盘交互改变视角。
- **模型编辑：**  
设计模型的编辑功能，提供一些组件供用户使用。
- **模型文件导出：**  
将编辑后的模型文件导出，以供用户他用。

### 1.3 术语说明

**GLFW:** GLFW 是一个用于创建 OpenGL 上下文和窗口的跨平台库。它提供了简单的 API，使开发者能够创建窗口、处理用户输入、创建 OpenGL 上下文以及管理 OpenGL 扩展等功能。GLFW 支持多个窗口和多个显示器，并且能够捕获和处理键盘、鼠标和游戏手柄等输入设备的事件。它是一个轻量级的库，易于使用，并且与现代 OpenGL 开发很好地配合使用。

**GLAD:** GLAD 是一个用于加载 OpenGL 函数指针的库。它帮助开发者在运行时动态加载所需的 OpenGL 函数，以便在程序中使用它们。由于不同的操作系统和图形驱动程序可能提供不同的 OpenGL 函数，GLAD 使得在不同平台上运行

OpenGL 代码变得更加简单。

ImGui: ImGui 是一款轻量级图形界面库，允许开发者为他们的应用程序创建简单高效的用户界面，易于集成和高度可定制。

## 2 需求分析

### 2.1 业务需求

本程序实现类似 Libigl 的三维模型可视化增强功能，对网格化的模型进行可视化处理，为模型的分析提供有效工具。

### 2.2 功能需求

- 模型文件的导入
- 模型文件的保存
- 模型坐标变化：平移、旋转、放大缩小、真实感显示
- 模型的可视化增强

### 2.3 数据需求

数据应能与着色器进行交流，包括变换矩阵的输入和输出，具体可参考 doxygen 生成的网页

### 2.4 性能需求

性能上，重复加载纹理可能导致运行时间过慢，数据量上，根据不同用户的硬件性能，在此限制了模型的大小。

## 3 类库已有功能分析

### 3.1 总体架构设计

文件过多，详见 doxygen 生成类图

### 3.2 类体系设计

文件过多，详见 doxygen 生成类图

### 3.3 主要类设计

#### 1. ``igl::viewer::Viewer``:

这个类实现了一个简单的交互式三维模型查看器，可以显示和操作三模型。它提供了用于设置渲染参数、处理用户输入和显示模型的功能。

#### 2. ``igl::read*`` 和 ``igl::write*`` 系列函数:

`libigl` 提供了一系列用于读取和写入各种三维模型文件格式的函数。这些函数用于加载和保存三维模型数据。

#### 3. ``igl::opengl::glfw::Viewer``:

这个类是一个基于 `GLFW` 的交互式三维模型查看器，提供了更多的功能和控制选项，比如相机控制、材质设置和光照等。

#### 4. ``igl::Mesh``:

这个类表示一个三角网格，包含顶点坐标、拓扑关系和边界信息。它提供了一些用于处理和修改三角网格的函数，比如平滑、细分和剖分等。

#### 5. ``igl::bfs`` 和 ``igl::dfs``:

这些函数实现了广度优先搜索（BFS）和深度优先搜索（DFS）算法，用于遍历三角网格的拓扑结构。

#### 6. ``igl::collapse_edge`` 和 ``igl::flip_edge``:

这些函数用于修改三角网格的拓扑结构。``igl::collapse_edge`` 可以折叠一个边，合并两个相邻的三角形，而 ``igl::flip_edge`` 可以翻转一个边，改变三角形的拓扑关系。

