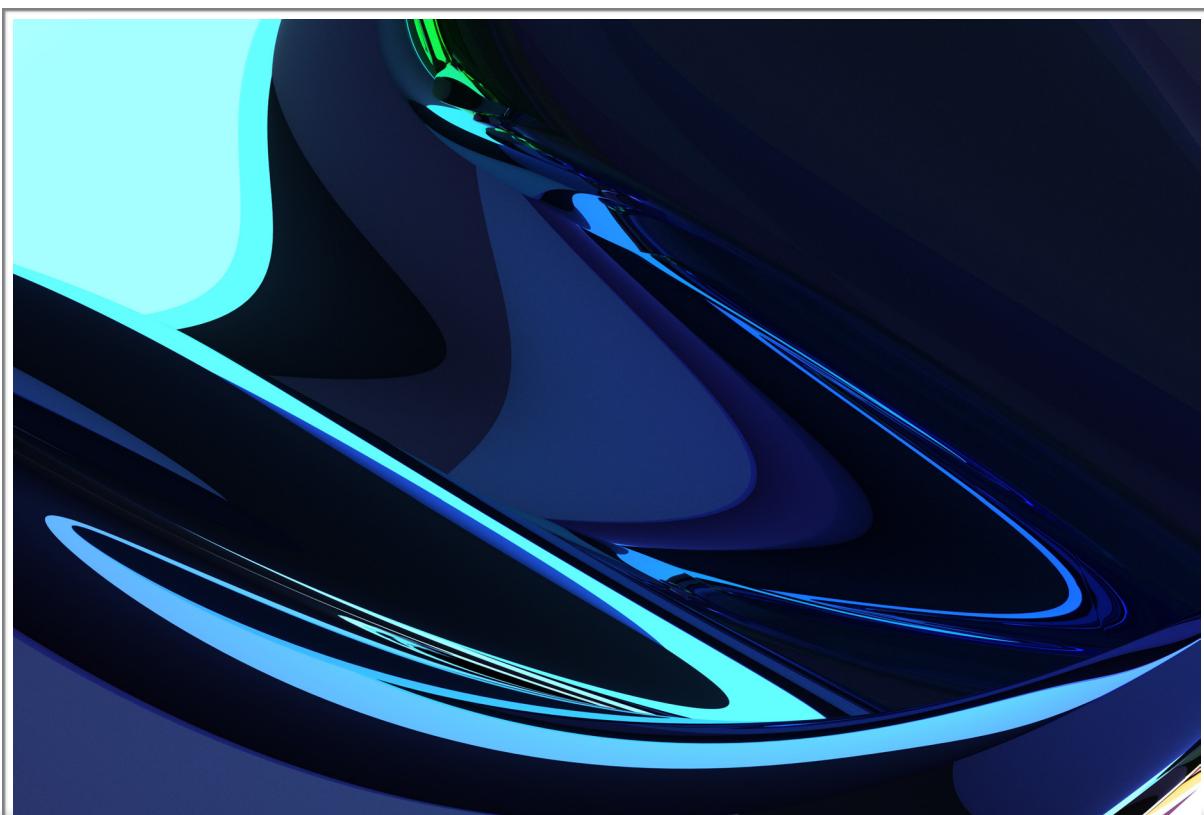


图形学实验报告：三维渲染

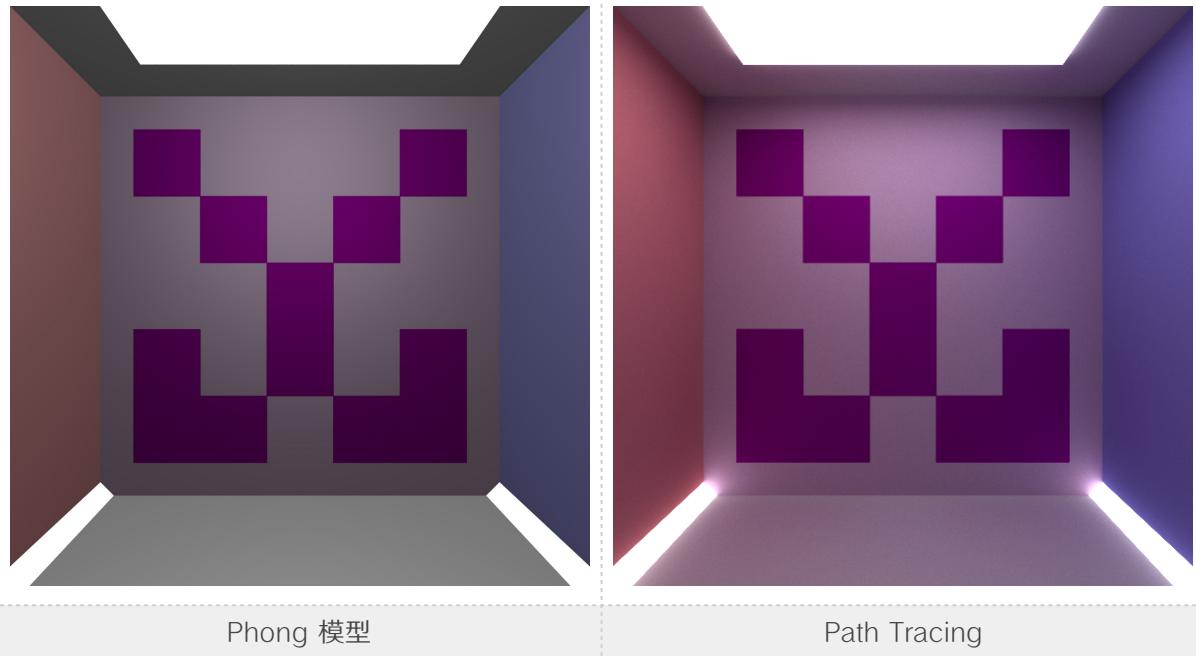
涂轶翔 2017011422



功能概述

- ❖ 项目主要使用 **Path Tracing** 完成，而后在 PT 框架上实现了 **Photon Mapping**
- ❖ 实现了三角面、矩形面、球体、复杂网格、曲面的渲染，全部位于 object/ 中，分别有继承自统一接口 Object 的类型
- ❖ 光线进行方向包括漫反射、光滑镜面反射、粗糙镜面反射（类似金属效果）、菲涅尔效应折射、粗糙折射（类似毛玻璃效果）
- ❖ PT 的渲染支持**景深**、**软阴影**、**抗锯齿**、**纹理和法向贴图**，PM 的光子追踪支持**色散**效果和**动画渲染**
- ❖ **网格绘制**使用包围盒、**kd-tree** 进行加速，支持**纹理贴图**、**法线插值**，同时会根据渲染情况移除不需要的三角面
- ❖ **曲面绘制**支持**任意可求导参数曲面**（不局限于 Bézier 曲面或旋转曲面），基于构建网格求初值，然后阻尼牛顿迭代求精确解，同样使用包围盒、kd-tree 进行加速

Path Tracing



相比于 Phong 模型，蒙特卡洛光线追踪算法可以实现全局光照，几乎可以无偏模拟真实场景。

原理

从摄像机向每个像素的方向发出射线，在每一处交点计算物体发光强度以及颜色，然后以随机转盘的方式选择下一步光线方向

