**《信息可视化》课程实验报告**

212138 卓旭

本课程实验包括两个Project：①显示球面和正方体表面；②体绘制Ray Casting算法实现。

实验使用C++和OpenGL实现。项目需要安装的第三方依赖库有：

·glfw3：对OpenGL的操作系统特定操作进行了封装，便于与GL上下文交互

·glad：对OpenGL的显卡驱动特定操作进行了封装，便于调用GL系列相关函数

·glm：提供了向量、矩阵类和相关运算，以及图形学常用矩阵的生成

·rapidjson：增强实验Project 2的可配置性，借助该库解析JSON配置文件。

C++的依赖安装配置较为复杂，但微软提供的包管理工具vcpkg给我们提供了极大的便利。vcpkg自身的安装配置过程此处不述。在安装好vcpkg后，执行vcpkg install glfw3 glad glm rapidjson命令即可安装所有依赖。

**一、自己设计的OpenGL辅助框架：ReNow的介绍**

在正式介绍Project的具体细节前，笔者将先介绍在两个Project中广泛使用的一个自己设计的OpenGL辅助框架：ReNow。该框架的雏形在本人本科的《计算机图形学》课程（东南大学，学号09017227，杨武老师）实验中成型（代码地址：[github.com/z0gSh1u/typed-webgl](https://github.com/z0gSh1u/typed-webgl)）。其原先使用TypeScript语言开发，工作在WebGL下。借助本课程实验的契机，我开发了一个更完善的工作在C++和OpenGL下的版本，在framework目录下。

ReNow的核心思想是使用ReNowHelper类接管、包装OpenGL的一些操作。包括Program的创建、Shader的创建、Program的切换、2D材质的创建、VAO与VBO的创建等。

ReNow最关键的功能是提供prepareAttributes和prepareUniforms两个函数，以低代码量与着色器进行交互，将数据传送到顶点着色器或片元着色器的Attribute或Uniform型变量上。使用例子如下：

|  |
| --- |
| float vtBack[nPoints \* 2] = {0, 0, 1, 0, 1, 1, 0, 1}; // texture coords  GL\_OBJECT\_ID vtBackBuf = helper.createVBO();  helper.prepareAttributes(vector<APrepInfo>{   {vtBackBuf, vtBack, nPoints \* 2, "aTexCoord", 2, GL\_FLOAT},  });  // -------------------------------- mat4 perspectiveMat = glm::perspective(radians(fovy), aspect, near, far);  helper.prepareUniforms(vector<UPrepInfo>{   {"uPerspectiveMatrix", perspectiveMat, "Matrix4fv"},   {"uColor", normalizeRGBColor(RGBColor(29, 156, 215)), "4fv"},  }); |

本例子上半部分表示：借助vtBackBuf这个Buffer，将vtBack数组的内容，共nPoints \* 2个，传递到着色器名为aTexCoord的Attribute上，每2个数据为一组，类型为GL\_FLOAT。本例子下半部分表示，将着色器名为uPerspectiveMatrix的uniform的值，赋为perspectiveMat这个矩阵，内部使用的方法为glUniformMatrix4fv；uColor同理。通过这样的封装，减少了与着色器交互的代码量，代码更清晰，避免了大量的获取着色器变量位置以及编写通常是默认值的参数的工作。

