

# CG 大作业——盗墓笔记项目报告

## 一、项目简介

### 1. 场景简介

本项目主要构建了两大场景。

场景一是在一片沙漠中，一本金属材质的书在中心旋转，书的四周旋转出现 5 个自身在旋转并且逐渐变大的锈掉的球，球的上面分别写有“金、木、水、火、土”，并且背景逐渐被黑雾笼罩。

场景二是在一个暗室里，地板上有骷髅头和石堆模型，打开手电筒可以看到带光照的效果。按下炸弹释放键，从天而降一个炸弹，碰到石堆以后发出一声巨响，石堆炸开，四散飞走。石堆消失后，出现一个佛像模型，打开手电筒，可以看到模型从粗糙的灰色石质逐渐变为银色至金色，并且变得逐渐光滑精细。

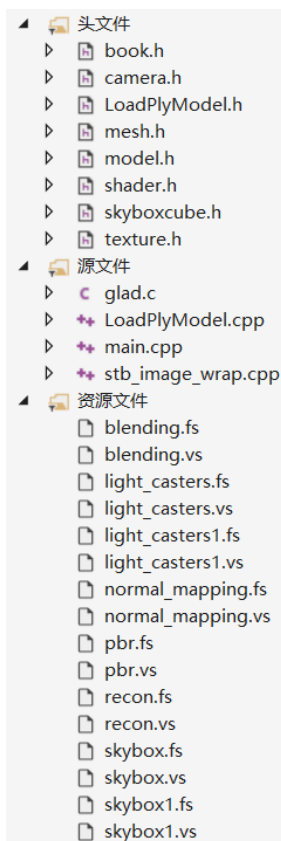
### 2. 交互简介

移动鼠标可以控制视角，滑动滚轮可以放大或缩小。

按键 w、s、a、d 分别控制视点向前、后、左、右移动。

在场景二中，按下 b 键可以释放炸弹，按下空格键可以打开手电筒。

### 3. 代码简介



book.h 绘制书籍，将法线贴图映射到书籍的六个面。

camera.h 控制观察相机，可以调整视角视点等。

loadPlyModel.h、loadPlyModel.cpp 读取 ply 佛像模型文件，处理顶点和面，计算法向量，绘制佛像模型。

mesh.h 将读入的模型进行绑定，加载纹理。

model.h 读取 obj 模型文件，处理顶点、面、法向量、纹理等，绘制模型。

shader.h 编译、链接和管理片段着色器、顶点着色器，并对它们进行错误检测。








skyboxcube.h 绘制天空盒背景。

texture.h 加载纹理和绑定纹理。

glad.c 和 stb\_image\_wrap 用于配置环境，便于能够找到相应的头文件。

main.cpp 为主函数，用于窗口创建、键盘处理、绘制场景、多线程管理等。

#### 4. 辅助库和参考资料

 <b>assimp</b>	assimp 库读取并加载 obj 模型。
 <b>glad</b>	glad 库可以简化在运行时获取函数地址并将其保存在一个函数指针中供以后使用的过程，并且依赖了 KHR 库。
 <b>GLFW</b>	GLFW 为基本库，用于图形、窗口、渲染等等。
 <b>glm</b>	glm 库进行向量运算，方便数学计算。
 <b>irrklang</b>	Irrklang 库用于播放音频。
 <b>KHR</b>	
 <b>stb_image.h</b>	stb_image.h 用于读取图片。

参考教程：

- <https://learnopengl-cn.github.io/> (<https://learnopengl.com/>)
- <http://ogldev.org/>

#### 5. 大作业要求完成情况

- 设计实现一个片头动画：要在一本金属材质的书封上出现《盗墓笔记》的 bump mapping 视效和动画，整个场景要有合适的背景，有灯光变换，然后这本书缓慢消失在迷雾中。

采用了 bump mapping 将书封映射到书的六个面，设置材质和光照，产生凹凸效果，书籍在中心旋转。采用了 Physically Based Rendering 绘制了五个生锈的铁球，表面有“金、木、水、火、土”字样。书籍和铁球进行旋转，铁球在逐渐变大。物体运动时，能够看到因为位置改变带来的光照效果改变。

整个场景在一片沙漠构成的天空盒中，随着时间的推移背景逐渐被黑雾笼罩，然后视线迅速往书籍里拉近，切换到第二个场景。

- 迷雾逐渐散去，出现一个暗室，键盘敲击空格键，会有一个手电朝前照射，看到一个石堆。此时，再按“B”键，一个炸弹向这个石堆飞去，碰撞产生爆炸，手电关闭。

暗室同样采用天空盒技术，形成四周墙壁，并且出现骷髅头和石堆。按下空格打开手电筒，可以看到骷髅头和石堆的光照效果。

按下 B 键，从天而降一个炸弹，石堆爆炸，发出一声巨响，手电筒关闭，石头四散飞走。

- 石块散去，剩下一个 Buddha 样子的粗糙模型，材质起初是粗糙的灰色石质，但当手电重新打开照亮模型，Buddha 模型开始变得越来越光滑精细，材质也逐渐转变为银色最后到金色。算法效率问题要解决。