#### 7. ``igl::isoline`` 和 ``igl::slice``:

这些函数用于提取等值线和切片，用于可视化三维数据。

### 3.4 源代码文件组织设计

文件过多，详见 `doxygen` 生成类图

### 3.5 关键功能类及函数设计描述

参见 `doxygen` 生成类图









## 4 新设计类功能说明

### 3.5 总体架构设计

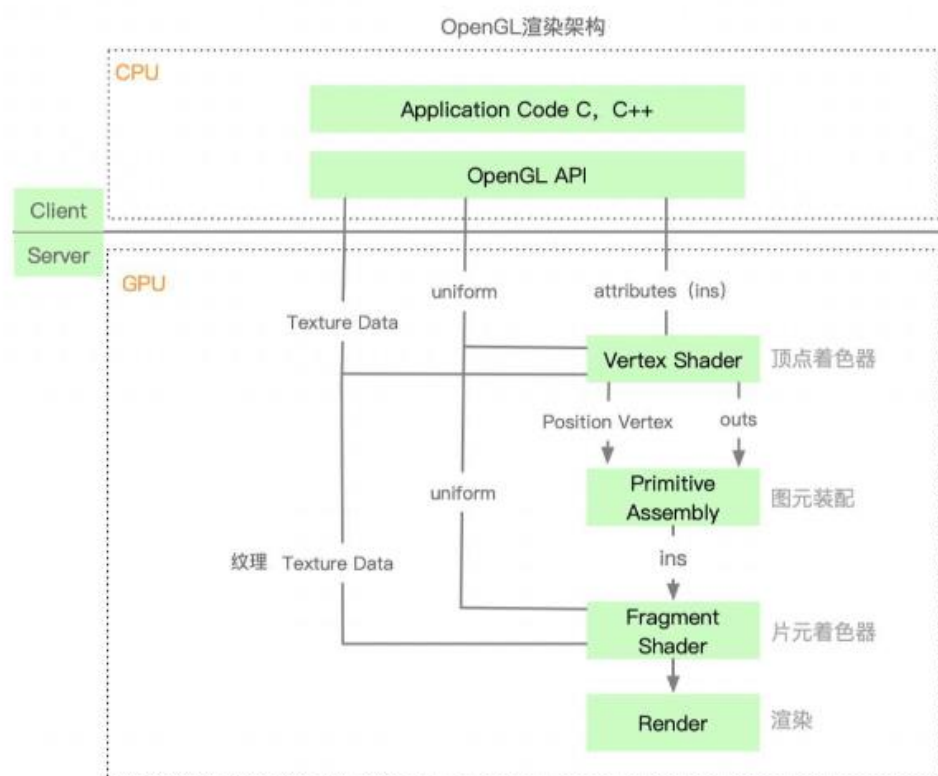
设计类：

#### Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

 Camera	
 Mesh	
 Model	
 Shader	
 stbi_io_callbacks	
 switcher	
 Texture	
 Vertex	

设计架构：基于 OpenGL 渲染架构



设计内容包括：着色器，用户图形交互界面，文件交互模块，功能实现模块等。

### 3.6 类模块体系设计

由以下类构成

#### Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<b>C</b> Camera	
<b>C</b> Mesh	
<b>C</b> Model	
<b>C</b> Shader	
<b>C</b> stbi_io_callbacks	
<b>C</b> switcher	
<b>C</b> Texture	
<b>C</b> Vertex	



## 3.7 数据结构类设计

开关类：用于总体控制各个窗口以及鼠标键盘输入

### switcher Class Reference

#### Public Attributes

bool	<b>firstMouse</b>
bool	<b>mouse_enable</b>
bool	<b>show_work_window</b>
bool	<b>show_another_window</b>
bool	<b>show_hint</b>
bool	<b>blinn</b>
bool	<b>blinnKeyPressed</b>
bool	<b>key_c</b>
bool	<b>save</b>

Mesh 类： 将模型网格化

### Mesh Class Reference

#### Public Member Functions

<b>Mesh</b>	(vector< <b>Vertex</b> > vertices, vector< unsigned int > indices, vector< <b>Texture</b> > textures)
void	<b>Draw</b> ( <b>Shader</b> &shader)

#### Public Attributes

vector< <b>Vertex</b> >	<b>vertices</b>
vector< unsigned int >	<b>indices</b>
vector< <b>Texture</b> >	<b>textures</b>
unsigned int	<b>VAO</b>

The documentation for this class was generated from the following file:

