图形学 PA1 report

计83 何雨泽 2018011351

1. 你所实现的光线投射算法逻辑是怎样的?

- 首先读入相机的配置信息并计算出 direction, up, horizontal 等信息；然后取一个假想的、垂直于 direction 的平面，再根据 angle 和 up 和 horizontal 求出平面投影区域的边界，将这个有限四边形区域平分成 w * h 块（参数均已给定），每一块都对应一条从摄像机引出并穿过它的一个向量（即 Ray），经过一系列算法后获取它对应的颜色，每个块最后在图片上体现出的都是一个像素点。
- 对于每个向量（Ray），都依次按算法判断它是否与某个立体图形相交（通过遍历 Group 里的 Vector 实现）。如果与任意图形不相交，直接设置为给定背景色；否则取其最前端相交的图形，通过 Phong 模型进行颜色的计算。

Phong 模型计算着色

$$I_{pixel} = c_a k_a + \sum_x c_x (k_d ReLU(\overrightarrow{L_x} \cdot \vec{N}) + k_s ReLU(\vec{V} \cdot \overrightarrow{R_x})^S)$$

> 其中 ReLU(x) = max(0, x)；由于不考虑环境光，$c_a = k_a = 0$；c_x 为第 x 个光源的颜色；$\vec{N}$ 为相机 Ray 与立体图形的交点 P 的单位法向量；$\overrightarrow{L_x}$ 为交点 P 到第 x 个光源的单位向量，$\vec{V}$ 是相机 Ray 方向向量（已归一化）的相反数；k_d, k_a, k_s, s 分别是物体的漫反射系数、镜面反射系数、环境光系数、光泽度，是已给定的条件；$\overrightarrow{R_x}$ 为交点 P 处的反射光方向，$\overrightarrow{R_x} = 2(\overrightarrow{L_x} \cdot \vec{N})\vec{N} - \overrightarrow{L_x}$

编程时所有信息均为成员变量或函数参数，直接编写公式即可。

判断向量与立体图形相交的算法

- 对于平面，我首先检测摄像机的 Ray 是否与平面的法向量垂直，若垂直，直接返回 false；然后按照下列公式计算摄像机 Ray 到平面的距离 t：

  $$t = \frac{d - \vec{n} \cdot \vec{o}}{\vec{n} \cdot \vec{d}}$$

  > 其中 d 是平面的 offset，$\vec{n}$ 是平面的法向量，$\vec{o}$ 是摄像机的位置，$\vec{d}$ 是摄像机 Ray 的方向向量。

  然后检测是否满足 $t < tmin$ 且 $t > h.getT()$，若满足则证明这是一个更近的 Ray 可以投射到的平面，更新 hit；若不满足，直接返回 false。

- 对于三角形，先按以下公式求出三角形所在平面的法向量和偏移量：

  $$\vec{n} = (\vec{b} - \vec{a}) \times (\vec{c} - \vec{a})$$
  $$d = \vec{n} \cdot \vec{a}$$

  然后也是首先检测摄像机的 Ray 是否与平面的法向量垂直，若垂直，直接返回 false；然后按照上述相同公式求出摄像机沿 Ray 到平面的距离 t，并检测是否满足 $t < tmin$ 且 $t > h.getT()$，若不满足直接返回 false。

若满足，进行三次叉积检验，若这三个叉积结果均与 Ray 和平面的交点向量**方向一致**（可用点乘是否大于 0 判断），那么则可判断该点在三角形中：

$$\overrightarrow{n_1} = (\vec{a} - \vec{p}) \times (\vec{b} - \vec{p})$$
$$\overrightarrow{n_2} = (\vec{b} - \vec{p}) \times (\vec{c} - \vec{p})$$
$$\overrightarrow{n_3} = (\vec{c} - \vec{p}) \times (\vec{a} - \vec{p})$$

若检验成功，以三角形的法向量和求出的 t 更新 hit 值；若不成功，直接返回 false。

- 对于球，我首先求得圆心到 Ray 的垂线所在直线的向量 $\vec{v}$：

$$\vec{n} = \vec{d} \times (\vec{c} - \vec{o})$$
$$\vec{v} = \vec{n} \times \vec{d}$$

然后通过解 $\vec{v}$ 与 $\vec{d}$ 的线性方程组得到垂点的坐标 $\vec{h}$ （需要讨论较多情况，比如光线恰好过球心时，法向量是非法值等等）；

由以下公式求得距离 t：

$$t = |\vec{h} - \vec{o}| - \sqrt{r^2 - |\vec{c} - \vec{o}|^2}$$

> 其中 r 是球的半径，$\vec{c}$ 是球的球心位置。

检验 t 是否满足 $t < tmin$ 且 $t > h.getT()$，若不满足直接返回 false；

最后求得交点位置的法向量：

$$\vec{p} = \vec{o} + t\vec{d}$$
$$\overrightarrow{n_p} = \vec{p} - \vec{c}$$

以求得的法向量和 t 更新 hit 值。

2. 你在实现中遇到了哪些问题？

- 助教给出的 ppt 中是否在三角形内的判断其实只是是否在一条线段某侧的判断，判断在三角形内需要将三角形的三条边都判断一遍，最开始没有注意到，浪费了一些时间；
- 做球的光线投射时，采用了较为基础的解法，导致需要讨论的情况很多，不然图像中会出现黑线、黑点等异常情况，花费时间较多；
- 最开始没有注意到框架中自动把输入的 angle 从角度制转换为弧度制了，de 了一会 bug。

3. 你在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？

没有！完成时间为 3.1，估计也没啥同学在这个时间点做完 >_<

4. 如何编译你的代码并输出上述 7 个测试用例的渲染结果？如果你没有使用框架代码，请放上你渲染的图片结果。

我将代码放在了远程 ubuntu 服务器上，用 vscode 进行 ssh 连接并编写。由于 CMakeLists.txt 等文件已经写好，我只需要在 terminal 中输入 `./run_all.sh` 即可完成编译和运行。

我直接使用了框架中 `Image` 类的 `SaveBMP` 方法进行图像渲染。

5. 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?

暂时貌似没有发现 bug。

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

个人感觉非常吼！但对于 0 基础的同学来说可能稍微有些挑战性，因为他们没有接触过大的框架以及 Cmake 等软件，可能上手会稍微困难，可以考虑加一些说明（当然不加也没有任何问题！自学是很重要滴！）

以及 ppt 上面一些不全面的（如判断点是否在三角形内部那一块似乎有公式缺失）可以考虑补充一下！