- 漫反射：随机选择一个方向，更倾向于仰角更大的方向，以符合物理上“仰角更大，则单位面积接收光强更强”
- 镜面反射：根据法向计算确定的镜面反射方向
- 折射：根据法向和折射公式确定折射方向，根据菲涅尔公式确定折射比例（算法上体现为概率）

效果

- 全局光照：可以让许多物体同时发光而不影响其渲染效率
- Color Bleeding：漫反射物体所反射的光也会影响到其他物体
- 软阴影：可以很好地模拟大面积光源，形成软阴影
- 抗锯齿：摄像机在每个像素点范围内沿随机方向发出射线，从概率上相当于无限倍采样抗锯齿

网格求交

三角面求交

参考: <https://www.cnblogs.com/samen168/p/5162337.html>

求解线性方程组 $\mathbf{s} + t \cdot \mathbf{d} = \mathbf{o} + u \cdot \mathbf{e}_1 + v \cdot \mathbf{e}_2$ ，其中 \mathbf{s} 为射线起始位置， \mathbf{d} 为射线方向， \mathbf{o} 为三角形一顶点， \mathbf{e}_1 和 \mathbf{e}_2 为三角形的两条边。满足 $t, u, v > 0, u + v \leq 1$

方程可化为 $[-\mathbf{d} \ \mathbf{e}_1 \ \mathbf{e}_2] \cdot [t \ u \ v]^T = \mathbf{s} - \mathbf{o}$ ，利用克莱姆法则求解

