

1 总览

在这部分的课程中，我们将专注于使用光线追踪来渲染图像。在光线追踪中最重要的操作之一就是找到光线与物体的交点。一旦找到光线与物体的交点，就可以执行着色并返回像素颜色。在这次作业中，我们需要实现两个部分：光线的生成和光线与三角的相交。本次代码框架的工作流程为：

1. 从 `main` 函数开始。我们定义场景的参数，添加物体（球体或三角形）到场景中，并设置其材质，然后将光源添加到场景中。
2. 调用 `Render(scene)` 函数。在遍历所有像素的循环里，生成对应的光线并将返回的颜色保存在帧缓冲区 (`framebuffer`) 中。在渲染过程结束后，帧缓冲区中的信息将被保存为图像。
3. 在生成像素对应的光线后，我们调用 `CastRay` 函数，该函数调用 `trace` 来查询光线与场景中最近的对象的交点。
4. 然后，我们在此交点执行着色。我们设置了三种不同的着色情况，并且已经为你提供了代码。

你需要修改的函数是：

- **Renderer.cpp 中的 `Render()`**：这里你需要为每个像素生成一条对应的光线，然后调用函数 `castRay()` 来得到颜色，最后将颜色存储在帧缓冲区的相应像素中。
- **Triangle.hpp 中的 `rayTriangleIntersect()`**：`v0`, `v1`, `v2` 是三角形的三个顶点，`orig` 是光线的起点，`dir` 是光线单位化的方向向量。`tnear`, `u`, `v` 是你需要使用我们课上推导的 Moller-Trumbore 算法来更新的参数。

2 开始编写

在本次作业中，你将使用一个新的代码框架。和之前作业相似的是，你可以选择在自己电脑的系统或者虚拟机上完成作业。请下载项目的框架代码，并使用

以下命令像以前一样构建项目：

```
1 $ mkdir build
2 $ cd build
3 $ cmake ..
4 $ make
```

之后，你就可以使用 `./Raytracing` 来运行代码。现在我们对代码框架中的一些类做一下概括性的介绍：

- `global.hpp`: 包含了整个框架中会使用的基本函数和变量。
- `Vector.hpp`: 由于我们不再使用 Eigen 库，因此我们在此处提供了常见的向量操作，例如：`dotProduct`，`crossProduct`，`normalize`。
- `Object.hpp`: 渲染物体的父类。`Triangle` 和 `Sphere` 类都是从该类继承的。
- `Scene.hpp`: 定义要渲染的场景。包括设置参数，物体以及灯光。
- `Renderer.hpp`: 渲染器类，它实现了所有光线追踪的操作。

3 评分与提交（Games平台要求，仅供助教同学参考）

评分：

- [5 分] 提交的格式正确，包含所有必须的文件。代码可以编译和运行。
- [10 分] 光线生成：
正确实现光线生成部分，并且能够看到图像中的两个球体。
- [15 分] 光线与三角形相交：
正确实现了 Moller-Trumbore 算法，并且能够看到图像中的地面。
- [-2 分] 惩罚分数：
未删除 `/build`，`/.vscode` 和 `assignment5.pdf`。
未按格式建立 `/images`，缺少结果图片。

未提交或未按要求完成 README.md。

代码相关文件和 README 文件不在你提交的文件夹下的第一层。

如果实现是正确的，你将得到下图：

