图形学大作业：手动实现光线追踪

手动实现RayTracing

- 这份作业实现了简单的光线追踪算法。实现标准基于C++ 14，无其他任何依赖项（例如opengl）。源代码在source文件夹中。
- 运行环境：Windows10
- 编译方式：在vs2017及以上版本中导入source文件夹中的全部文件，编译即可。
- 运行方式：将输出重定向到文件中，例如：raytracing.exe > sample.ppm
- 若需增删各几何形体、光源等，需在main.cc中修改完毕后，再重新编译。
- 参考资料：Ray Tracing in a Weekend，Games101等

立方体

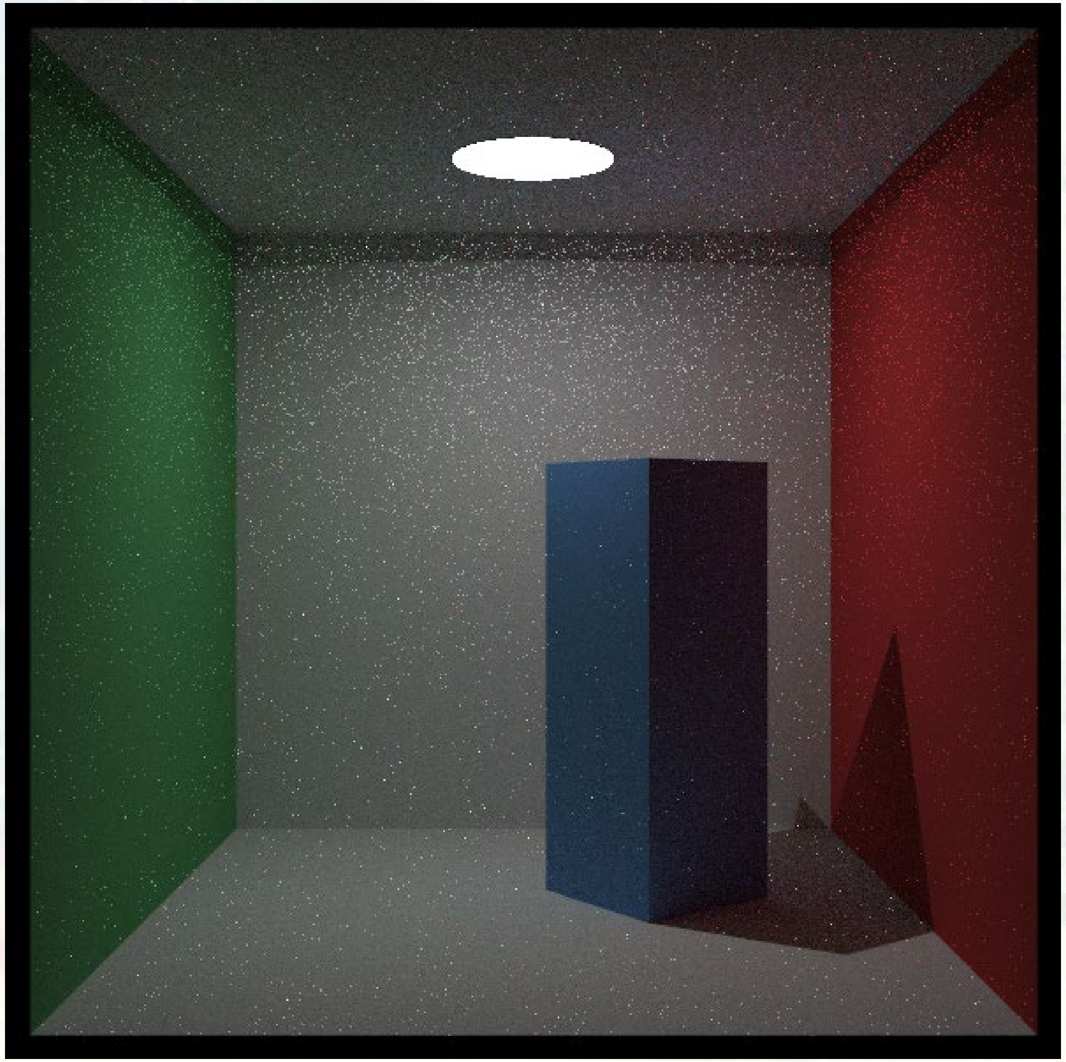
平面

设Camera发出的射线（Ray）方程为$A + tB$，其中，A为射线的源点，B为射线的方向，t是长度参数。

假设平面上一点为P_0，法向量为$\boldsymbol{n}$，那么平面上任意一点P都满足$(P - P_0) \perp \boldsymbol{n}$，即$(P - P_0) \cdot \boldsymbol{n} = 0$。代入射线的参数方程，得到$(A + tB - P_0) \cdot \boldsymbol{n} = 0$，解得$t = -\frac{(A - P_0) \cdot n}{B \cdot n}$。

然后判断t是否在范围(t_{min}, t_{max})中，即可求出射线与平面的交点，交点的法向量为n。

使用6个平面可以构成一个立方体，类比平面中直线和矩形求交，空间中光线所在的直线与立立方体所有面所在平面至多有6个交点。设光线与xOy, yOz, zOx平面的交点参数分别为$t_1, t_2, t_3, t_4, t_5, t_6$，那么与立方体最终的交点参数为$\max\{\min\{t_1, t_2\}, \min\{t_3, t_4\}, \min\{t_5, t_6\}\}$，最后再判断这个参数是否大于0且在某个矩形中。