包围盒

为了简化计算，采用平行于坐标轴的包围盒。计算时，则只需要分别考虑三个坐标方向：包围盒是三个有限区间，射线是三个单测无限区间（或者如果方向分量为 0 则为单点）。如果在某一坐标方向上二者没有交集，则说明射线与包围盒不相交；如果三个坐标方向上射线源点都在闭区间内，则说明射线与包围盒相交；其他情况，则找到射线最晚与包围盒相交的坐标方向，然后计算该点是否位于包围盒表面。

以上算法同时还能够计算出射线与包围盒的最短距离，如果最短距离长于目前已知的另一个交点便可以放弃该包围盒。

kd-tree (非标准)

对于三个坐标方向分别进行以下操作：

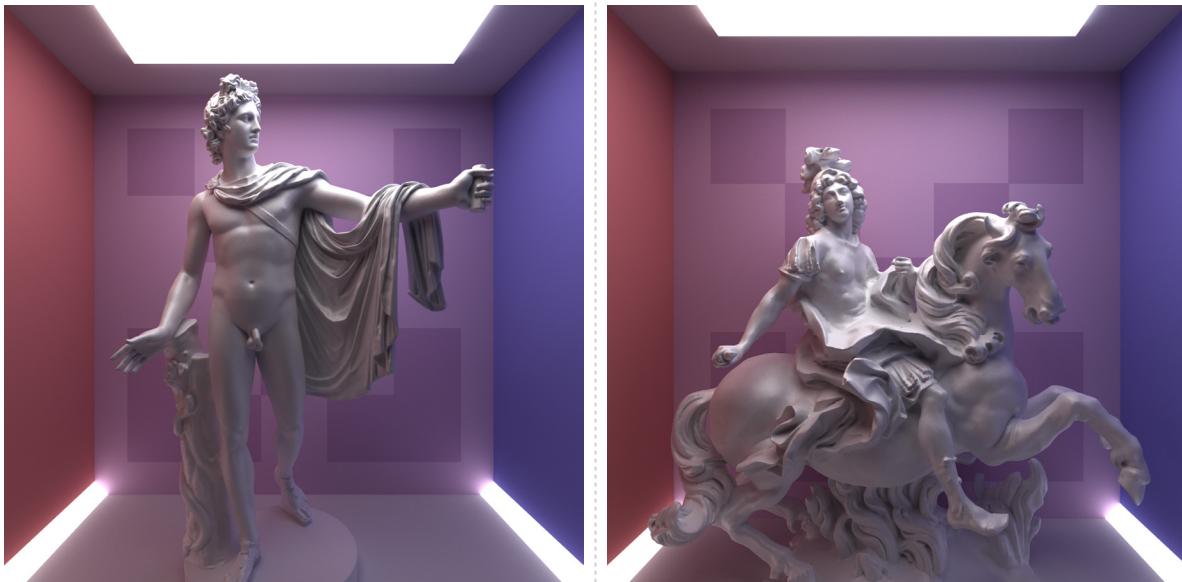
- 找到所有三角形在此方向的中位数（利用 `std::nth_element`）
- 计算如果按照此数进行划分有多少三角形会落在中间（即三角形的包围盒包含该值）

在三个坐标方向中选取最少重叠的方向作为划分，将小于或大于该值的三角形分到两边，而包含了该值的三角形则两边各复制一份。对于这两边分别重新计算总包围盒，并递归分块。

分块直到 1. 块内三角形数量小于一定值 2. 无论哪个方向都无法有效分块

对于 2,339,160 个三角面的草地模型，产生了 25 层共 359,951 个叶子节点，平均包含 19.6 个三角面，有效地加速了求交过程。

网格求交效果



Apollo.png
290,529 三角面
2048x2048 分辨率
6625 采样次数
199 CPU小时

Louis.png
421,965 三角面
2048x2048 分辨率
4297 采样次数
190 CPU小时



Man.png
1,921,486 三角面
9024x3840 分辨率
3606 采样次数
296 CPU小时

曲面求交

曲面求交只需输入曲面的参数方程以及其偏导数即可绘制

记曲面方程为 $\mathbf{f}(u, v)$ ，有定义域 $u \in [u_{min}, u_{max}]$, $v \in [v_{min}, v_{max}]$

初始交点

通过采样为曲面构造三角面网格进行求交：

- $\mathbf{f}(u_k, v_k), \mathbf{f}(u_k, v_{k+1}), \mathbf{f}(u_{k+1}, v_k)$
- $\mathbf{f}(u_{k+1}, v_k), \mathbf{f}(u_k, v_{k+1}), \mathbf{f}(u_{k+1}, v_{k+1})$