石块散去，出现佛像模型，打开手电筒，佛像从粗糙的灰色石质变得越来越光滑精细，材质也逐渐转变为银色最后到金色。采用了多线程提升算法效率。还尝试了一下边塌陷方法来减少顶点，但是感觉加上边塌陷效率提升并不明显，便没有加上边塌陷方法。

- 拓展尝试

加上了爆炸音效，提供了灵活的鼠标、键盘控制摄像机功能，以及尝试了 Physically Based Rendering 技术和边塌陷方法。

## 二、 具体技术方案

### 1. 光照计算

采用冯氏光照模型，一个物体的光照效果需要考虑环境光照、漫反射光照和镜面光照。

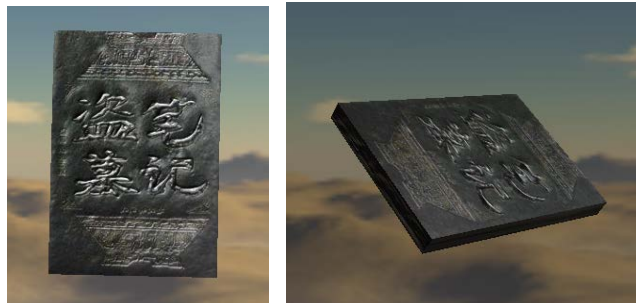


环境光照：简单设置一个光照颜色，乘以一个很小的常量环境因子，再乘以物体的颜色，得到最后的颜色结果。

漫反射光照：计算出顶点法向量，根据物体位置、光源位置、方向等等计算出漫反射颜色。

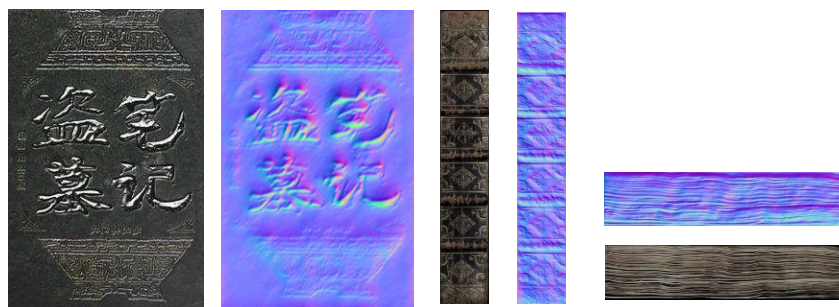
镜面光照：考虑观察者的位置，计算出反射向量，给出一个高光效果。

### 2. 书籍凹凸纹理贴图



为了加强真实感的绘制，不能单纯地使用一个相同的法向量进行纹理贴图，这样看上去会是一个光滑的平面。将表面分成很多小片段，每一片段使用自己的法线，结合光照可以产生凹凸的效果。

利用 Crazybump 产生书籍六个面的法线贴图：



从法线贴图中获得法线，重新利用之前的冯氏光照模型，计算出对应的颜色，

并且映射到书的六个面上。

### 3. 铁球的基于物理的渲染

基于物理原理的渲染能够比冯氏光照模型更加真实，需要考虑三个方面：基于微平面的表面模型、能量守恒、应用基于物理的 BRDF，考虑相应的物理实际绘制生锈的铁球。

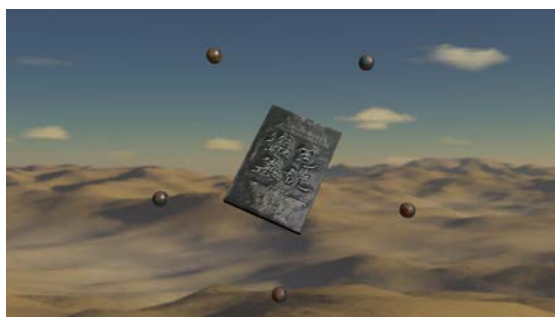
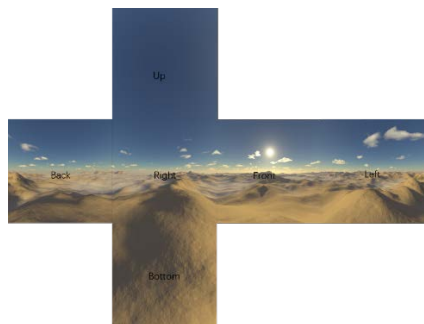