- **mesh.h**

Model 类：用于从 Assimp 库中读取模型数据并将数据转换为着色器能理解的结构并着色。

## Model Class Reference

### Public Member Functions

<b>Model</b> (string const &path, string const &fname, bool load=false)
void <b>Draw</b> ( <b>Shader</b> &shader)
void <b>loadModel</b> (string const &path, string const fname)

### Public Attributes

vector< <b>Texture</b> > <b>textures_loaded</b>
vector< <b>Mesh</b> > <b>meshes</b>
string <b>directory</b>
bool <b>gammaCorrection</b>
bool <b>isLoading</b>

The documentation for this class was generated from the following file:

- **Model.h**

**Camera 类：**处理摄像机坐标，视角方向和缩放，控制摄像机移动和鼠标键盘输入等

## Camera Class Reference

### Public Member Functions

<b>Camera</b> (glm::vec3 position=glm::vec3(0.0f, 0.0f, 0.0f), glm::vec3 up=glm::vec3(0.0f, 1.0f, 0.0f), float yaw=-90.0f, float pitch=0.0f)
<b>Camera</b> (float posX, float posY, float posZ, float upX, float upY, float upZ, float yaw, float pitch)
glm::mat4 <b>GetViewMatrix</b> ()
void <b>ProcessKeyboard</b> (Camera_Movement direction, float deltaTime)
void <b>ProcessMouseMovement</b> (float xoffset, float yoffset, GLboolean constrainPitch=true)
void <b>ProcessMouseScroll</b> (float yoffset)

### Public Attributes

glm::vec3 <b>Position</b>
glm::vec3 <b>Front</b>
glm::vec3 <b>Up</b>
glm::vec3 <b>Right</b>
glm::vec3 <b>WorldUp</b>
float <b>Yaw</b>
float <b>Pitch</b>
float <b>MovementSpeed</b>
float <b>MouseSensitivity</b>
float <b>Zoom</b>

The documentation for this class was generated from the following files:

- **camera.h**
- camera.cpp

Shader 类：将数据与着色器绑定

## Shader Class Reference

### Public Member Functions

<b>Shader</b> (const char *vertexPath, const char *fragmentPath)
void <b>use</b> () const
void <b>setBool</b> (const std::string &name, bool value) const
void <b>setInt</b> (const std::string &name, int value) const
void <b>setFloat</b> (const std::string &name, float value) const
void <b>setVec2</b> (const std::string &name, const glm::vec2 &value) const
void <b>setVec2</b> (const std::string &name, float x, float y) const
void <b>setVec3</b> (const std::string &name, const glm::vec3 &value) const
void <b>setVec3</b> (const std::string &name, float x, float y, float z) const
void <b>setVec4</b> (const std::string &name, const glm::vec4 &value) const
void <b>setVec4</b> (const std::string &name, float x, float y, float z, float w) const
void <b>setMat2</b> (const std::string &name, const glm::mat2 &mat) const
void <b>setMat3</b> (const std::string &name, const glm::mat3 &mat) const
void <b>setMat4</b> (const std::string &name, const glm::mat4 &mat) const

### Public Attributes

unsigned int <b>ID</b>
------------------------

The documentation for this class was generated from the following file:

- [shader.h](#)