其中 $u_k = u_{min} + \frac{k}{n}(u_{max} - u_{min})$, $v_k = v_{min} + \frac{v}{n}(v_{max} - v_{min})$, n 为采样数量，通常取 100

阻尼牛顿法迭代求精确解

对于方程 $\mathbf{r} = \mathbf{f}(u, v) - (\mathbf{s} + t \cdot \mathbf{d}) = \mathbf{0}$ ，某一步的求解情况为

$$\mathbf{f}(u_k, v_k) - (\mathbf{s} + t_k \cdot \mathbf{d}) = \mathbf{r}_k$$

进行下一步迭代，需要解线性方程组 $\frac{\partial \mathbf{f}}{\partial u} \Delta u + \frac{\partial \mathbf{f}}{\partial v} \Delta v - \mathbf{d} \Delta t = -\mathbf{r}_k$,

得 $\Delta u, \Delta v, \Delta t$

再令权重系数 $\lambda = 1$ ，带入
$$\begin{cases} t_{k+1} = t_k + \lambda \Delta t \\ u_{k+1} = u_k + \lambda \Delta u \\ v_{k+1} = v_k + \lambda \Delta v \end{cases}$$

如果得到的 $\|\mathbf{r}_{k+1}\| \geq \|\mathbf{r}_k\|$ ，则令 λ 减半重新计算参数，直到满足 $\|\mathbf{r}_{k+1}\| < \|\mathbf{r}_k\|$ 为止（阻尼牛顿法思想）。如果无法满足，则求解失败。

重复迭代直到 $\|\mathbf{r}_k\|$ 小于某足够小的数

曲面求交效果



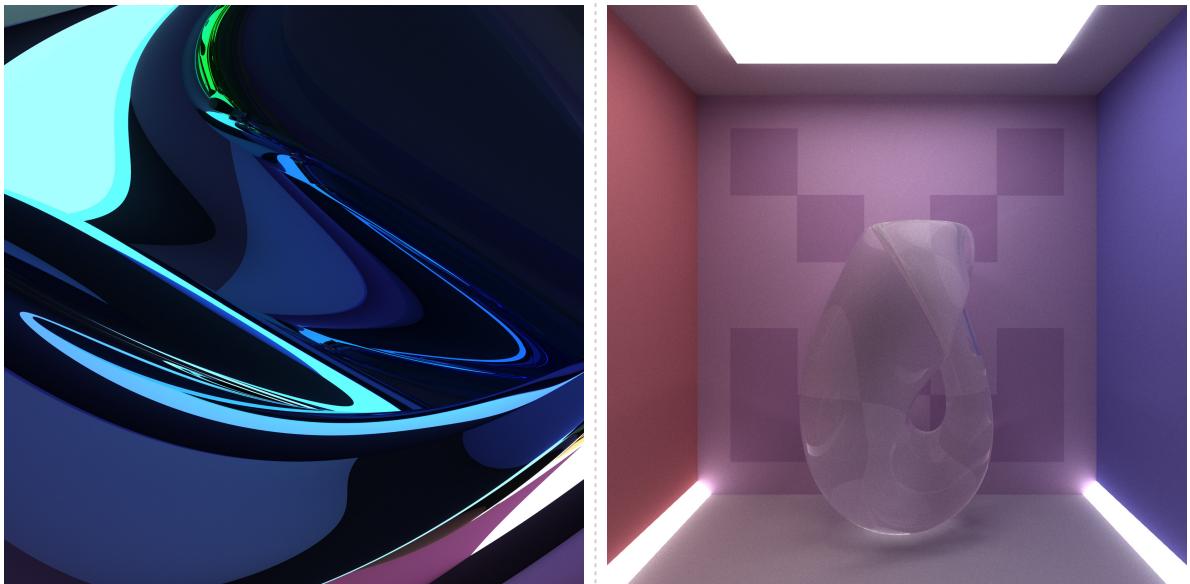
Ring Glaze.png
2048x2048 分辨率
4330 采样次数
332 CPU小时

Ring Refl.png
2048x2048 分辨率
1537 采样次数
156 CPU小时

方程： $\mathbf{f}(u, v) = \begin{bmatrix} \cos v \cdot [6 + (\frac{5}{4} + \sin 3u) \cdot \sin(u + 3v)] \\ \sin v \cdot [6 + (\frac{5}{4} + \sin 3u) \cdot \sin(u + 3v)] \\ \cos(u + 3v) \cdot (\frac{5}{4} + \sin 3u) \end{bmatrix}$