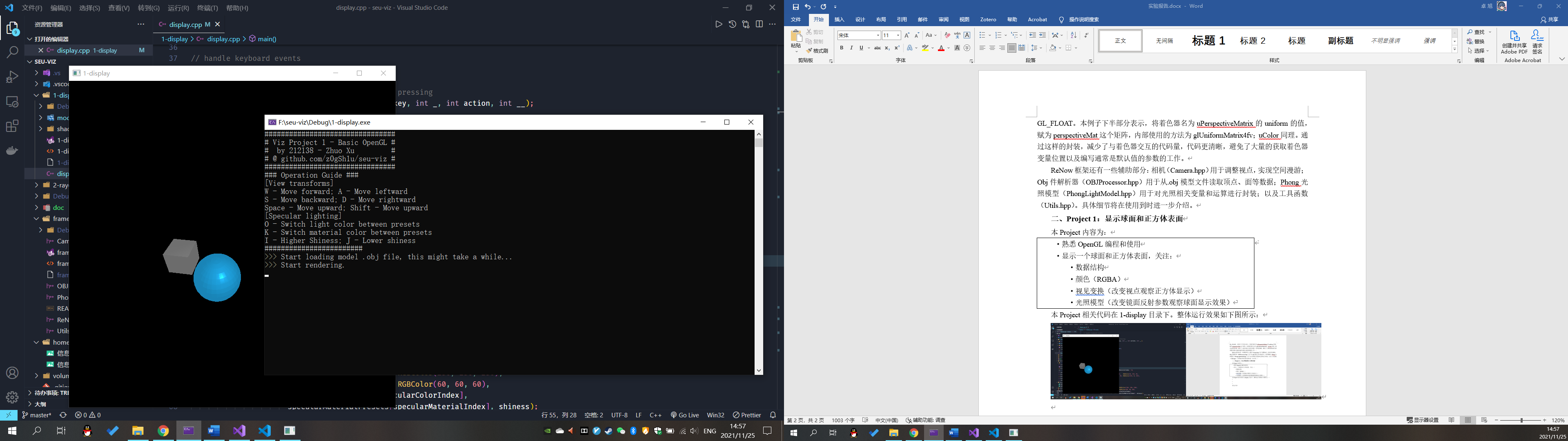
ReNow框架还有一些辅助部分：相机（Camera.hpp）用于调整视点，实现空间漫游；Obj件解析器（OBJProcessor.hpp）用于从.obj模型文件读取顶点、面等数据；Phong光照模型（PhongLightModel.hpp）用于对光照相关变量和运算进行封装；以及工具函数（Utils.hpp）。具体细节将在使用到时进一步介绍。

**二、Project 1：显示球面和正方体表面**

本Project内容为：

|  |
| --- |
| ·熟悉OpenGL编程和使用  ·显示一个球面和正方体表面，关注：  ·数据结构  ·颜色（RGBA）  ·视见变换（改变视点观察正方体显示）  ·光照模型（改变镜面反射参数观察球面显示效果） |

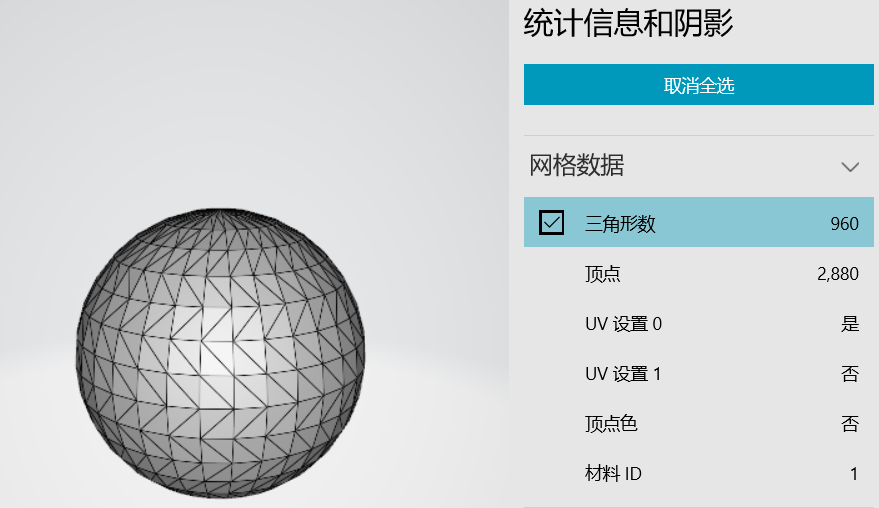
本Project相关代码在1-display目录下。整体运行效果如下图所示。左侧为球面和正方体表面的显示效果，可见颜色和光照的作用效果。右侧为操作说明，可通过WASD-Space-Shift键进行三维空间漫游，通过鼠标的移动改变视角；通过OKIJ键调整光照Specular（镜面）分量。整体完整实现了Project的要求。



