

图形学实验 PA1：光线投射

指导教师：胡事民 助教：陈拓、冯启源

2025 年 3 月 18 日

1 实验综述

本实验要求实现课上所讲的光线投射（Ray Casting）算法，你将实现一个简单的渲染引擎，支持包括球、平面和三角面片等多种基本几何体的绘制，并为它们加入基本的 Phong 材质。本次作业将是本课程渲染大作业的基础，请大家充分重视，尽早开始。和光栅图形学作业一样，我们为大家准备了一些基本的框架代码。在本课程的渲染大作业中，你需要基于这套框架代码完成完整的光线追踪算法。我们将主要在 7 个测例上评测你的代码。

2 细节说明

2.1 算法框架

光线投射的基本框架我们在第 4 周的课程上介绍，其基本思想是：把屏幕想象成相机的成像平面，每个像素发射一条射线和场景中的物体求交并找到最近交点，之后计算该点处由光源照射影响产生的局部光强即可。本次实验中我们不要求实现阴影。代码如下：

```
1 // 循环屏幕空间的像素
2 for (int x = 0; x < camera->getWidth(); ++x) {
3     for (int y = 0; y < camera->getHeight(); ++y) {
4         // 计算当前像素(x,y)处相机出射光线camRay
5         Ray camRay = sceneParser.getCamera()->generateRay(Vector2f(x, y));
6         Group* baseGroup = sceneParser.getGroup();
7         Hit hit;
8         // 判断camRay是否和场景有交点，并返回最近交点的数据，存储在hit中
9         bool isIntersect = baseGroup->intersect(camRay, hit, 0);
10        if (isIntersect) {
11            Vector3f finalColor = Vector3f::ZERO;
12            // 找到交点之后，累加来自所有光源的光强影响
13            for (int li = 0; li < sceneParser.getNumLights(); ++li) {
14                Light* light = sceneParser.getLight(li);
15                Vector3f L, lightColor;
16                // 获得光照强度
17                light->getIllumination(camRay.pointAtParameter(hit.getT()), L, lightColor);
18                // 计算局部光强
19                finalColor += hit.getMaterial()->Shade(camRay, hit, L, lightColor);
20            }
21            setPixel(x, y, finalColor);
22        } else {
23            // 不存在交点，返回背景色
24            setPixel(x, y, sceneParser.getBackgroundColor());
25        }
26    }
```

2.2 几何求交

从相机发出的射线需要和场景进行求交，一条射线 R 由发射点 O_R 和出射方向 \vec{d}_R 组成的二元组定义。我们推荐采用课程上介绍的几何方法求解光线和球体以及平面的交点，和三角形求交的时候先求和三角形所在平面的交点，再判断交点是否在三角形内部（即重心坐标 $\alpha, \beta, \gamma \in [0, 1]$ ）。

2.3 透视相机模型

透视相机（也称小孔相机：Pinhole Camera）依据的是小孔成像原理，这种模型渲染出的图片和真实世界拍摄的图片非常相似。描述一个相机自身的参数也成为相机的内参（Intrinsics），最常用的透视相机表示方式含有 6 个参数 $(w, h, f_x, f_y, c_x, c_y)$ ，其中 (w, h) 是所拍照片的宽度和高度， (c_x, c_y) 是光心位置，一般为图像中心点 $(w/2, h/2)$ ， (f_x, f_y) 是图像空间到真实世界空间的尺度参数，即图像中一个像素的距离在真实场景中对应的长度。 (f_x, f_y) 与图像的分辨率和相机的视场角有关。我们在场景配置文件中给出了相机垂直方向和水平方向观测范围的夹角，即图1中的 *angle*。假设我们需要计算像素 (u, v) 处的射线，我们首先计算相机空间下（即以相机为原点的坐标系）的射线 R_c ：

$$O_{R_c} = (0, 0, 0)^\top, \quad \vec{d}_{R_c} = \text{normalized}\left(\frac{u - c_x}{f_x}, \frac{c_y - v}{f_y}, 1\right)^\top, \quad (1)$$