## 3.8 源代码文件组织设计

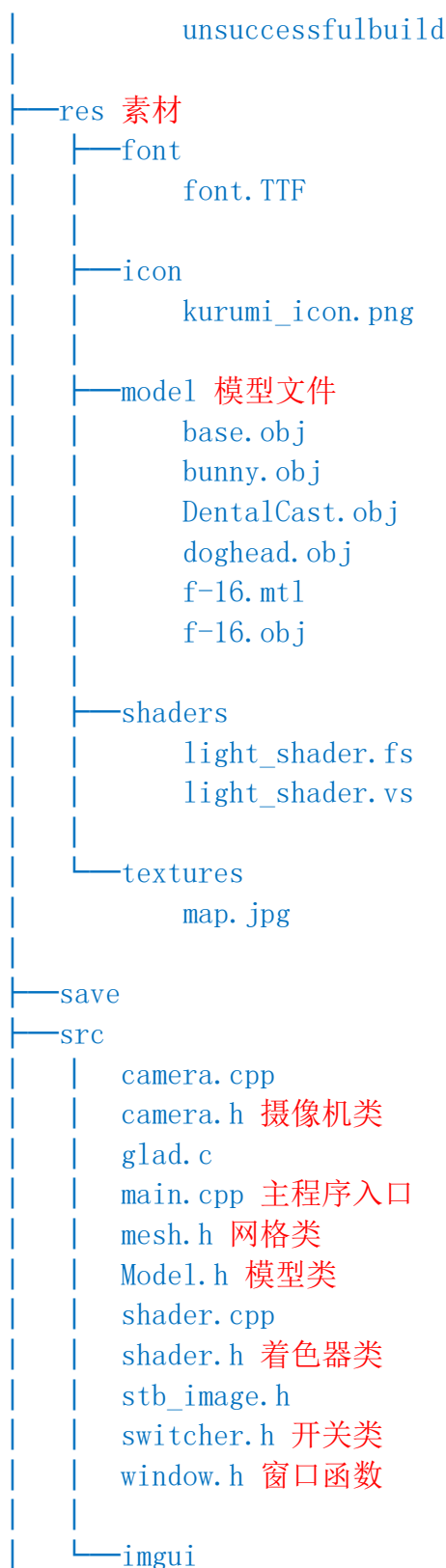
<文件目录结构>

1) 文件函数结构

```

├── Debug
│   │   vc143.idb
│   │   vc143.pdb
│   └── GeoKurumi.tlog
│       CL.command.1.tlog
│       GeoKurumi.lastbuildstate

```



## 2) 多文件构成机制

项目文件较多，可以通过 doxygen 查看具体文件。

主要机制：将类有关的文件分为头文件.h 与实现文件.cpp。

#define 保护由新版的 #pragma once 替代。

外部变量的使用：使用了部分窗口参数作为全局变量，其余均被封装在各自文件

中。

## 3.9 重点类及函数设计描述

### Camera Class Reference

#### Public Member Functions

<b>Camera</b> (glm::vec3 position=glm::vec3(0.0f, 0.0f, 0.0f), glm::vec3 up=glm::vec3(0.0f, 1.0f, 0.0f), float yaw=-90.0f, float pitch=0.0f)
<b>Camera</b> (float posX, float posY, float posZ, float upX, float upY, float upZ, float yaw, float pitch)
glm::mat4 <b>GetViewMatrix</b> ()
void <b>ProcessKeyboard</b> (Camera_Movement direction, float deltaTime)
void <b>ProcessMouseMovement</b> (float xoffset, float yoffset, GLboolean constrainPitch=true)
void <b>ProcessMouseScroll</b> (float yoffset)

#### Public Attributes

glm::vec3 <b>Position</b>
glm::vec3 <b>Front</b>
glm::vec3 <b>Up</b>
glm::vec3 <b>Right</b>
glm::vec3 <b>WorldUp</b>
float <b>Yaw</b>
float <b>Pitch</b>
float <b>MovementSpeed</b>
float <b>MouseSensitivity</b>
float <b>Zoom</b>

The documentation for this class was generated from the following files:

- [camera.h](#)
- [camera.cpp](#)

摄像机类：控制用户视角

函数原型：以 ProcessKeyboard 为例

函数参数：枚举类 Camera\_Movement direction 浮点数 float deltaTime

函数功能：处理键盘输入的 WASD, ctrl, space, 转换为摄像机坐标的变换

返回值：void

重要局部变量：无

算法步骤：

- glfw 传入移动类型：枚举类 direction 中的各类 front, back, right 等方向
- 将坐标的变换输出到渲染前一步的 view 变换矩阵