首先介绍物体的**数据结构**。球和方体均使用3D建模软件Blender（[www.blender.org](http://www.blender.org)）构建，导出为.obj格式的模型文件。其中所有面片都转换为三角形，点坐标均做了向[-1, 1]的归一化，以便于绘制。模型文件在1-display/model目录下。这样，即可获得顶点坐标、法向量、面片等。解析工作由framework/OBJProcessor.hpp逐行读取完成。关于.obj文件的具体格式组成，可参考维基百科：[en.wikipedia.org/wiki/Wavefront\_.obj\_file](https://en.wikipedia.org/wiki/Wavefront_.obj_file)。此处做一些基本介绍，以下为从1-display/model/UnitCube.obj中截取的一段和相应解释：

|  |  |
| --- | --- |
| **示例行** | **解释** |
| v 1.000000 -1.000000 -1.000000 | 定义一个顶点，(*x*,*y*,*z*) = (1,-1,-1) |
| vt 0.625000 0.250000 | 定义一个材质贴图采样点，(*x*,*y*) = (0.625, 0.25) |
| vn 0.0000 -1.0000 0.0000 | 定义一个法向量，(*x*,*y*,*z*) = (0,-1,0) |
| f 4/12/6 6/8/6 8/6/6 | 定义一个面（斜杠首元），由4、6、8号顶点（v）组成；  定义一个材质贴图面（斜杠中间元），将12、8、6号采样点（vt）决定的图块贴到定义的面上；  声明面的法向量（斜杠末元）是6号vn决定的。  注意.obj中编号从1开始。 |

本Project中图形填充纯色，不需要材质贴图相关信息。使用Windows 10自带的3D浏览器打开UnitCube.obj文件，即可清晰地看到各三角形面片（mesh网格）：



下面介绍绘制时**颜色（RGBA）**的设定。由于进行纯色绘制，故将颜色在着色器中使用Uniform型变量uColor存储。正方体的颜色为灰色：(R, G, B, A) = (127, 127, 127, 1.0)，在向着色器传递时，需要将RGB分量归一化，即除以255，故结果为 (R, G, B, A) = (0.498, 0.498, 0.498, 1.0)。球体的颜色为蓝色：(R, G, B, A) = (29, 156, 215, 1.0)，送入着色器前同样进行归一化。由于无特殊需求，二者的不透明度A均设为1。

在讲解**视见变换**的实现前，首先对顶点着色器中顶点坐标的变换过程进行说明。见1-display/shader/vMain.glsl，与几何相关的部分如下：

|  |
| --- |
| gl\_Position = uPerspectiveMatrix \* uWorldMatrix \* uModelMatrix \* aPosition; |

其中，aPosition为传入的顶点原始坐标；uModelMatrix用于实现对物理自身的变换；uWorldMatrix用于实现世界坐标系的变换，是实现漫游观察的重点；uPerspectiveMatrix用于产生透视投影效果。

在本Project中，我们固定uPerspectiveMatrix为glm::perspective(radians(fovy), aspect, near, far)的运算结果，其中fovy表示y向视野大小，设为90（度）；aspect为视场宽高比，设为窗口宽高比；near和far两个参数用于…由于