接着，通过坐标转换，将相机空间下的射线 R_c 转换到世界空间下的射线 R_w ，便于和场景计算交点：

$$O_{R_w} = \mathbf{t}, \quad \vec{d}_{R_w} = \mathbf{R}\vec{d}_{R_c}, \quad (2)$$

其中， $\mathbf{R} \in \mathbb{R}^{3 \times 3}$ 是相机的旋转矩阵， $\mathbf{t} \in \mathbb{R}^3$ 是相机的位移（即相机中心点在世界坐标系下的位置），这两个参数确定了相机的位姿（Pose），有时也被称作相机的外参（Extrinsics）。旋转矩阵的 3 列可以看做 $\mathbb{R}^{3 \times 3}$ 的一组基，给定了相机的三个正交的主方向单位向量 $\vec{up}, \vec{horizontal}, \vec{direction}$ （如图1所示），则 $\mathbf{R} = [\vec{horizontal}, -\vec{up}, \vec{direction}]$ 。

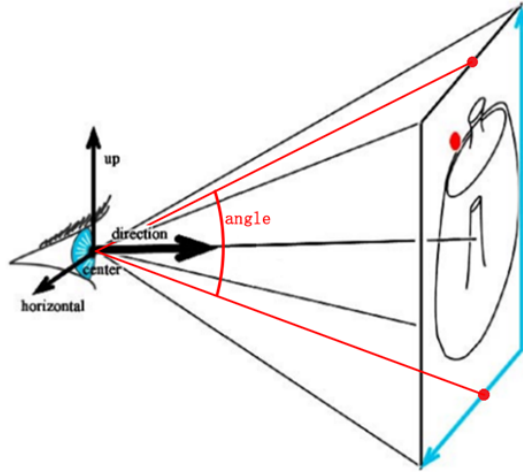


图 1: 透视相机模型

如果想详细地学习这种小孔相机模型，我们推荐Scratchapixel网站作为参考，上面包含了相机成像原理的详细解释和更多的示例代码。

2.4 Phong 模型着色

使用 Phong 模型计算局部光强，每个像素最终的着色公式为：

$$I_{\text{pixel}} = \underbrace{c_{\text{ambient}} k_a}_{\text{Ambient}} + \sum_i c_i \left(\underbrace{k_d \text{clamp}(\vec{L}_i \cdot \vec{N})}_{\text{Diffuse}} + \underbrace{k_s \text{clamp}(\vec{V} \cdot \vec{R}_i)^s}_{\text{Specular}} \right).$$

本次作业的所有例子都不包含环境光项（即 $c_{\text{ambient}} = 0, k_a = (0, 0, 0)^T$ ）。 c_i 是第 i 个光源的颜色。 \vec{N} 代表相机射线 R_w 和物体相交处 P 的法向， \vec{L}_i 代表从相交处 P 指向第 i 个光源的单位向量， $\vec{V} = -\vec{d}_{R_w}$ ， \vec{R}_i 代表光源发出的光在物体表面 P 处的反射光线方向，其计算公式为：

$$\vec{R} = 2(\vec{N} \cdot \vec{L}_i)\vec{N} - \vec{L}_i,$$

注意上述所有方向向量均为单位向量， $\vec{X} \cdot \vec{Y} = \vec{X}^T \vec{Y}$ 为向量 \vec{X} 和 \vec{Y} 的点积 (dot product)。 k_a, k_d, k_s 是物体的材质属性（环境光系数、漫反射系数、镜面反射系数）， k_d 可以设置为物体本身的固有颜色，而 k_s 是镜面反射的高光颜色，通常是白色， s 是光泽度。材质及光照均为包含 RGB 三个分量的向量。clamp(\cdot) 表示截断函数，即将负数值置零。

3 框架代码说明

3.1 环境配置与编译

环境要求：

- Windows: WSL 下安装 CMake
- Linux: CMake
- MacOS: CMake

如果你使用其他系统的时候遇到了编译问题，请先尝试自行解决。我们的框架代码没有任何外部依赖，请在包含有 `run_all.sh` 的文件夹下打开终端，并执行：

```
1 bash ./run_all.sh
```

