# 计算机图形学大作业
# 局部光照模型

刘彦铭 ID: 122033910081

Last Edited: 2023 年 6 月 9 日

## 目录

**1 概览** **2**

**2 实现方法** **2**
2.1 概览 . . . . . 2
2.2 Phong 局部光照模型 . . . . . 3
2.3 阴影贴图 . . . . . 4
2.4 半透明 Blend . . . . . 4
2.4.1 Blend . . . . . 4
2.4.2 半透明 & 阴影 . . . . . 4
2.4.3 法线问题 . . . . . 4

**3 结果展示** **5**

# 1 概览

表 1: 功能特性一览表

| 功能/特性 | 状态 |
|---|---|
| 系统平台 | macOS 13 / Windows 11* |
| 开发语言 | C++ |
| 图形 API | OpenGL 4.1 / GLSL version 330 |
| 局部光照模型 | Phong |
| 环境光 | ✓ |
| 漫反射 | ✓ |
| 镜面反射 | ✓ |
| 阴影 | ✓ |
| 半透明 | ✓ |
| 阴影实现方式 | Shadow Map |
| 半透明实现方式 | Blend (非 OIT) |
| RGB | ✗ |
| 纹理 | ✗ |
| 抗锯齿 | ✗ |
| 光源数量 | 2 点光源 |
| 圆柱 | 1 |
| 立方体 | 3 (1 不透明 +2 半透明) |
| 圆椎 | 1 |
| 圆环 | 1 |
| 20 面体 | 1 |
| 球体 | 2 (1 网面球 +1 棱角球) |
| 视角可转动 | ✓ |
| 运动光源 | ✗ |
| 场景布局从模型文件中加载 | ✓ |
| 物体材质从模型文件中加载 | ✗ |

本次大作业选取了 Project 1，实现了 Model 1 中的 Phong 局部光照模型。实现了作业要求中提及的环境光、漫反射、镜面反射、阴影以及透明效果。具体情况如表1所示。

因为是从零开始尝试学习 OpenGL，代码中有诸多早先遗留下来的问题，比如颜色上只实现了灰度，比如反射系数、光源强度等都是硬编码的 (相应的，我导出的模型.obj 文件中虽然有 UV 坐标，但我并没有提供贴图)，考虑到不太影响局部光照模型的实现，简单起见就没有去改这些特性了。但对作业要求中提到的几种光照相关的特性都进行了实现。此外，物体布局没有硬编码，可以通过编辑 scene.txt、scene-2.txt 文件来进行调整。

# 2 实现方法

## 2.1 概览

整体实现流程如下:

1. 加载模型

*本次作业在 macOS 13 arm64 平台上开发，主要针对该平台进行调试。针对 macOS 13 提供了通用 (arm64 / x86_64) 的预编译静态链接库和二进制可执行文件，详见 README.md。针对 Windows 11 x86_64 平台，也提供了相应的预编译好的二进制可执行文件，但没有打包对应的头文件、静态链接库以及 makefile。

- 读取模型文件
- 生成并绑定 Vertex Array Object、Vertex Buffer Object 等，传递三角形顶点的坐标信息

2. 生成深度贴图

- 生成、绑定 Depth Texture
- 编译 Shader Language
- 对每个光源，以光源为摄像机位置，进行一次 Draw Call，绘制整个场景并把深度信息记录到绑定的 Depth Texture 里

3. 准备工作

- 编译实现 Phong 局部光照模型的 Shader Language
- 计算要用到的 Model、View、Projection 矩阵，并通过 Uniform 变量传递

4. 主循环

- Draw Call；在 Fragment Shader 阶段完成光照相关的计算。
  - 先打开深度测试，打开深度写入，关掉 Blend，渲染非透明物体
  - 再打开深度测试，关闭深度写入，打开 Blend，渲染半透明物体
- 交换帧缓存到窗口
- 接收键盘事件，更新 View 矩阵，循环

## 2.2 Phong 局部光照模型

按照公式 (1) 实现

$$I = I_{\texttt{Ambient}} + \sum_{i=1}^{n=2} \frac{1}{a + bd_i + cd_i^2} \left(k_d I_i (l \cdot n) + k_s \cdot I_i (r \cdot v)^{\alpha}\right) \tag{1}$$

，其中 $d_i$ 是光源到片元 (fragment) 的距离，$I_i$ 是光源强度，$k_d$,$k_s$ 分别是漫反射和镜面反射的系数，单位向量 $l, n, r, v$ 分别是光线入射方向，片元的法向，光线出射方向，摄像机相对片元的方向。常数如表2所示：