球

球体

设圆心为C，半径为R。那么射线与球面的交点满足

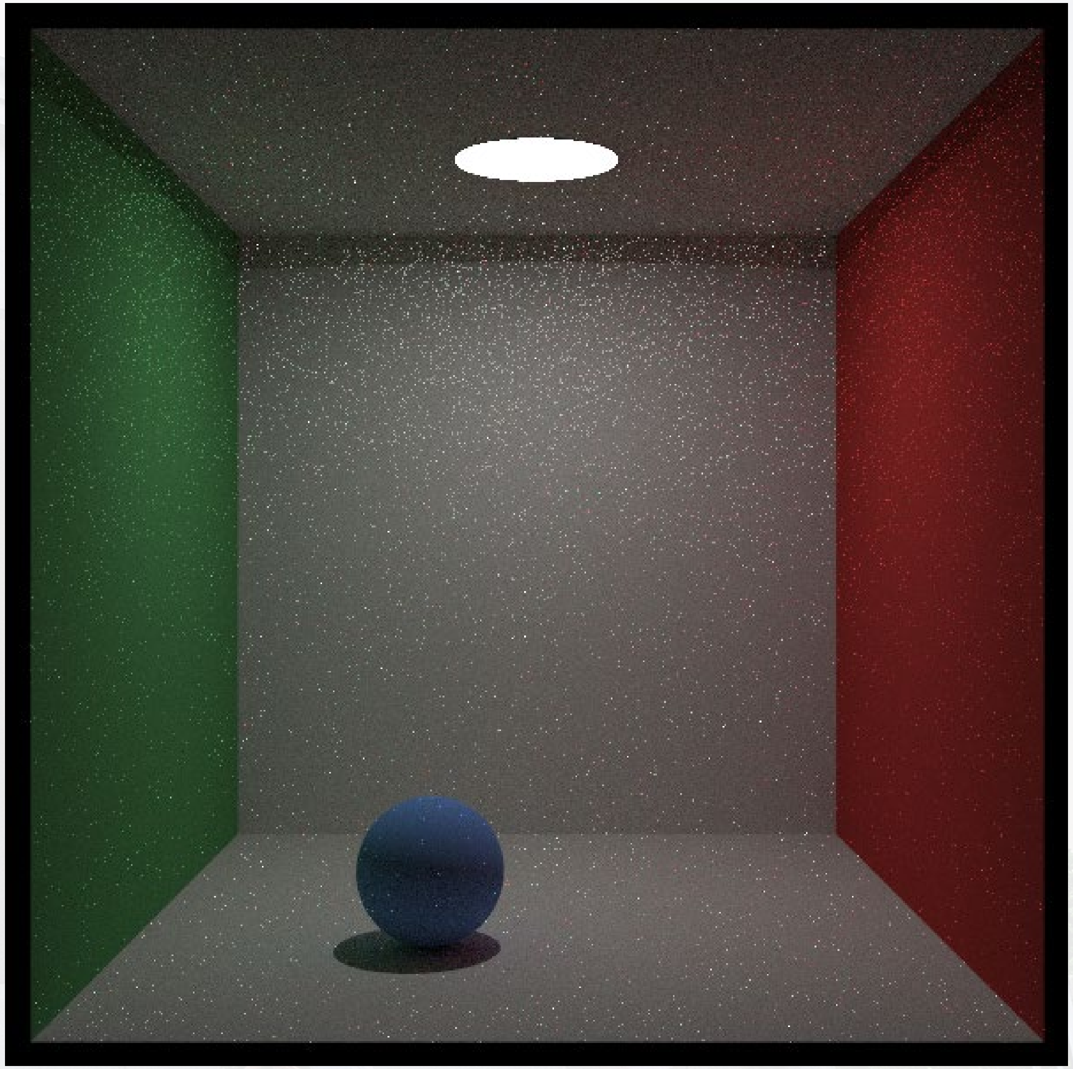
$$(A_x + tB_x - C_x)^2 + (A_y + tB_y - C_y)^2 + (A_z + tB_z - C_z)^2 = R^2$$

即

$$(A + tB - C) \cdot (A + tB - C) = R^2 \Rightarrow (B \cdot B)t^2 + 2B \cdot (A - C)t + (A - C) \cdot (A - C) - R^2 = 0$$

当该一元二次方程无解时，球面与射线无交点。

假设该方程存在解，设为t_1, t_2，那么t_1, t_2中在区间(t_{min}, t_{max})中的最小者为交点对应的参数。设交点为P，那么法向量为$P - C$。