这段脚本会自动设置编译，并在 7 个测例上运行你的程序。你的程序最终会被编译到 `bin/` 文件夹中，而输出的渲染图片位置在 `output/` 文件夹中。我们在框架代码中去除了光线投射的重要逻辑，因此现在这段代码仅仅是能通过编译而已，你需要按照课上讲的算法自行进行实现。

调试方法可参照 PA0 的 README.pdf。

4 测试用例

为了测试代码是否正确无误，我们构建了以下 7 个测试用例，你也可以根据场景的文件格式构造样例进行自我测试。但我们在检查作业的时候将主要检查这 7 个样例的输出结果。

五个球体 `testcases/scene01_basic.txt` 包含了一个平面和五个球体，其中一个红色球体包含有高光，参考效果如图2所示。

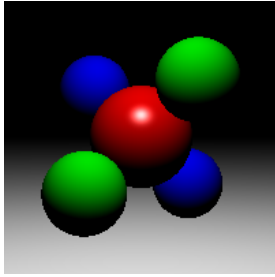


图 2: 五个球体

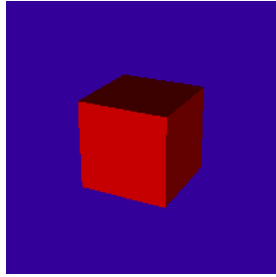


图 3: 正方体

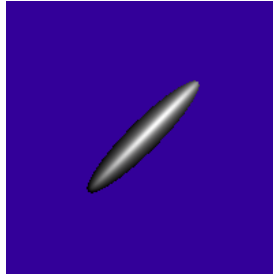


图 4: 变形球体

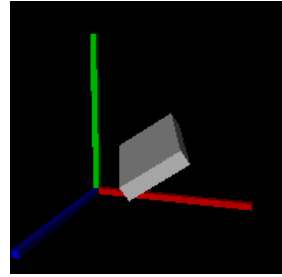


图 5: 坐标系



图 6: 简单兔子



图 7: 复杂兔子

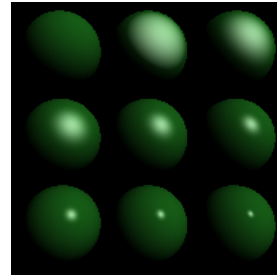


图 8: 镜面反射

正方体 `testcases/scene02_cube.txt` 包含一个红色的正方体, 该正方体实际上是一个 obj 模型, 所有 obj 模型都放在了 `mesh` 文件夹下, 如果你的程序提示找不到文件, 请检查路径是否正确。参考图如图3所示。

变形球体 `testcases/scene03_sphere.txt` 一个压缩并旋转了的球体, 看起来像一根针, Transform 类的正确实现已经为你提供, 因此如果你的 Sphere 求交算法实现正确, 则理论上可以直接输出结果。关于 Transform 类对物体进行变形的原理, 请参考这个文档10.8 节的 Instancing。参考图请见图4。

坐标系 `testcases/scene04_axes.txt` 用拉长了的正方体组成的坐标系, 效果如图5所示。

简单兔子 `testcases/scene05_bunny_200.txt` 一个包含了 200 个三角面片的斯坦福兔子 (原始模型下载地址)。效果如图6。

复杂兔子 `testcases/scene06_bunny_1k.txt` 同样的斯坦福兔子, 面片个数为 1000, 同时对网格进行了缩放。参考效果如图7。

镜面反射 `testcases/scene07_shine.txt` 由 9 个绿色的球组成 (微笑), 每个球的光泽度 (shininess) 都不一样。参考效果如图8所示。

5 作业要求

因为本次作业代码量比较大, 同学们可以按照本节推荐步骤进行一步步的代码实现。

- 首先找一下熟悉的感觉, 我们的 `vecmath` 库以及 `Image` 类和上次 PA 是一样的。实际上之后的 PA 也大多会依赖于前一次 PA 的框架, 我们希望大家通过不断修改、完善, 最