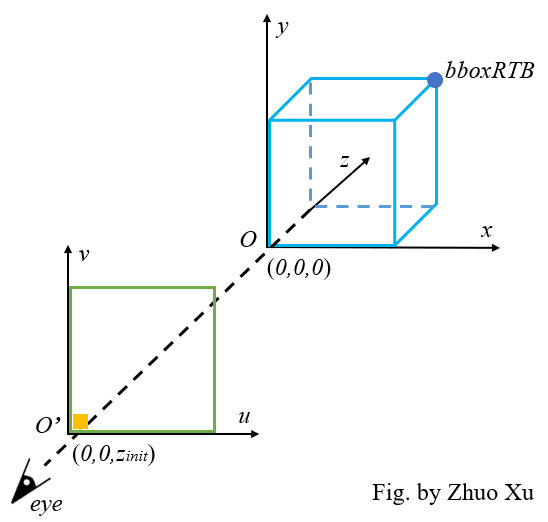
相机的维护

光照

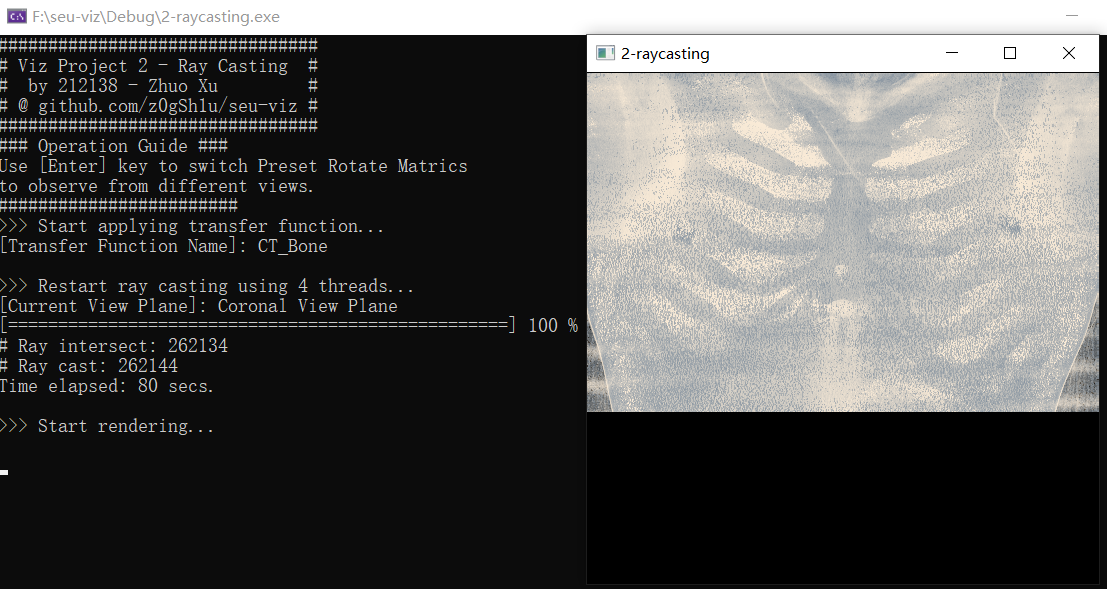
**三、Project 2：体绘制的Ray Casting算法实现**

首先，笔者将结合罗立民老师的课件：Viz7-体积绘制，对体绘制的Ray Casting算法（下面简称RC算法）的主要流程与思路进行说明。

在具体进行体绘制之前，有必要先对本Project中采用的成像几何模型进行说明，以免后续造成误解。如下图所示。**世界坐标系**为右手系*O*-*x*-*y*-*z*，蓝色方体表示体数据的包围盒（Bounding Box），包围盒左前底部角放在世界坐标系*O*点即(0,0,0)处，这样，进一步规定*x*、*y*、*z*轴分别为包围盒左右、上下、前后面的法向量，则只需要使用包围盒右后顶部角的点坐标*bboxRTB*，即可唯一确定该包围盒。**视平面坐标系**为二维坐标系*u*-*O****’***-*v*，*O****’***点落在*z*轴上；*u*在列方向，与*x*轴同向；*v*在行方向，与*y*轴同向；视图生长的方向为*u*、*v*正向。需要注意，该定义下*v*坐标值代表的不是视图第*v*行，而是第*H*-1-*v*行（*H*为视图高），视平面坐标系原点*O****’***放置在世界坐标系下的(0, 0, *zinit*)处。



进一步地，我们规定采用平行光投影方式，即光线如下图所示。那么成像几何可以进一步简化。



相交与出入点判断

传递函数设计

颜色融合

多线程RC算法

结果图绘制

https://www.khronos.org/registry/OpenGL-Refpages/gl4/

**四、其他说明**