其中 $u \in [0, 2\pi]$, $v \in [0, 2\pi]$

颜色也是关于 u, v 的方程



Ring Zoom.png
4096x4096 分辨率
409 采样次数
131 CPU小时

Klein.png
2048x2048 分辨率
2270 采样次数
32 CPU小时

左图是前页右图的局部放大

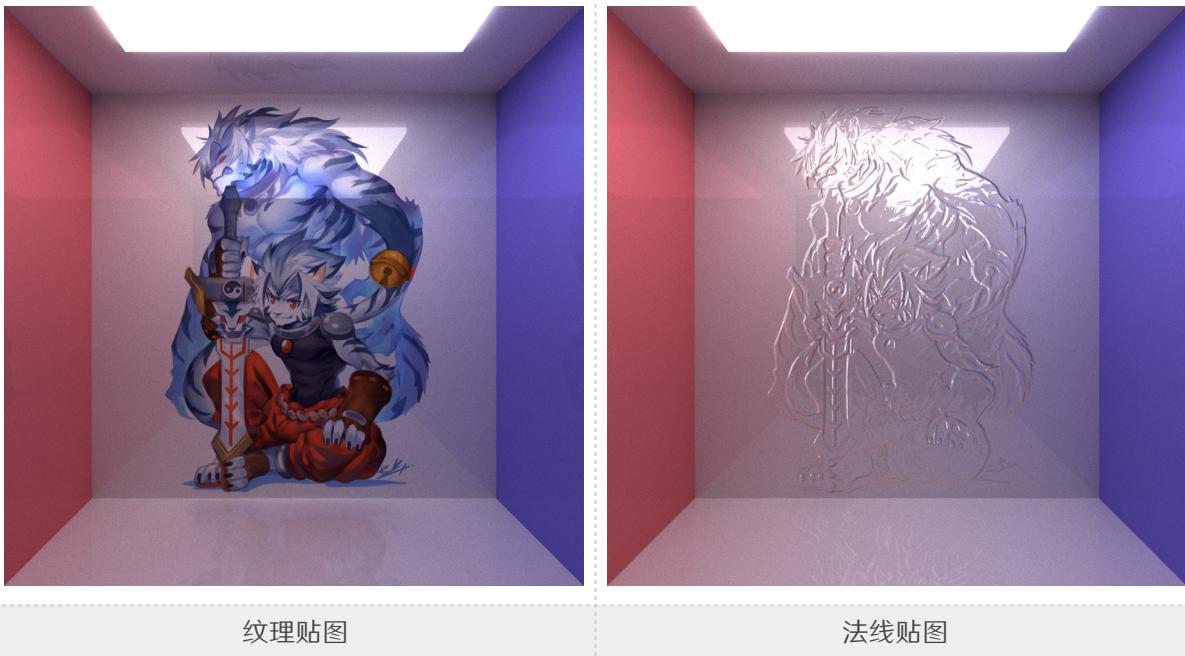
右图由两部分方程构成

$$\mathbf{f}_1(u, v) = \begin{bmatrix} 0.25668 - 0.295 \cdot (-2 + \cos u) \cdot \cos u \cdot \cos v + 0.5 \cos u \cdot (1 + \sin u) \\ 2.16979 + (-1.56 + (-0.59 + 0.295 \cos u) \cdot \cos v) \cdot \sin u \\ -0.295 \cdot (-2 + \cos u) \cdot \sin v \end{bmatrix}$$

$$\mathbf{f}_2(u, v) = \begin{bmatrix} 0.25668 + 0.59 \cos v + \cos u \cdot (-0.5 + 0.295 \cos v + 0.5 \sin u) \\ 2.16979 + 1.56 \sin u \\ -0.295 \cdot (2 + \cos u) \cdot \sin v \end{bmatrix}$$

