Rays, a Simple Camera, and Background

Xiaomx32

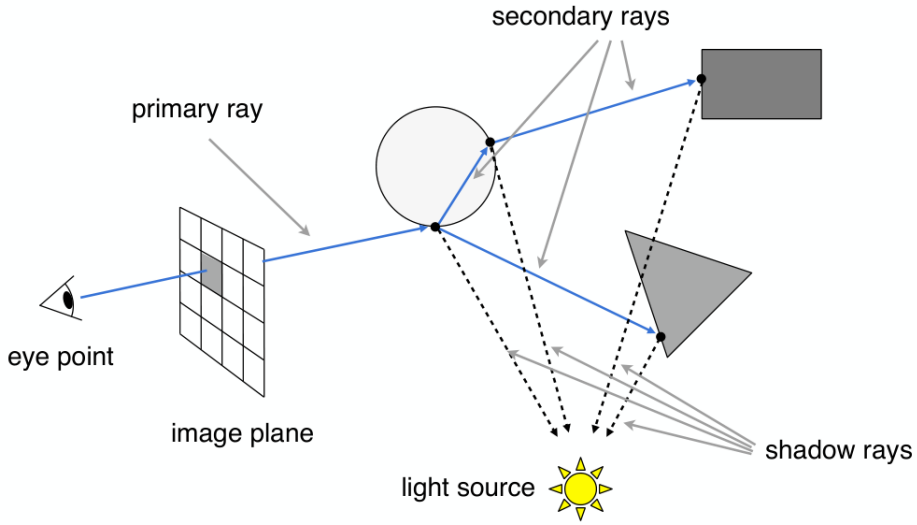
2022 年 6 月 7 日

总览

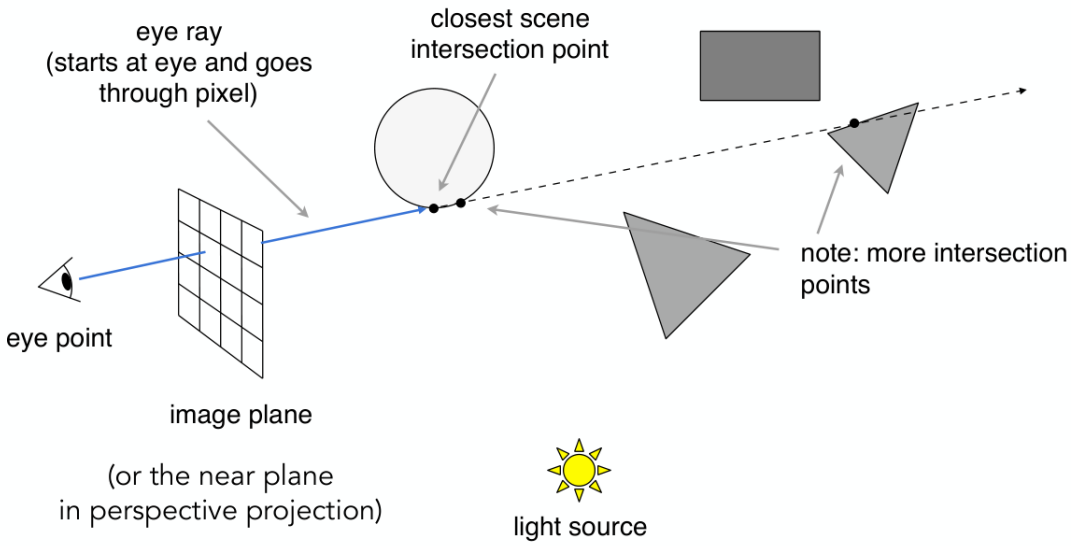
在这一节，加入光线、简单的照相机、图片背景；绘制一幅沿 y 轴的白蓝渐变图。

光线追踪，会从终点（眼睛/相机）开始，首先做的是光线投射（ray casting）

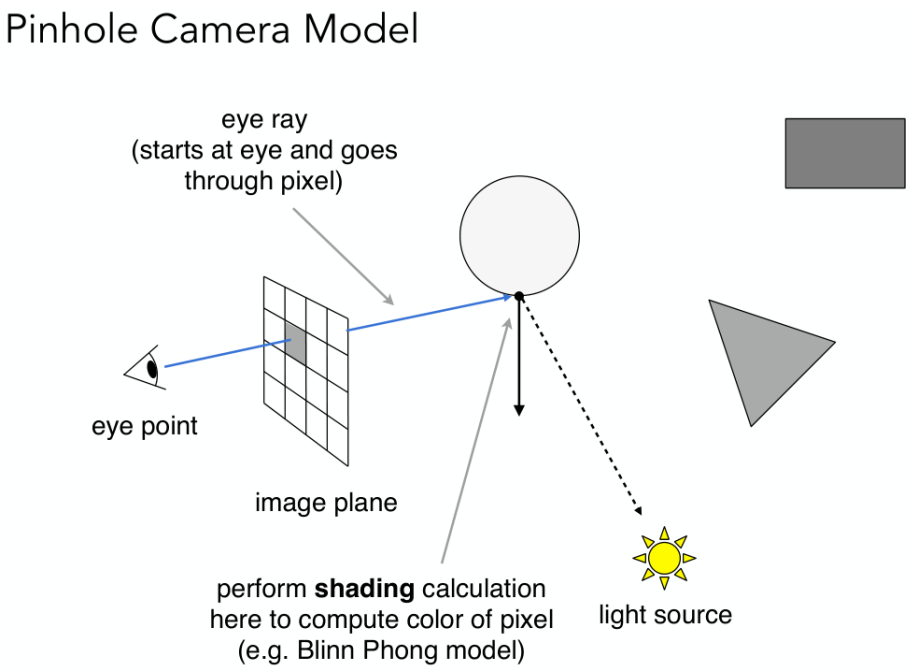
如下图所示，我们假设往虚拟的世界中看，眼前放了一个成像平面（相当于真实屏幕），成像平面被我们化成不同的像素格子。对于每一个像素，我们可以从相机连一条线，穿过这个像素，这样就可以打出一根光线达到场景中。如果光线和场景的某一物体相交，那么连接交点和光源，看光源是否可见这个点，如果可见，那么就会形成一条有效的光路，进而可以计算这条光路上的能量，进行着色；如果不可见，则表示光线被沿途物体遮挡，也即该点在阴影里。