### 4. 动画

动画的实现比较简单，因为没能够实现骨架动画，现在只能将整个物体做平移、缩放、旋转这样的基础动画，主要采用了 glm 库中的矩阵变换函数 `translate`、`scale`、`rotate`。

### 5. 天空盒技术

因为不需要场景的其他东西，所以采用天空盒技术可以很好的满足想要的效果，传给渲染器相应的顶点参数，将沙漠的六个面贴到场景的六个面形成沙漠背景。结合之前的模型构成第一个场景。



### 6. 雾化效果

没能实现物体模型的淡入淡出效果，所以考虑怎么能够流畅转到第二个黑暗的场景，最后决定采用显示背景产生黑雾，然后摄像机迅速往前拉近产生全黑情景。

因此为天空盒增加逐渐产生黑色迷雾的效果，主要算出背景离摄像机的距离，然后随着时间 `time` 的递增，`Fog` 值由距离与时间共同得出，并确保 `Fog` 值处于 0 到 1 之间，然后算出背景纹理的 `color` 值，求得 `Fog` 与 `color` 的线性插值，从而达到背景逐渐产生黑雾的效果。

### 7. 摄像机控制

摄像机的控制比较简单，只需要设置摄像机的位置和方向，再利用 glm 提供产

生的 lookat 矩阵就可以获得观察矩阵，进而传给着色器使用即可。

为了使视角切换更加灵活，加入一些移动、缩放的交互设计，需要检测到相应的键盘、鼠标输入，调整摄像机位置、方向等参数即可。

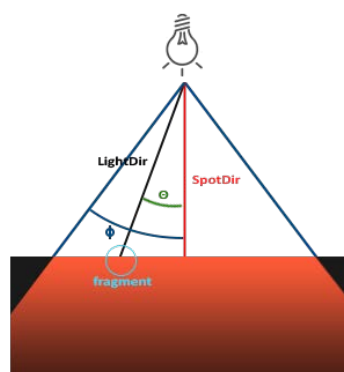
## 8. 使用库的模型导入

构建 model 类读取模型文件，利用 assimp 提供好的接口得到顶点、纹理、面等等模型信息，存储起来用于绘制。构建 mesh 类处理获得的模型文件，解析顶点、面、纹理信息，用相应的着色器进行绘制。可以得到如下场景结果，此时没有打开光照，但是已经能看到模型轮廓。



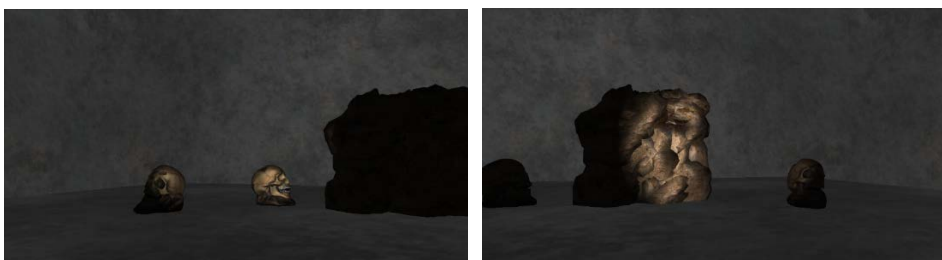
## 9. 手电筒光照效果

聚光是位于环境中某个位置的光源，它只朝一个特定方向而不是所有方向照射光线。这样的结果就是只有在聚光方向的特定半径内的物体才会被照亮，其它的物体都会保持黑暗。这里需要的参数是位置、方向和切角来计算对应的光照值。



在切角范围内结合光照位置计算出正确的光照，在切角范围外就只使用环境光，从而可以形成聚光效果。但是这样在分界线处边缘化比较严重，不自然，因此利用平滑技术软化边缘，计算出内切光角和外切光角，使边缘的强度介于两边之间，结果不会突兀，从而达到更加柔和的效果。

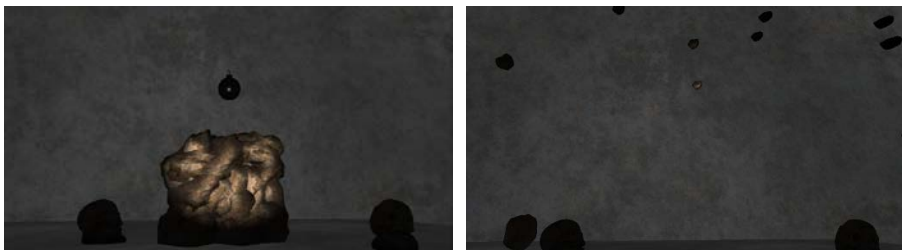
以下是添加了光照的效果：



## 10. 炸弹和爆炸

导入炸弹模型，这里简单地设置触发炸弹后炸弹从天而降，触碰石堆以后石堆发生爆炸，并且碰撞检测也很简单，只比较了炸弹和模型最高点的y坐标是否相等，相等则触发爆炸，并且播放爆炸音效，石堆产生很多小石头碎片，给予石堆初速度和生命，迅速飞出，速度逐渐变慢，最终消失。