其中 $u \in [0, \pi]$, $v \in [0, 2\pi]$

贴图



纹理贴图

法线贴图

贴图可以增加场景的真实感

原理

对于每个三角面，其三个顶点分别在贴图文件上存在一个锚点，从而根据交点位置获取交点颜色或法向

法线贴图恰和纹理贴图一样存在三个分量，但是额外需要模型确定其平行于三角面的两个分量的方向

法线插值

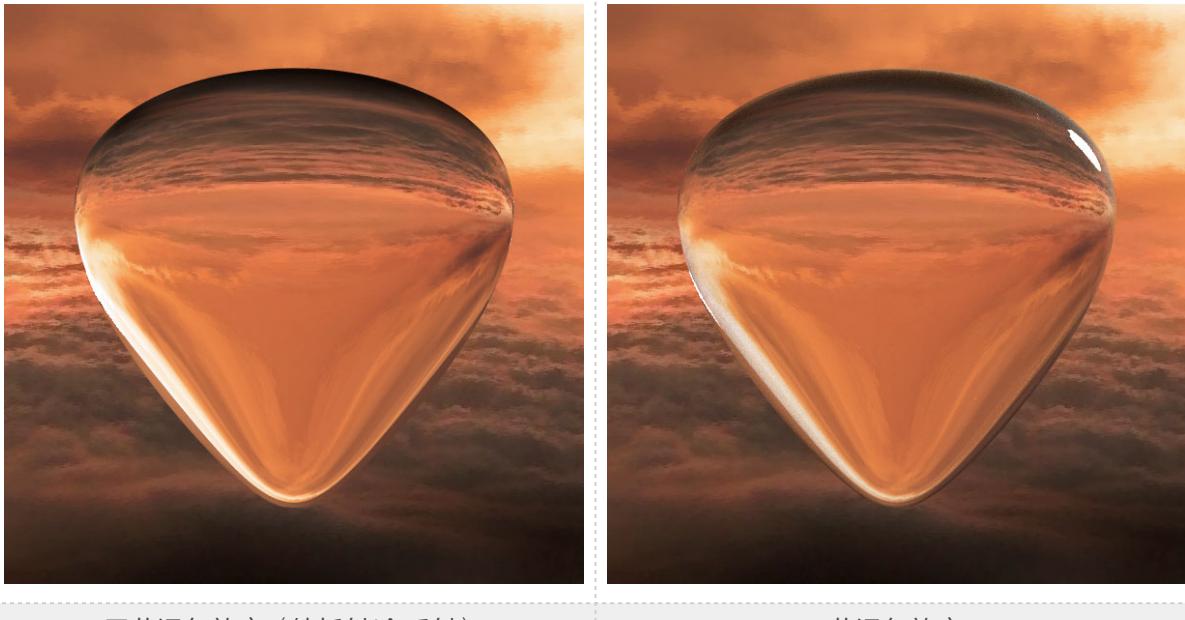


法线插值可以让网格的表面更加平滑

原理

对于网格的每个顶点，根据通过这个顶点的所有三角面计算出平均法向作为这个顶点的法向。然后每个三角面的重心位置对应三角面的法向，三个顶点分别对应三个顶点的法向。求得光线交点后，将线性插值的结果作为交点法向

菲涅尔效应



无菲涅尔效应（纯折射/全反射）

菲涅尔效应

真实的折射物体，在入射角较大时，其反射比例会增加，符合以下公式：

$$R_s = \left| \frac{n_1 \cos \theta_i - n_2 \cos \theta_t}{n_1 \cos \theta_i + n_2 \cos \theta_t} \right|^2$$