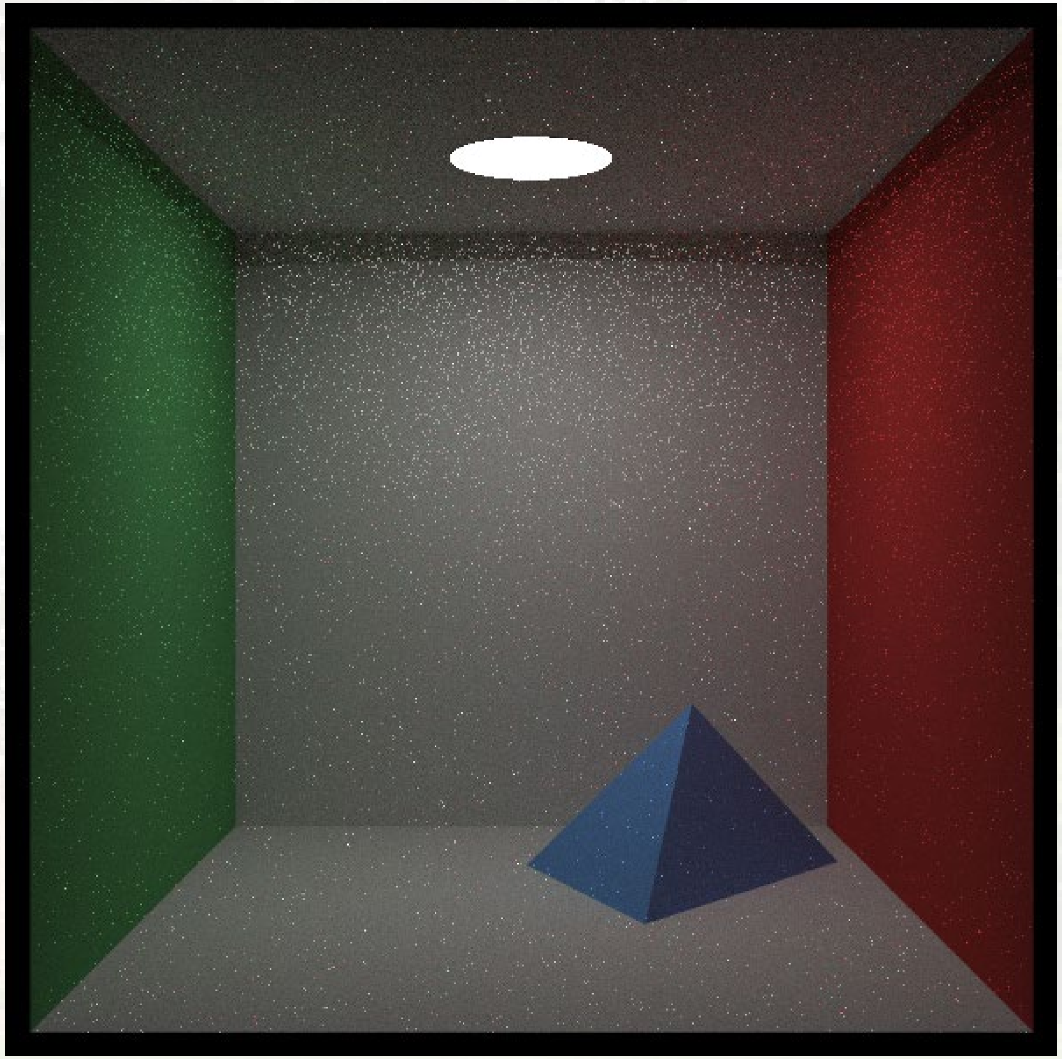


三棱锥

三角形和三棱锥

对于三角形的实现，首先可以根据光线和平面的求交法算出光线和三角形所在平面的交点P，然后通过叉乘法来判断点P是否位于ΔABC中，以及计算出三角形的法向量n。

使用四个顶点构建四个三角形面片，进而可以求出交点和对应的法向量。



圆柱体

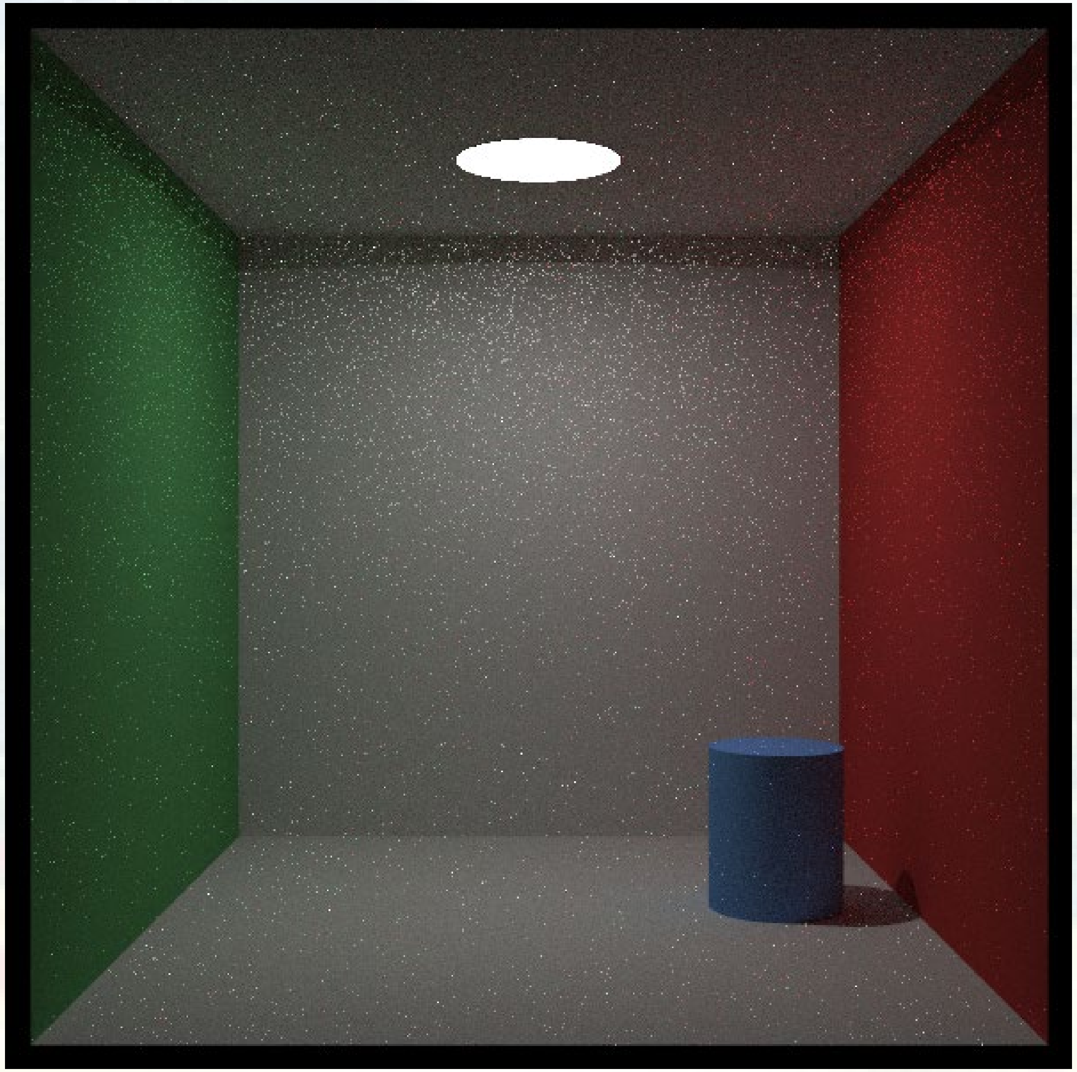
圆柱体

设圆柱体底面和顶面的圆心分别为O_1,O_2，半径为R，由此可以求出底面指向顶面的单位向量$\boldsymbol{n}=\frac{\overrightarrow{O_1O_2}}{||\overrightarrow{O_1O_2}||}$。如果光线与圆柱体的侧面相交，交点为$C$，那么不难发现：

$$|\overrightarrow{O_1C}\times\boldsymbol{n}|=|\overrightarrow{O_1C}|\times|\boldsymbol{n}|\times sin\theta=R$$

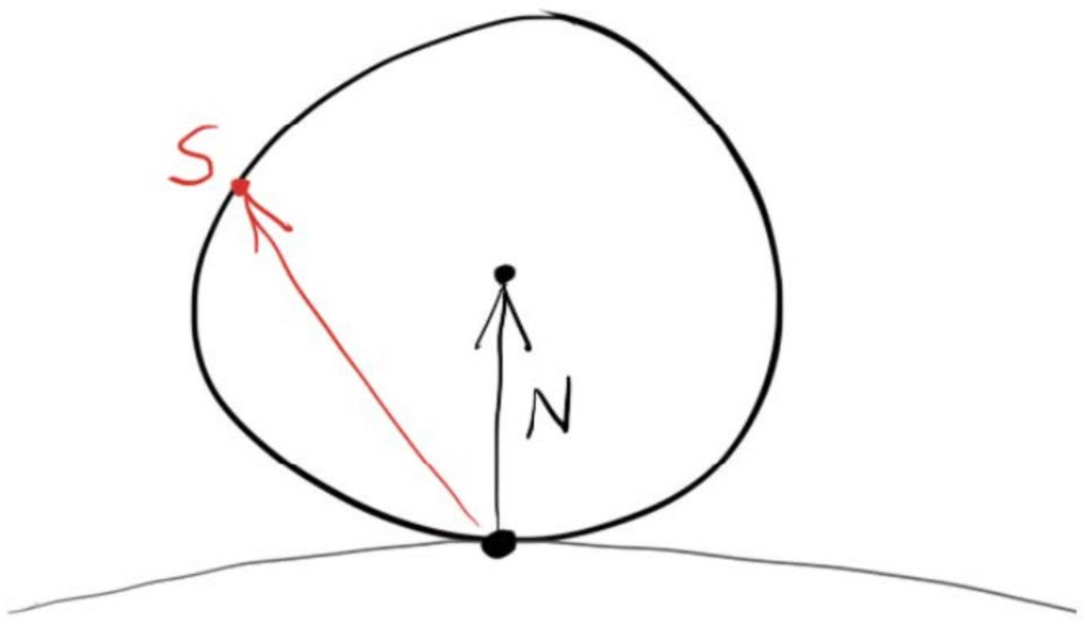
交点可以表示为$C=P+tD$，其中P为光线源点，D为光线方向。带入即可求出光线与圆柱体侧面的两个可能交点t_1和$t_2(t_1\le t_2)$。若$0<t_1\le t_2$，则t_1为可能的交点。若$t_1<0<t_2$，那么t_2为可能的交点，否则无解。然后通过$|\overrightarrow{O_1C}\times\boldsymbol{n}|>0$且$|\overrightarrow{O_2C}\times(-\boldsymbol{n})|>0$来判断交点是否在圆柱体的高度范围内。