## 4 部署运行和使用说明

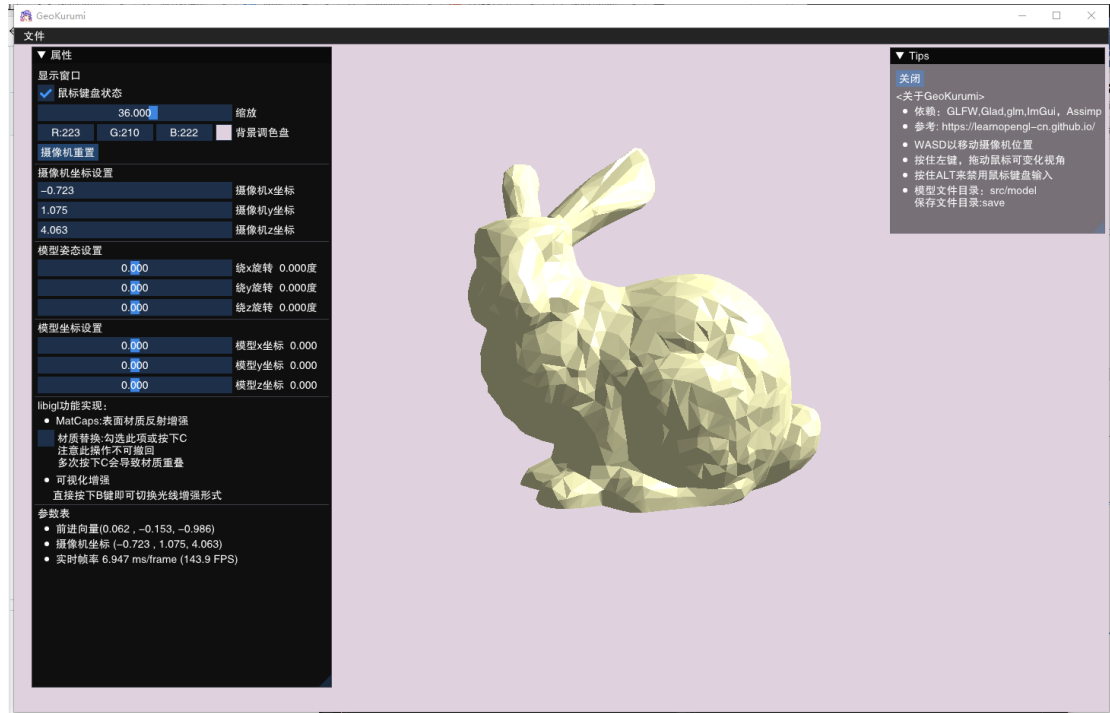
### 4.1 编译安装

本程序提供了完整的工程文件，通过 Visual Studio 后编译可直接生成可执行文件。生成的可执行文件须放在 .vcxproj 相同目录下

## 4.2 运行测试

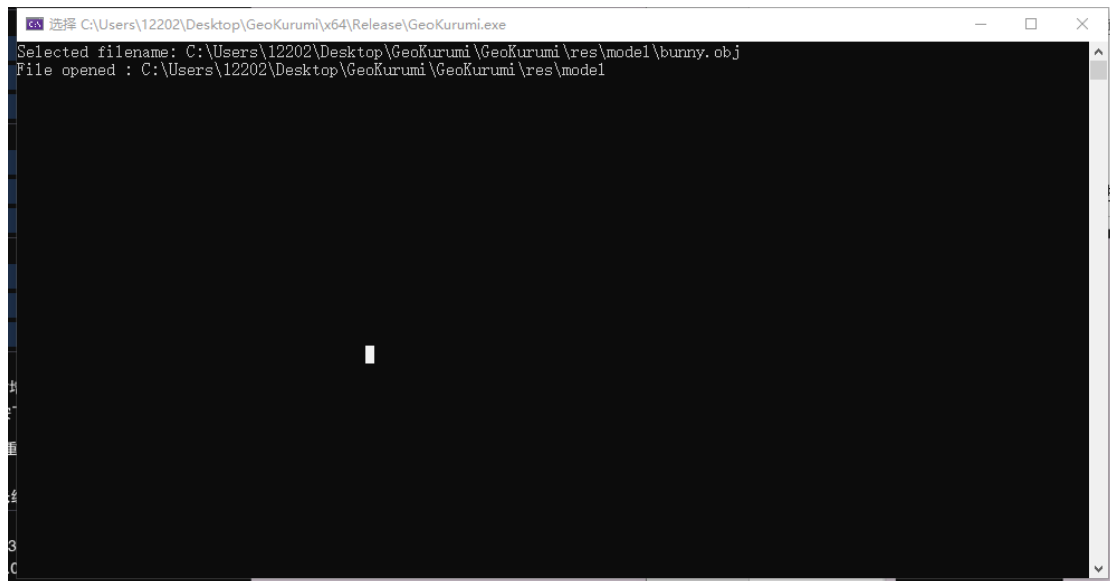
<选取测试阶段典型的案例，说明如何设计测试数据，发现和定位错误的，测试结果可以含有屏幕截图。>