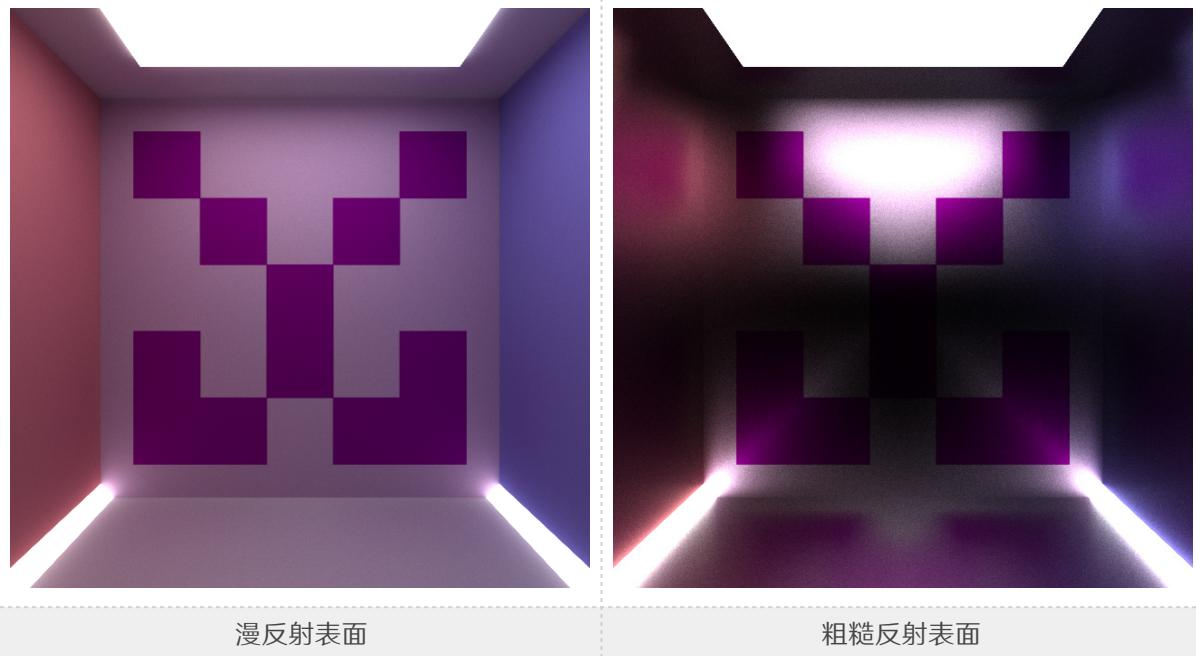
$$R_p = \left| \frac{n_1 \cos \theta_t - n_2 \cos \theta_i}{n_1 \cos \theta_t + n_2 \cos \theta_i} \right|^2$$

$$R_{eff} = \frac{1}{2}(R_s + R_p)$$

原理

尽管随机转盘结果可能是“折射”，但是根据菲涅尔效应可以计算出反射部分比例。即便没有到达全反射，仍有一定概率从“折射”转入“反射”

粗糙表面



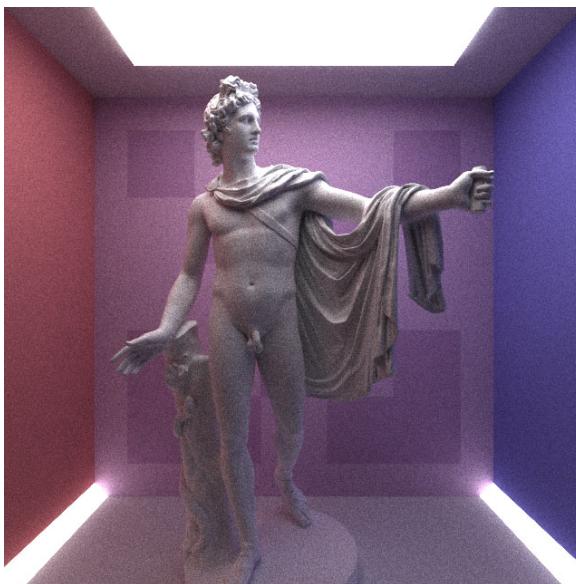
许多物体的反射效果介于镜面反射与漫反射之间，因此可以通过混合两种反射来模拟这样的效果

原理

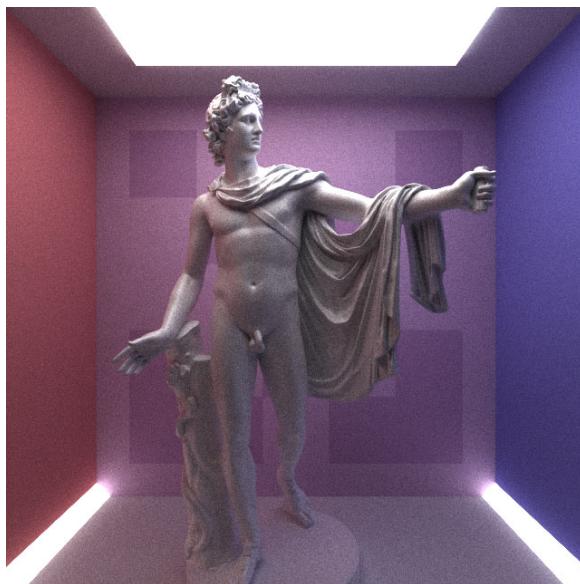
在计算光线方向时，将镜面反射方向与一定比例的随机方向进行混合，就形成了粗糙表面的镜面反射效果。上右图就在正常反射方向基础上加入了长度为 $1/4$ 单位向量的随机扰动。