综合判断光线与圆柱体上下地面的交点，即可求出光线与圆柱体的交点，进而求出交点的法向量。

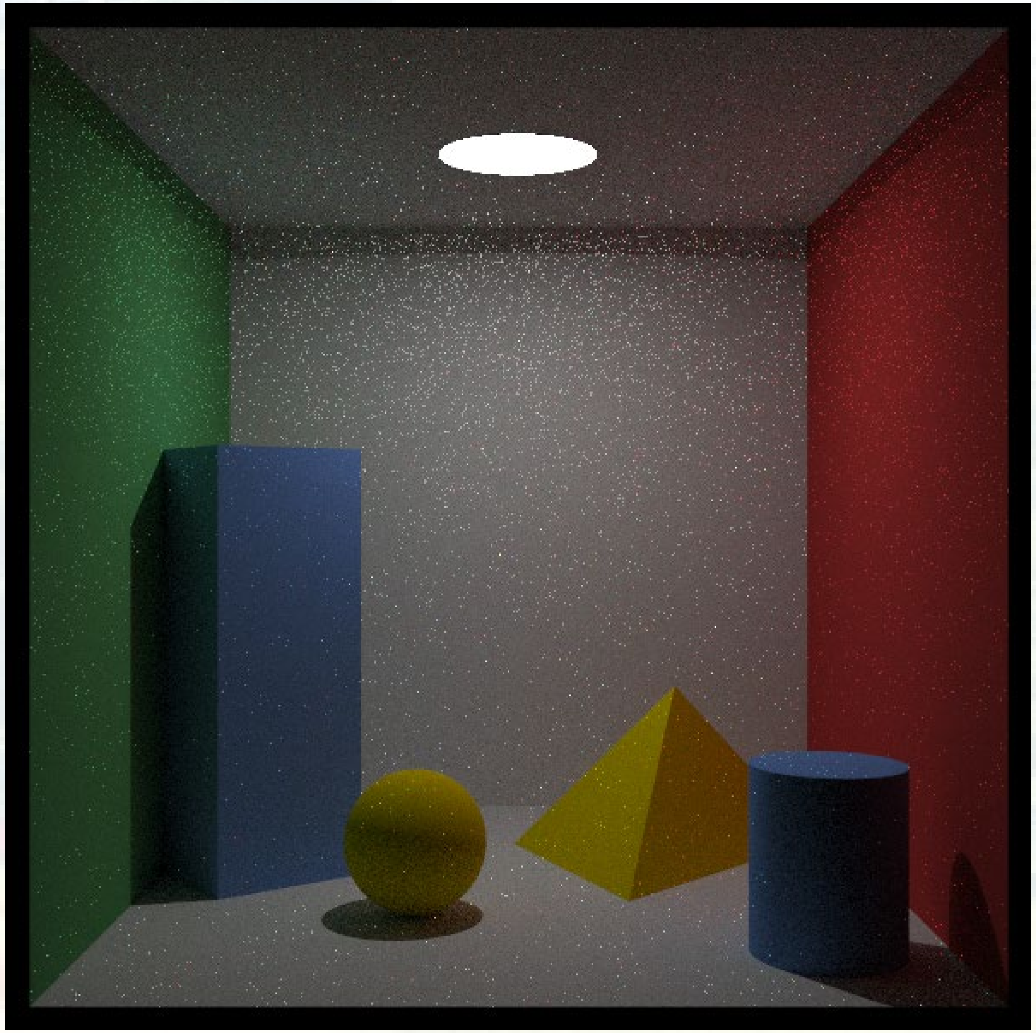


Lambertian

漫反射和Lambertian材质

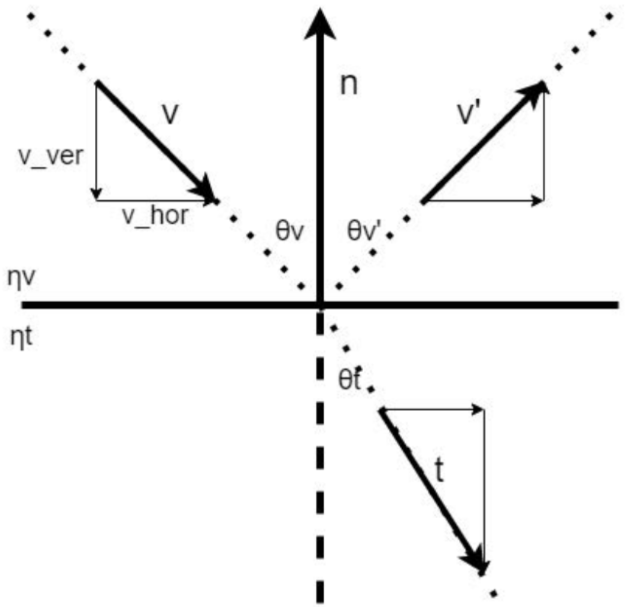


在光线与物体表面交点法向量（法向量向外）处生成一个单位球，在这个球面生成随机的一个点，并与交点相连作为一个向量，这个向量就是出射的光线的向量。再考虑光线吸收等特性，即可得到Lambertian漫反射材质。



Metal

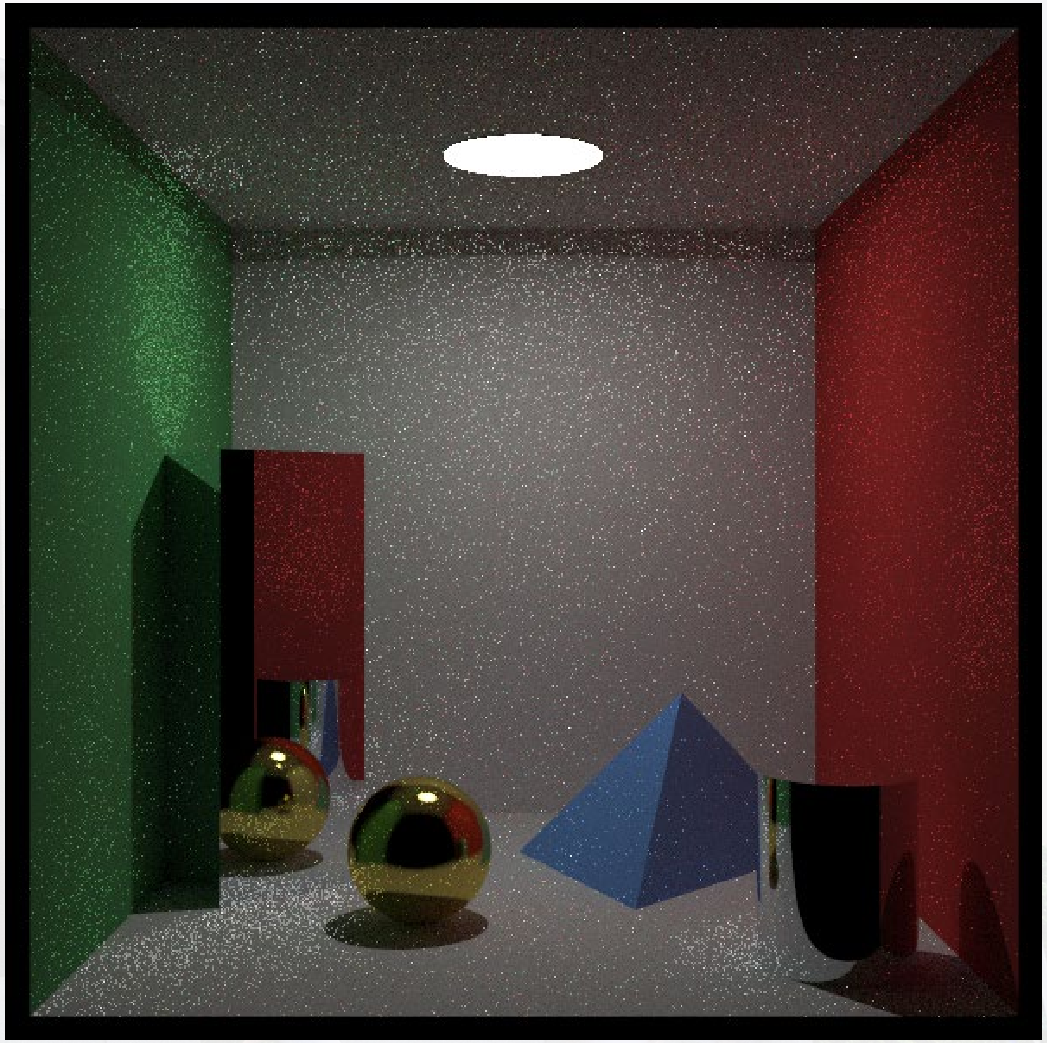
反射和Metal材质



如图所示，入射光线为v，出射光线为v'。入射光线可以分解为垂直于平面和平行于平面的$v_\perp$和$v_{||}$。平面的法向量为n。入射光线在法向量方向的投影为$(v\cdot n)n$，注意到此向量的方向和法向量的方向是相反的。因此出射光线在法向量方向的投影为$-(v\cdot n)n$。那么出射光线为