但有一个小问题是加上音效时，碰撞之后会有延迟才产生爆炸，但是不加上音效就很正常，关于这个问题没有找到合适的解决办法。



## 11. 佛像模型显示

因为感觉重点就是读取模型文件处理模型，所以用库的意义不大，所以考虑自己设计一个读取类，用于处理和渲染佛像模型。读取模型文件，存储顶点和面信息，计算法向量进行存储。

按照材质表查找选择合适的灰质石头、银、金的材质参数，传进着色器进行光照计算。为了使颜色能够平滑过渡，将材质根据时间进行线性计算，从而实现更好的渐变效果。



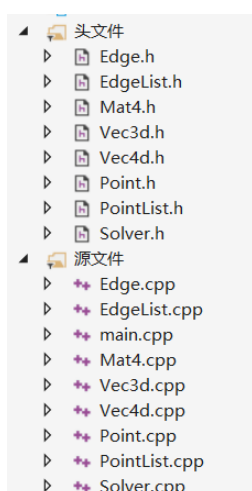
## 12. 多线程优化

采用多线程技术，同时处理四个模型读取和主函数的计算、纹理加载，从而可以提升效率。

## 13. 边塌陷技术

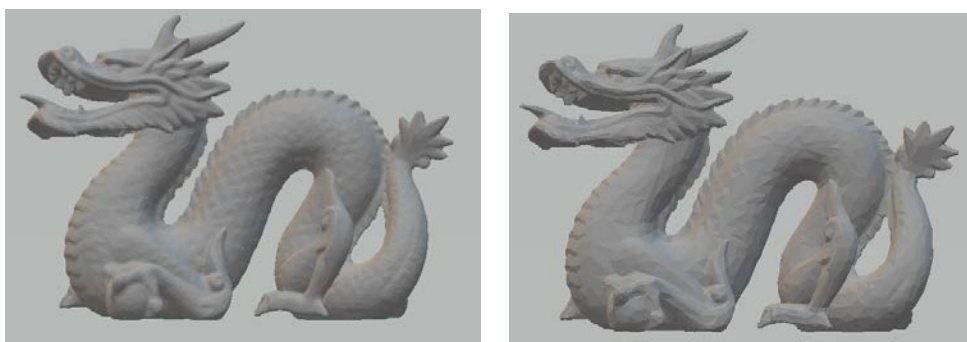


尝试了一下边塌陷技术，采用了最优点放置策略达到简化网格的目的。



Edge 表示一条边,EdgeList 表示模型中所有边,Mat4、Vec3d、Vec4d、Solver 用来处理数学计算,Point 表示一个点,PointList 表示模型中所有点。

需要传入文件名和简化率,便会读取顶点和面信息,生成顶点列表、边列表,再根据简化率确定简化循环什么时候结束。简化过程为遍历需要简化的边,得到边的顶点信息以及顶点的邻边信息,算出法向量、误差矩阵,求出价值最低的点,计算得到最优收缩点。以简化率 0.1 效果如下:



整体的模型轮廓依然保持不变,但是表面的凹凸形状随着边塌陷可见有不同程度的模糊效果。对于 50000 个顶点的模型,在简化率为 0.7 时可以在 1s 内完成简化,简化率为 0.1 时需要花费 4s 才能完成简化,因此觉得对于优化并没有太多作用,因此舍弃了边塌陷简化。

### 三、 程序操作说明

1. 每一个场景都可以滑动鼠标调整摄像机方向
2. 按键 wasd 可以调整摄像机位置, 分别代表上左下右
3. 在第二个第三个场景按下空格键可以打开手电筒
4. 在第二个场景按下 b 键可以出现炸弹

### 四、 项目感想

通过此次实践,对于 Opengl 有了更加深刻的体会,采用了更加灵活的库 GLFW,学会了编写着色器,能够更好地从原理上理解了光照、渲染、贴图、摄像机等等技术。整体项目从布置的时候就开始编写,期间也遇到了很多问题,包括选择什么库合适、如何调整参数、怎样提升渲染效率等等,也想添加更多灵活的动画,但是技术不够也没有实现。但是也实现了很多好看的效果,提升了很多能力。