目前，我们只考虑眼睛是一个针孔摄像机，即眼睛是一个点，一个位置，不考虑实际相机的处理，以及镜头等（这部分会在路径追踪说）。对于场景中的物体，我们假设光打到它之后会发生完美的折射与反射。下图从眼睛开始，穿过成像平面的一个像素，投射一条光线（eye ray），这条光线会打到场景的某一个位置上，我们取最近的交点（这一步其实就解决了深度测试的问题）



当我们发现了一个点之后，我们要考虑这个点会不会被照亮。我们从这点到光源连一条线（shadow ray），如果可以连上就表示能被照亮（下图黑色箭头为法线）。有了法线，入射方向，出射方向，我们就可以做着色，写入像素的值，这时候可以用各种各样的着色模型，比如之前的 Blinn Phong。



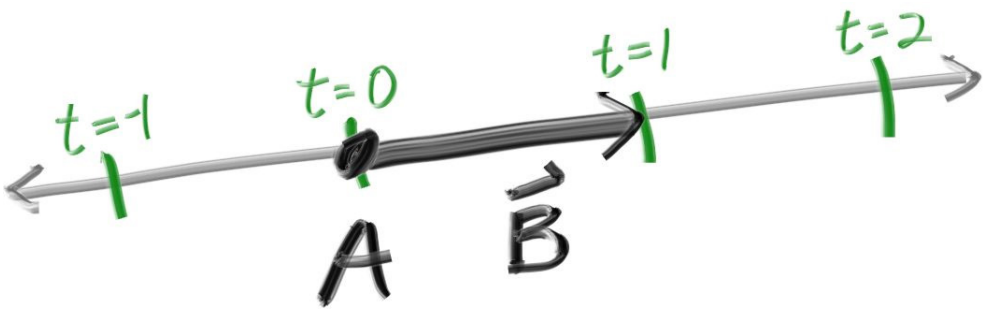
总结：光线投射做了这么一件事，每一个像素投出去一个光线，和场景相交求的话求最近交点，最近交点和光源连线，判定是否可见，然后算着色，写回像素的值。

光线

我们将光线定义为一个向量，该向量由起点和方向构成：

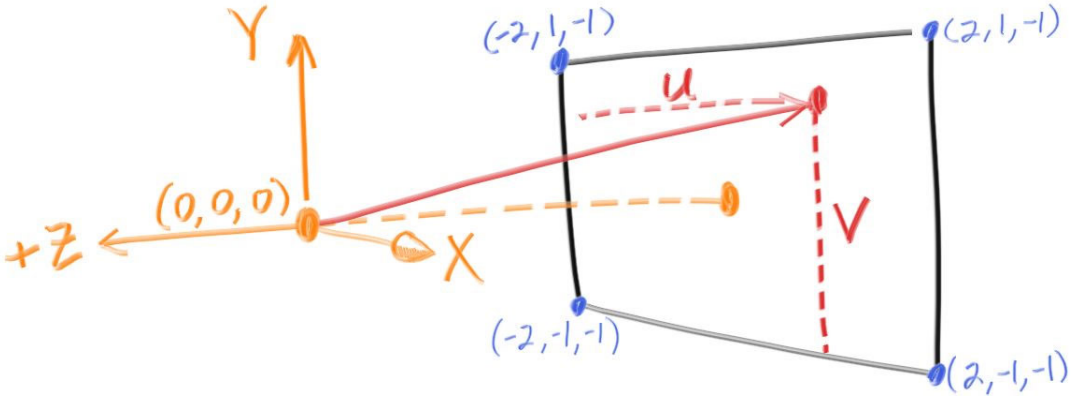
$$P(t) = \boldsymbol{a} + t \cdot \boldsymbol{b},$$