$$v' = v'_{||} + v'_{\perp} = v_{||} - (v\cdot n)n = v_{||} + (v\cdot n)n - 2(v\cdot n)n = v_{||} + v_{\perp} - 2(v\cdot n)n = v - 2(v\cdot n)n$$

由此加上光线吸收等特性，可以得到金属材质。



Dielectric

折射和Dielectric材质

由斯涅尔定律$\eta_v sin\theta_v = \eta_t sin\theta_t \Rightarrow sin\theta_t = \frac{\eta_v}{\eta_t} sin\theta_v$，需要注意约束$sin\theta_v \le \frac{\eta_t}{\eta_v}$。

变形得到$|t_{||}| = \frac{\eta_v}{\eta_t}|v_{||}|$。由于二者在同一方向，那么$t_{||} = \frac{\eta_v}{\eta_t}v_{||} = \frac{\eta_v}{\eta_t}(v - v_\perp)$。

又存在$|t_{||}|^2 + |t_\perp|^2 = |t|^2$，求出$|t_\perp| = \sqrt{1-|t_{||}|^2}$（考虑$t, v$都是单位向量），那么 $t_\perp = -\sqrt{1-|t_{||}|^2}n$。

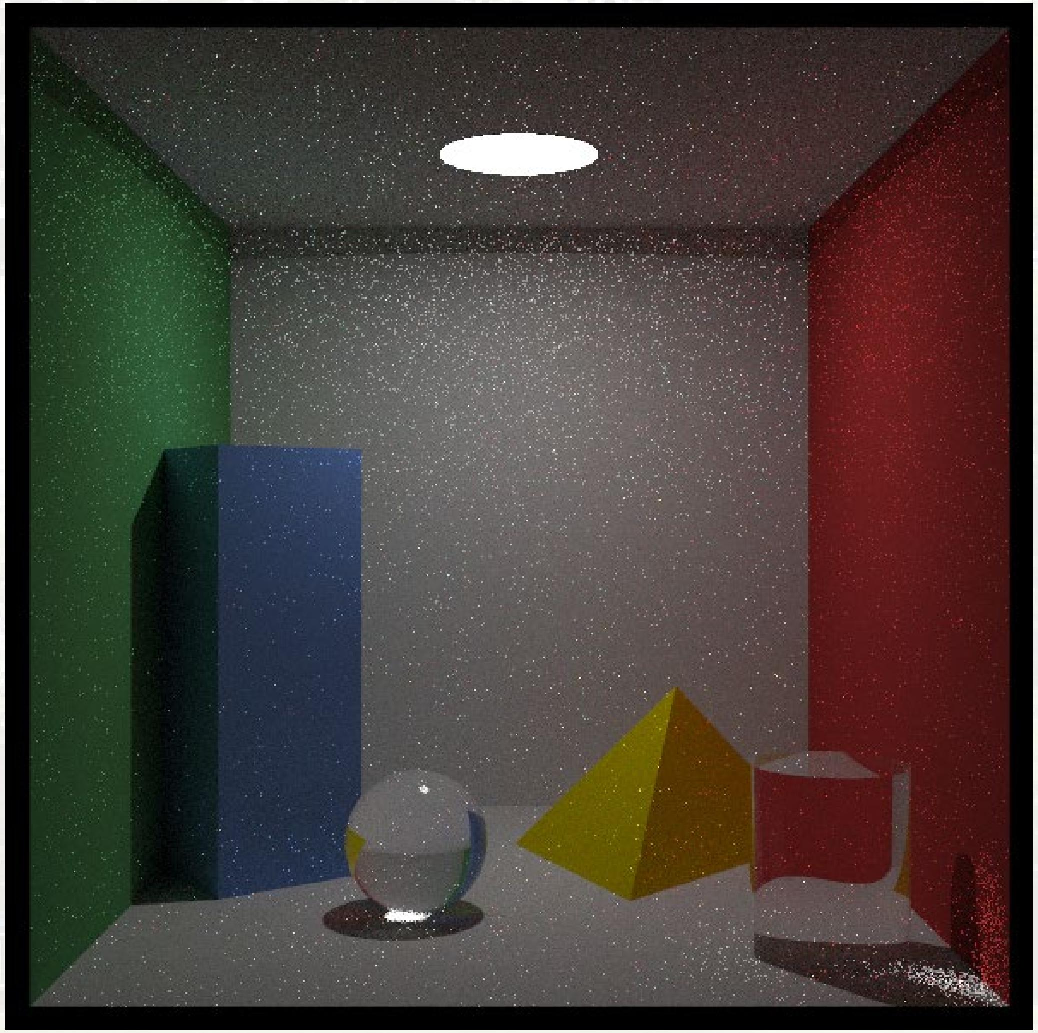
$t_{||} + t_\perp = \frac{\eta_v}{\eta_t}(v - v_\perp) - \sqrt{1-|t_{||}|^2}n = \frac{\eta_v}{\eta_t}v - \frac{\eta_v}{\eta_t}(v \cdot n)n - \sqrt{1-|t_{||}|^2}n$

由于t是单位向量，那么$|t_{||}|^2 = sin^2\theta_t = (\frac{\eta_v}{\eta_t})^2 sin^2\theta_v$。

代入上式得

$$\begin{aligned} t_{||} + t_\perp &= \frac{\eta_v}{\eta_t}v - \frac{\eta_v}{\eta_t}(v \cdot n)n - \sqrt{1-(\frac{\eta_v}{\eta_t})^2 sin^2\theta_v}n \\ &= \frac{\eta_v}{\eta_t}v - \frac{\eta_v}{\eta_t}(v \cdot n)n - \sqrt{1-(\frac{\eta_v}{\eta_t})^2 (1-cos^2\theta_v)}n \\ &= \frac{\eta_v}{\eta_t}v - \frac{\eta_v}{\eta_t}(v \cdot n)n - \sqrt{1-(\frac{\eta_v}{\eta_t})^2 (1-(v \cdot n)^2)}n \end{aligned} \tag{1}$$