- 通过 ImGui 实时渲染功能进行测试，测试数据主要依据模型文件。



例如从左侧滑块定位出问题，实时打印在左侧框中。

- 使用控制台输出错误信息：

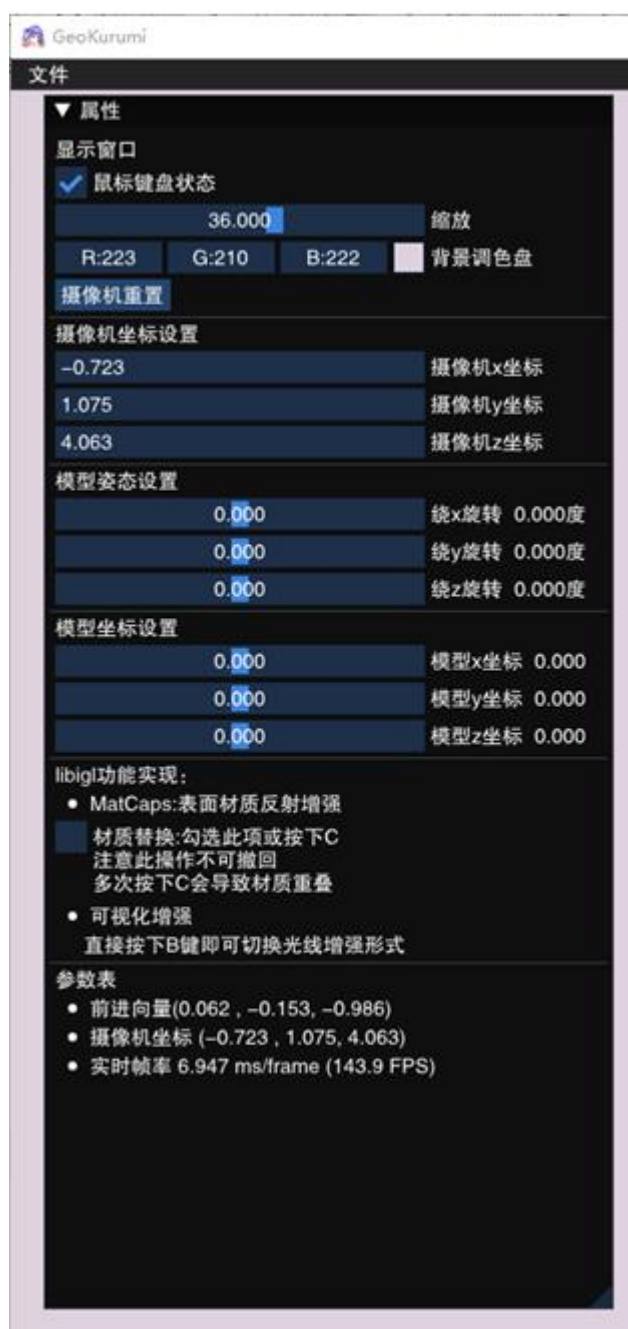


## 4.3 使用操作

使用操作写在右侧边框中



左侧栏中的每一步都有对应中文解释



## 4.4 收获感言

开发过程中，利用了 OOP 的封装、继承、多态的特性，体会到了程序开发的乐趣和困难。

由于时间有限，实现的功能有限，但使我对图形学产生了浓厚的兴趣，收获到了开发的经验和 debug 的教训。



## 5 参考文献资料

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