其中，P 是光线指向的目标位置，$\boldsymbol{a}$ 是光线的起点，$\boldsymbol{b}$ 是光线指向的方向，t 是光线的步长（$0 \leq t < \infty$）。如图所示：



照相机

本书默认的坐标系统：



坐标 $(0,0,0)$ 表示我们的眼睛（照相机）。

代码解读

main.cpp

主要关注 image、camera 部分。

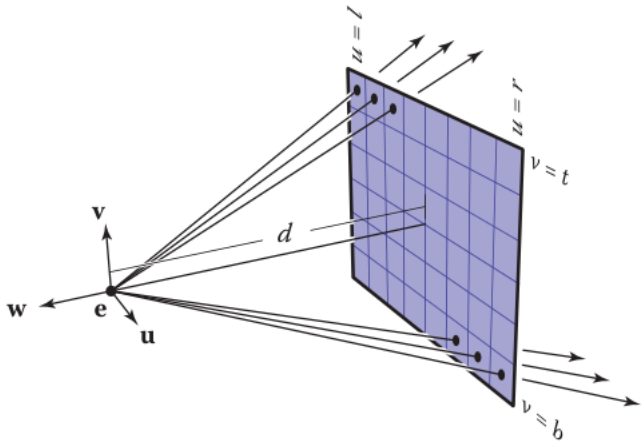
想象这么一个场景：乡下小屋，门前流水；屋外的鸟鸣吸引了你的注意，你朝窗前走去，看到屋外春意盎然，好不热闹。

此时你突然想起了第 11-13 和 16-18 这两处代码的区别是啥？

相同点：它们都在设置“窗子”的大小。

不同点：前者设置的是“画布”大小。画布相当于是你站在屋内看到窗外所有的景色范围大小（注意，你看到的景色范围远比你家的窗子大，这也解释为什么“画布”比视口大），也就是最终要画在真实屏幕上景色的大小。后者设置的是视口大小。视口相当于你的窗子，不难理解，看到景色，相当于打在景色上的光都通过窗子到达了你的眼睛（更进一步，你离窗子越近，看到的景色范围越大）。

总结，光线投射实际上是将光线投射在画布上，如下图：



Perspective projection

其中蓝色区域为图像物体显示范围，视口隐含在画布和眼睛之间，u，v，w 是相机坐标系，相机朝向 $-w$（这是一般规定，你也可以不遵守）。

在设计好画布和视口后，会分为以下几步来获得最终图像：

1. 确定分辨率，即画布大小 `image_with` 和 `image_height`；
2. 确定开始位置和终止位置，假定从下至上，混合颜色为白色和蓝色；
3. 确定视线（光线），即从眼睛（照相机）出发到视口像素中心点的向量（光线的起点为眼睛（照相机）的坐标，方向为从眼睛指向像素中心）；
4. 从画布到视口做一个映射：先转换位标准坐标，变成 $[0,1]$，再乘以步长，变为视口大小；
5. 确定比例系数 t。

18 行：设置焦点，简单理解为照相机（眼睛）到视口的距离；
20-22 行：设置照相机原点、视口 x 轴长度、视口 y 轴长度；
23 行中的 `lower_left_corner` 表示视口的左下角，可以这么理解：

$$\begin{aligned}\texttt{lower\_left\_corner} &= \boldsymbol{O'} - \frac{1}{2}\texttt{horizontal} - \frac{1}{2}\texttt{vertical} \\ &= \texttt{origin} - (0,0,1) - \frac{1}{2}\texttt{horizontal} - \frac{1}{2}\texttt{vertical}\end{aligned}$$

33 行：位于原点的光线从左至右，从上至下，逐一扫描每一个像素点。

代码运行

可使用第一节介绍的方法，也可以直接双击 `run.bat` 文件。

参考文献

[1] https://zhuanlan.zhihu.com/p/128582904

[2] https://www.cnblogs.com/lv-anchoret/p/10163205.html