由此可以直接实现Dielectric玻璃材质。



Microfacet BRDF

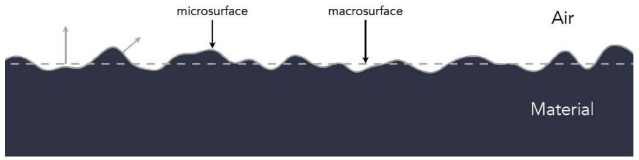
Microfacet BRDF

这部分的实现比较困难，参考了Games101的讲解，[Coding Labs :: Physically Based Rendering - Cookâ€"Torrance](#)这篇博客和Microfacet Models for Refraction through Rough Surfaces这篇论文。

考虑Cook-Torrance BRDF方程：

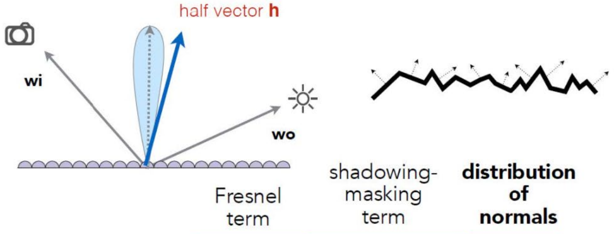
$$f_r = k_d f_{lambert} + k_s f_{cook-torrance}$$

前者表示macrosurface，即平坦但粗糙的表面，后者表示microscale，即起伏但光滑的表面，k_d和k_s分别表示它们的颜色，实际实现中，二者是相同的。



$f_{lambert}$的计算可以直接搬运Lambertian材质。

$f_{cook-torrance}$比较复杂，实现思路如下



$$f(\mathbf{i},\mathbf{o}) = \frac{\boxed{\mathbf{F}(\mathbf{i},\mathbf{h})}\boxed{\mathbf{G}(\mathbf{i},\mathbf{o},\mathbf{h})}\boxed{\mathbf{D}(\mathbf{h})}}{4(\mathbf{n},\mathbf{i})(\mathbf{n},\mathbf{o})}$$

F表示的是菲涅尔项，F用于模拟光以不同角度与表面相互作用的方式，实际上用的是Shlick近似，η_1和η_2为两种介质的折射系数：

$$F = F_0 + (1-f_o)(1-cos\theta)^5, F_0 = (\frac{\eta_1-\eta_2}{\eta_1-\eta_2})^2$$

几何函数G用于描述由于微小平面相互阴影而导致的光衰减。这又是一个统计近似值，它模拟了在给定点微面被彼此遮挡的概率，或者光线在多个微面上反弹，在此过程中失去能量，然后到达观察者的眼睛。"参考了GGX函数的实现：

$$G(\omega_i,\omega_o,m,n,\alpha) = G_p(\omega_o,m,n,\alpha)G_p(\omega_i,m,n,\alpha)$$

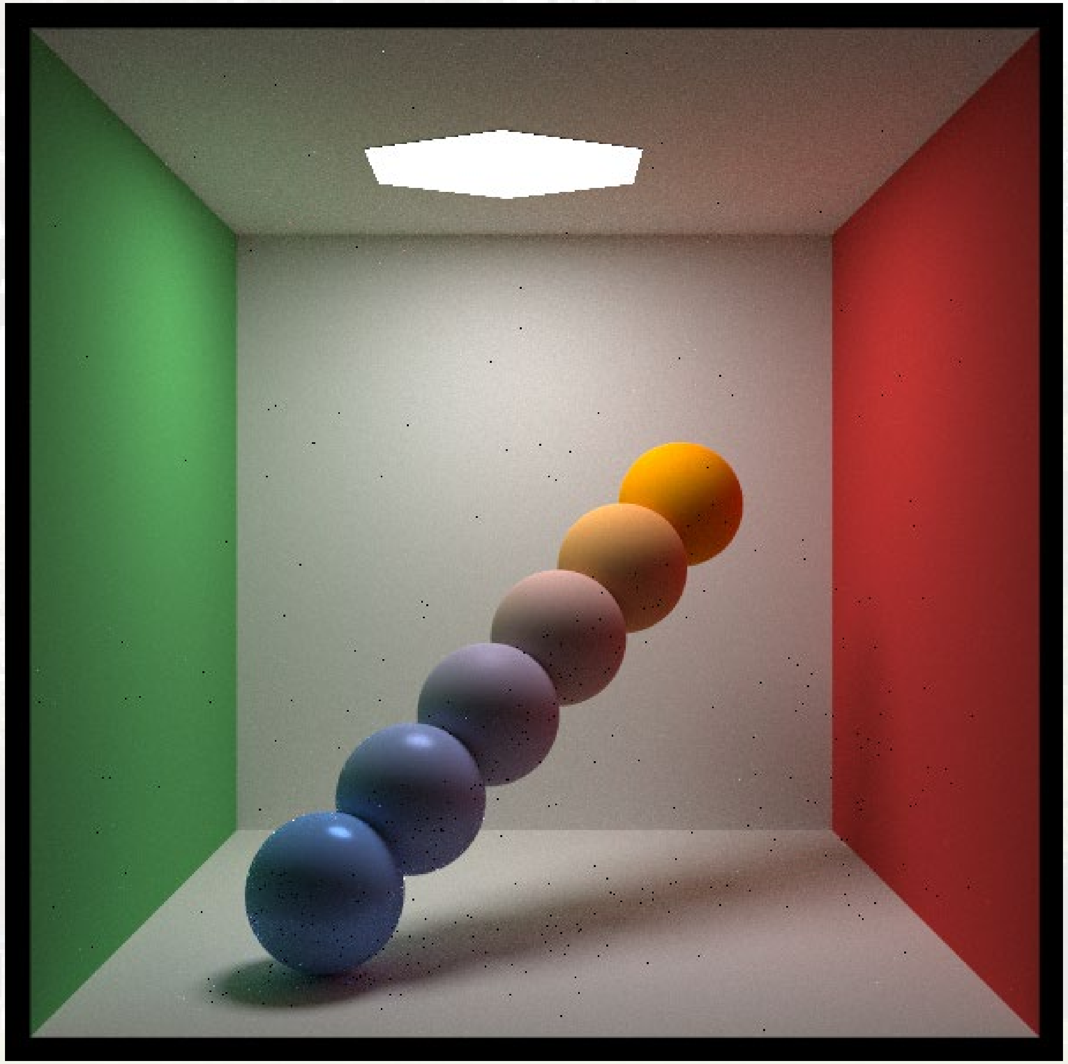
$$G_p(\omega,m,n,\alpha) = \chi(\frac{\omega\cdot m}{\omega\cdot n})\frac{2}{1+\sqrt{1+\alpha^2\frac{1-(\omega\cdot m)^2}{(\omega\cdot m)^2}}}$$

法向分布函数D用于描述微观面在某个给定点的统计方向。例如，如果 20% 的分面朝向某个向量 m，则将m 馈送到分布函数中将得到 0.2。文献中有几个函数来描述这种分布（例如 Phong 或 Beckmann），但是我们将用于分布的函数将是 GGX，其定义如下：

$$\chi(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x \le 0 \end{cases}$$

$$D(m,n,\alpha) = \frac{\alpha^2\chi(h\cdot n)}{\pi((m\cdot n)^2(\alpha^2+tan^2(\theta_m))^2} = \frac{\alpha^2\chi(h\cdot n)}{\pi((m\cdot n)^2(\alpha^2+(\frac{1-(m\cdot n)^2}{(m\cdot n)^2}))^2}$$