终形成一套自己的图形学系统。从代码风格上来看，大多数实现都直接写在了头文件里，你可以选择性地重构代码。

- 看 `Object3D` 类，是一个虚基类，提供了 `intersect` 方法，代表了三维世界中的物体，你自己的物体（例如球、平面）需要继承它并实现虚方法。
- 实现 `Sphere` 类的 `intersect` 方法，一个球体由中心和半径来定义。求交的时候注意两个量，第一个是 `Hit` 结构中的 t ，另一个是 `tmin`，前者存储了之前的相交测试中最小的 t ，如果此次相交算出的 t 比上次的大，则说明本次相交并不是离相机最近的，应该舍弃，否则应该相应地更新 t 。`tmin` 是一个常量，定义了最小可能的 t 值，如果 $t < \text{tmin}$ ，本次相交也应该舍弃。此外，你也需要将计算好的相交处法向 \vec{N} 传入 `Hit`。
- 相类似地，按照课件上的公式和逻辑实现 `Plane` 和 `Triangle` 类的 `intersect` 方法。类 `Mesh` 的相交方法遍历了所有存储的三角形依次求交，最终形成一个完整的模型渲染，已经为你实现好了。
- 看 `Transform` 类，作用是对物体进行仿射变换。在求交过程中，我们不会将变换矩阵作用到物体上，而是将其逆变换作用到射线上。你需要检查前面的求交代码处理 `Transform` 类物体的正确性。
- 实现 `Group` 类，该类存储了 `Object3D` 的一个列表，推荐用 STL 容器实现。
- 实现 `PerspectiveCamera` 类的构造函数和 `generateRay` 方法。具体的相机模型可以参考 2.3 节。
- 看 `SceneParse` 类，作用是读入配置文件，已经为你实现好了。
- 实现 `main.cpp` 中的主逻辑，我们在 2.1 节中给出的伪代码基本上可以直接拷贝过来使用，但你首先需要构造一个 `SceneParse` 类来从命令行读入配置文件，具体命令行输入请参考框架代码。
- 实现 `Material` 类的 `Shade` 方法，该方法已经将计算 Phong 模型光照（2.4 节）所需的变量都传进来了，因此你只需要把公式输入进去即可。

所有需要填写的地方都被标记了 `TODO`，每实现完成一个功能，你就可以去掉一个 `TODO`。由于我们的大作业光线追踪的基础就是光线投射，因此我们也鼓励你通读所有的代码，了解程序的整体运行机制。

在确认执行 `bash ./run_all.sh` 可以输出正确的图片后，请将你代码中的 `include` 与 `src` 文件夹一起打包成 `zip` 文件提交到 TUOJ 平台进行测试，同时将文档 `REPORT.pdf` 文档文件提交至网络学堂。

报告 `REPORT.pdf` 中应包含且只包含以下几个部分：

1. 代码逻辑：你所实现的光线投射算法逻辑是怎样的？你在实现中遇到了哪些问题？
2. 代码参考：完成作业的时候和哪些同学进行了怎样的讨论？是否借鉴了网上/别的同学的代码？
3. (可选) 问题：你在实现过程中遇到了哪些问题？
4. (可选) 未解决的困难：你的代码有哪些未解决的 bug？如果给你更多时间来完成作业，你将会怎样进行调试？

5. (可选) 建议: 你对本次作业有什么建议? 文档或代码中有哪些需要我们改进的地方?

注意事项:

- 如果本地测例输出正确, 但是提交到 TUOJ 平台上发现失败了, 那可能是因为你的代码中存在 Undefined Behavior 或者内存越界等错误, 恰好在你自己的系统 + 编译器下能跑通, 请检查自己的代码。
- 报告请提交 pdf 文件, 不要提交 doc, docx 或 markdown 文件。如果你使用 word、latex 或 markdown 来撰写报告, 请转换为 pdf 提交。

本次作业的 Deadline 以网络学堂为准。迟交的同学将得到一定的惩罚: 晚交 3 天内分数将降低为 80%, 3 天以上 1 周以内分数降为 50%, 迟交一周以上的同学分数为 0。

6 致谢

本实验文档和代码部分借鉴于MIT Open Courseware, 按照其发布协议, 本文档原则上允许同学们以CC BY-NC-SA 4.0协议共享引用, 但是由于教学需要请同学们尽量不要将本文档或框架代码随意传播, 感谢同学们的支持。