表 2: 常数表

| 常数 | 数值 |
|---|---|
| 环境光 $I_{\texttt{Ambient}}$ | 0.1 |
| 漫发射 $k_d$ | 0.3 |
| 镜面反射 $k_s$ | 1.0 |
| 镜面反射 $\alpha$ | 2 |
| 距离 $a, b, c$ | $0, 0, 1$ |
| 光源强度 $I_1, I_2$ | $90, 135$ |

## 2.3 阴影贴图

开启深度测试和深度写入。对每个光源，以光源为摄像机位置，进行一次 Draw Call。将深度信息存入一个预先准备好的 Texture Buffer 中。由于开启了深度测试和深度写入，Texture Buffer 实际上记录了从光源位置出发能够直接看到的场景中的点的深度信息，也即是离光源的远近信息。可以理解为，是记录了光源射出的光线所经过的第一个物体表面里离光源的距离（经过折算，转换到 $(0, 1)$）。

如果是点光源就用透视投影，平行光就用正交投影。本次作业里面是点光源，故才用透视投影。视角为 80°。

计算光照时，用光源对应的 Model、View、Projection 矩阵计算片元在光源视角下的位置，比较它的深度和 Texture Buffer 中存下的深度，如果片元的深度更深，说明片元被遮挡，应该有阴影，故而为光线强度乘上一个衰减系数，这里直接设为 0。否则，片元能被光直射，没有阴影。

一个小的 Trick: 为了避免出现错误的条纹状的阴影效果，需要在比较深度时加入一个阈值，差异超过阈值时才渲染阴影。本作业中采取了 OpenGL-Tutorial 中的示例，基于入射光线和法线夹角的正切来计算这一阈值。

## 2.4 半透明 Blend

### 2.4.1 Blend

本次作业使用的 OpenGL4.1 版本有些旧了，似乎用不了 Shader Storage Buffer Object 相关的特性，不太方便实现顺序无关透明 (OIT)，故而只利用 Blend 实现了最简单的半透明效果。简单来说就是按照系数 $\alpha$，$1 - \alpha$ 来混合半透明物体和背景物体的颜色。这里取 $\alpha = 0.2$ 。

因为是顺序有关的，因此要先正常渲染非透明物体，然后在非透明物体的深度信息的基础上，在继续绘制非透明物体。绘制中，继续打开深度测试（抛弃被非透明物体遮挡的半透明片元），但关闭深度写入。这样不能处理复杂场景，比如透明物体之间有交叉重叠的情况。但对简单的场景还是能够进行看起来勉强合理的渲染的。

### 2.4.2 半透明 & 阴影

另一方面，因为加入了半透明物体，其阴影也要做出相应的半透明效果。为此，在生成阴影贴图时，将非透明物体和透明物体分开生成深度贴图。

计算光照时，假设片元在光源视角下的深度是 $d$, 非透明深度贴图为 $d_o$, 半透明深度贴图为 $d_t$。如果 $d > d_o$, 说明被不透明物体遮挡，光线置为 0; 如果 $d \leq d_o$ 但 $d > d_t$, 说明只被半透明物体遮挡，光线乘以 $1 - \alpha$; 如果 $d \leq d_o$ 且 $d \leq d_t$, 则没有阴影。

这样做有个缺陷，就是光线透过多个半透明物体和透过单个半透明物体所产生的阴影是一样的。但好在方法简单，而且简单场景下的效果其实还可以。

### 2.4.3 法线问题

如果光线入射角和片元法线夹角大于 90°，漫反射那一项会是负的，这时候我们可能需要将它置零才能正确渲染。一般非透明的情形下这不会有什么问题，但半透明的场景下会出现一些不太一样的情况。

利用 Blender 之类的软件建模出来的模型，它的表面法向都是向外的。但如果是半透明物体，相对光源深度上靠后的面的法线方向其实是朝内的，而且由于关闭了深度写入，这些较深的面也是会计算光照的，所以需要反转它们的法线方向。如图1所示。

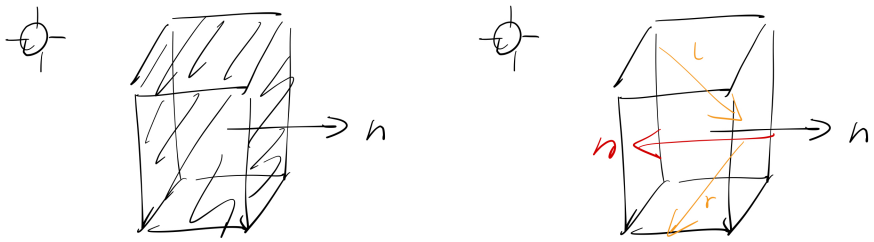


图 1: 示意图。左图是非透明情形，右图是透明情形。远离光源的面的实际上的法向是朝物体内部的，而模型文件里面记录的是朝外的。

## 3 结果展示