其中α表示物体表面的粗糙程度，范围在$[0,1]$之间，可以调节。

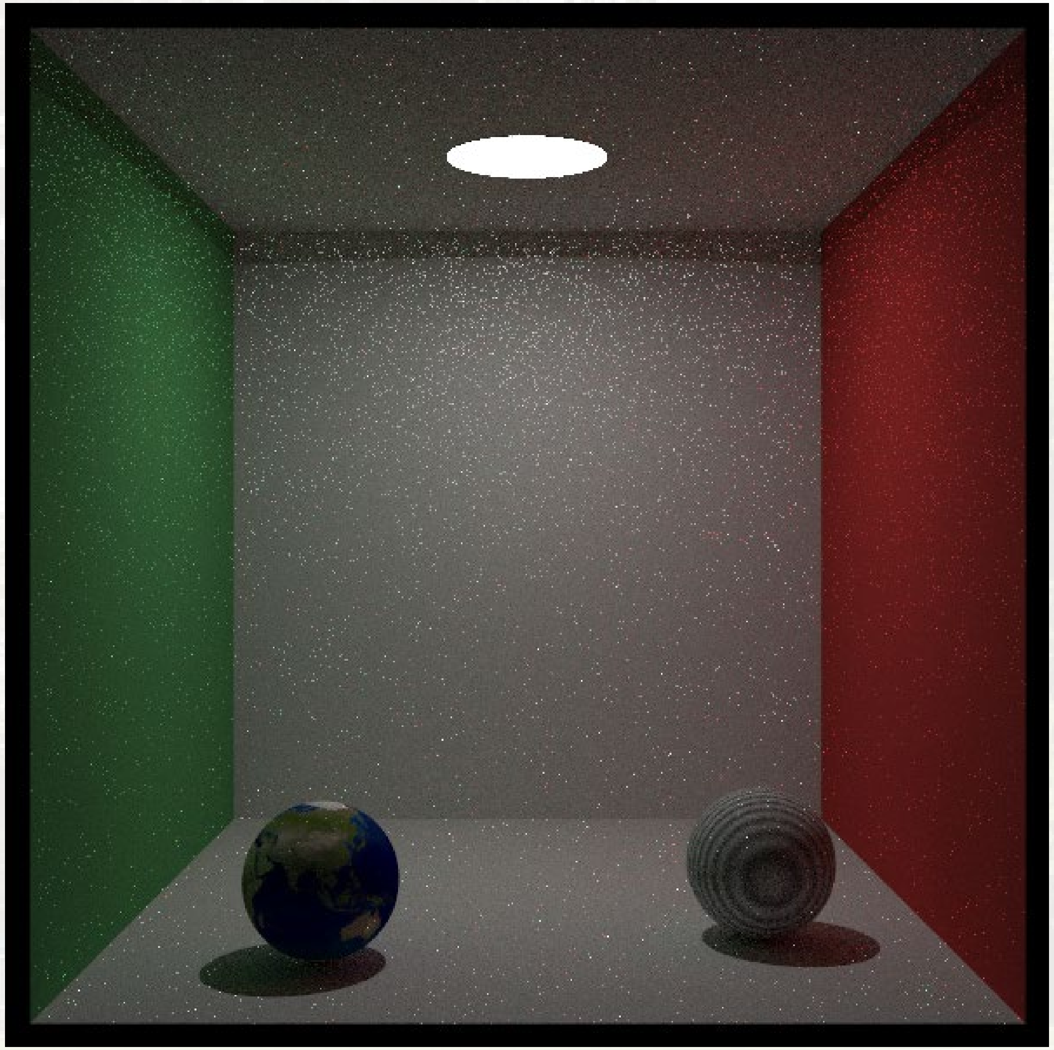


纹理

纹理

物体表面的点是三维的，可以通过各种坐标转换方式作二维映射，得到一组新的坐标(u, v)，这对应着二维纹理贴图（例如地球贴图）的坐标。

对于噪声纹理，同样也是讲三维的点坐标转换为表示颜色的值。



光源和软阴影

三角形光源和多边形光源

多边形光源可以拆分成多个三角形，因此问题简化为三角形光源。

在实际的实现中有50%的概率对光源进行采样，余下的50%按照材质的性质计算反射光线的方向。

设三角形的三点分别为A,B,C，则采样的方程为：

$$
\begin{gathered}
\xi_1 = random\_uniform(0,1) \\
\xi_2 = random\_uniform(0,1) \\
u = 1 - \sqrt{\xi_1} \\
v = \xi_2\sqrt{\xi_1} \\
P = u * A + v * B + (1 - u - v) * C
\end{gathered}
$$

圆形光源

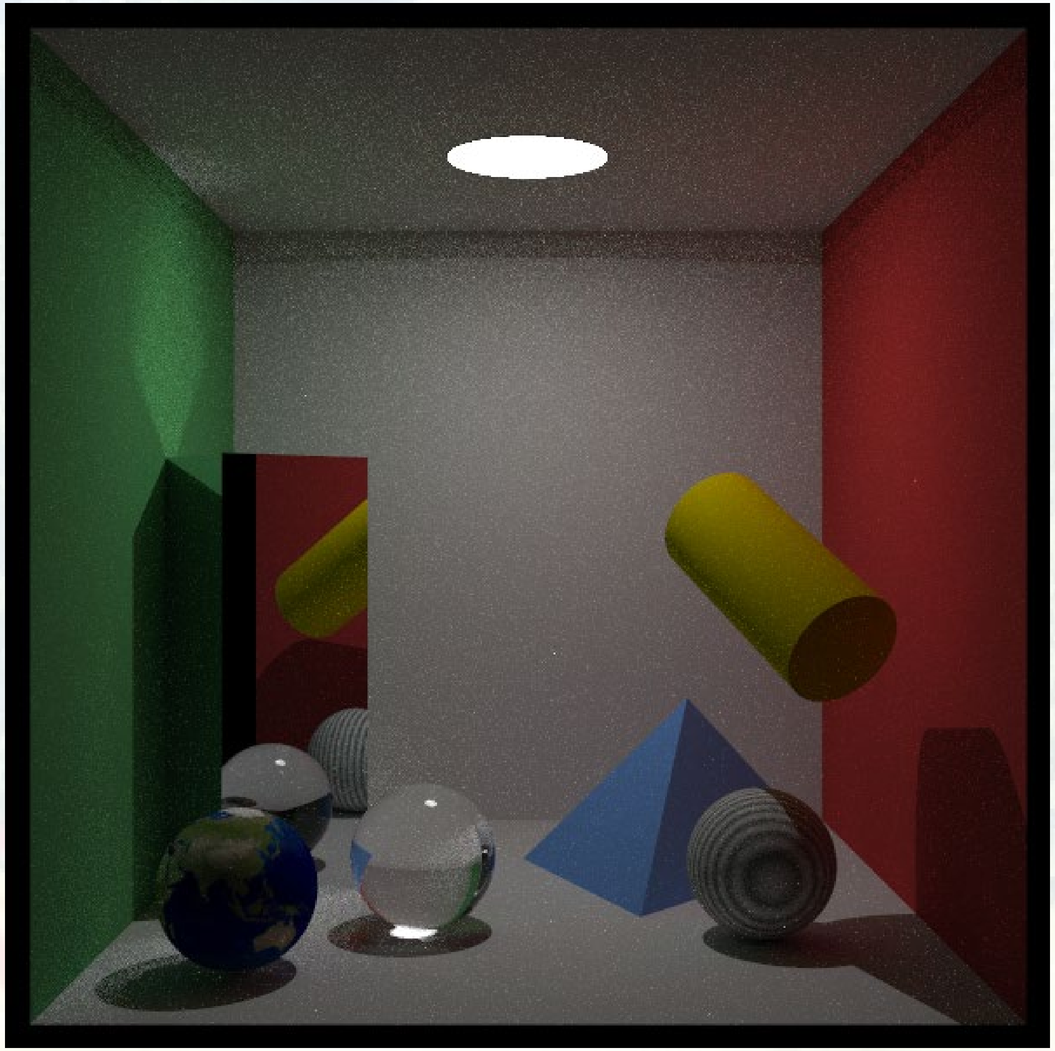
仅考虑XZ方向的圆形。设C为圆心，R为半径。

采样的方程：

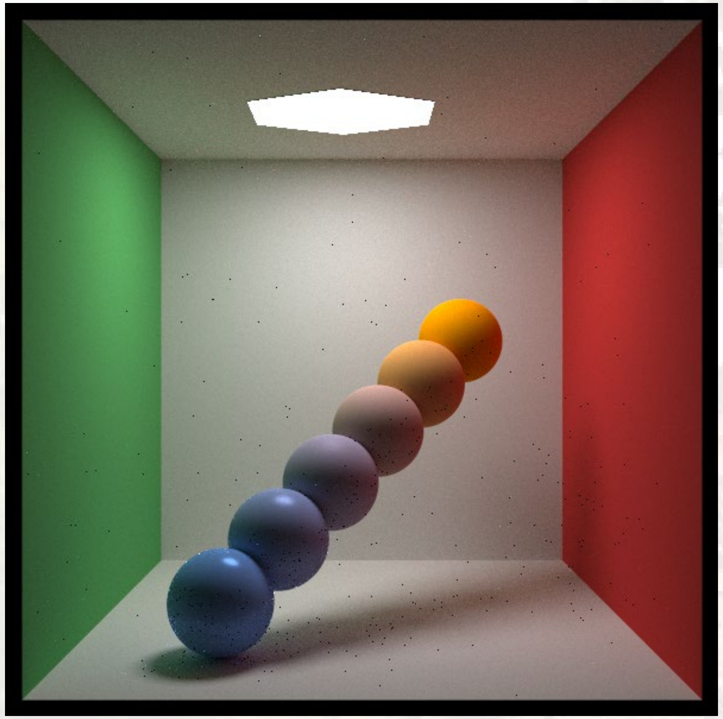
$$
\begin{gathered}
\rho = \sqrt{random\_uniform(0,1)} \\
\theta = random\_uniform(0,2\pi) \\
P = (C.x + \rho cos\theta, C.y, C.z + \rho sin\theta)
\end{gathered}
$$

环境光

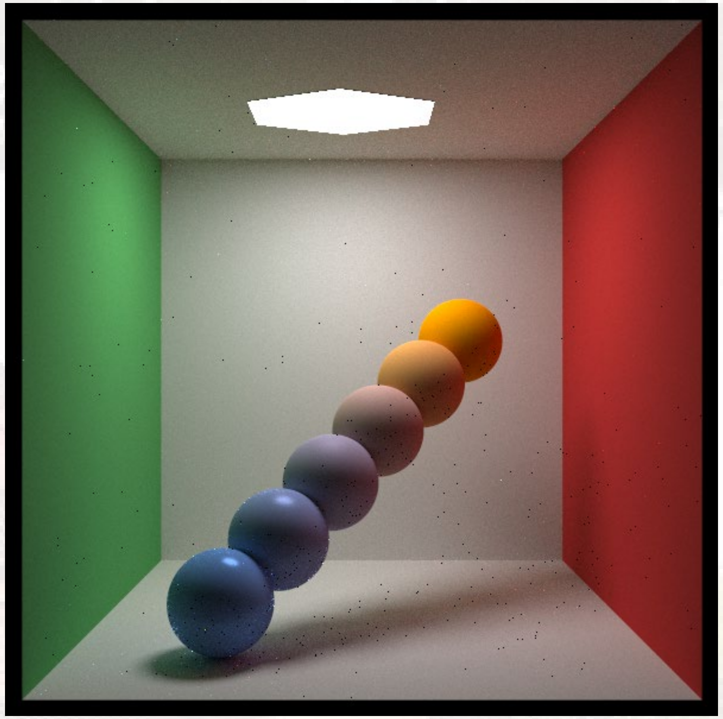
环境光可以视作长和宽为无穷大的矩形，因此当射线不与任何物体相交或和设置的无限大矩形光源相交时，直接返回光源的光强。



采样



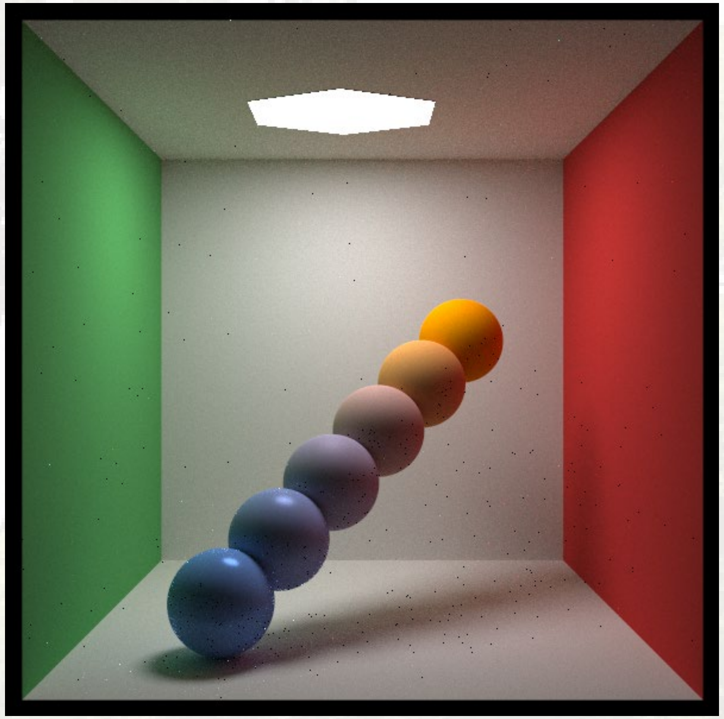
Random



Hammersley采样

Hammersley采样用于构造均匀分布的2D随机采样点，其原理是通过对二进制数进行Radical Inverse，其采样点得到的坐标为$p_i = (x_i, y_i) = (i/N, \phi(i))$。其中$N$是每个像素点中采样点的数量，$\phi(i)$计算如下

```
float HammersleyPhi(int i)
{
    float x = 0.0;
    float f = 0.5;
    while (i != 0)
    {
        x += f * (i & 1);
        f *= 0.5;
        i /= 2;
    }
    return x;
}
```



Jittered采样

将每个像素均匀划分为若干个正方形，在每个正方形中随机采样。