效果

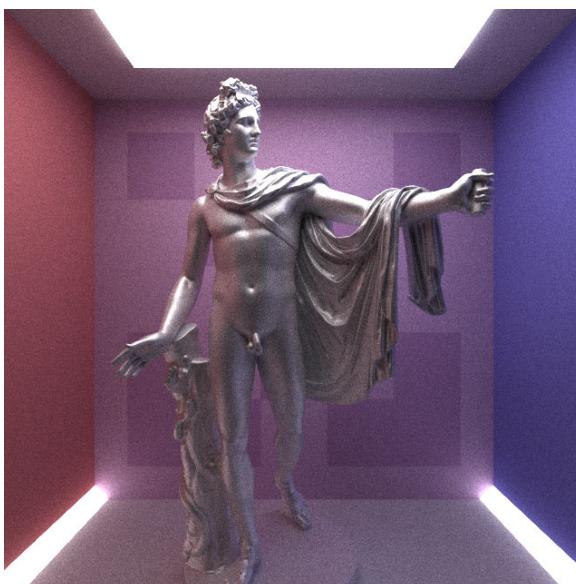
当混合了漫反射与粗糙反射后，可以形成更加真实和多样的物体表面效果



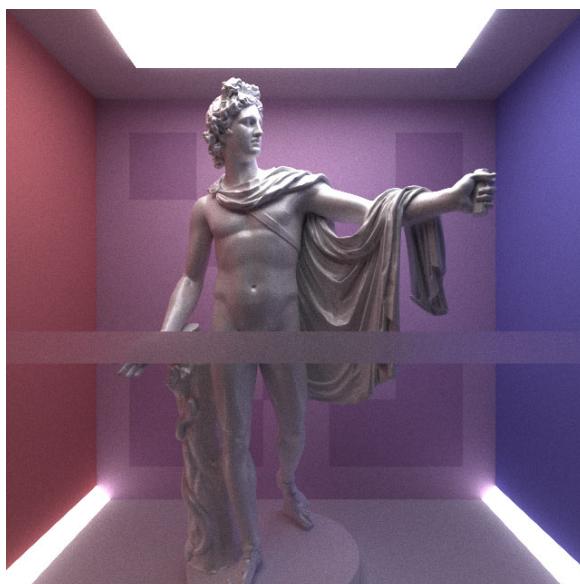
漫反射 (石膏)



混合 10% 粗糙反射



混合 50% 粗糙反射 (金属)



粗糙折射 (毛玻璃)

景深



景深可以模拟摄像机的对焦效果，增加真实感

原理

对摄像机的位置加一随机扰动，同时调整射线方向，使得原本射向焦距面上某一点的射线仍然射向该点，但射线在其他距离的位置便会发生变化

因为每个像素点发出的射线不再确定，带景深渲染的噪点会十分明显，需要显著更长的渲染时间来消除噪声。

景深效果



Grass Bubble.png
1200x1200 分辨率
3761 采样次数
242 CPU小时

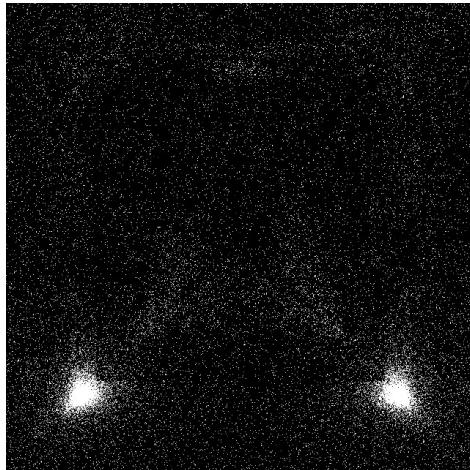
Grass Diamond.png
2048x2048 分辨率
5656 采样次数
~900 CPU小时



焦距 f=1

焦距 f=1.8

Photon Mapping



Path Tracing 几乎不可能渲染一个光源非常小但亮度非常高的场景，也很难渲染出清晰的光线聚焦效果。而使用 Photon Mapping 来追踪光子的运动，便可以达成这样的效果

原理

从光源处发出大量光子。其反射过程与 PT 类似，而每当光子进入漫反射面，就保存该光子的信息。产生了足够多的光子之后，再进入摄像步骤，由摄像机发出射线，求出射线与物体交点位置的光子情况

为了加速，我采用漫反射表面贴图来记录光子——每个光子打在漫反射面上后就存在这个面的一个贴图中。这样，只要总光子亮度不超过浮点数上界，就可以几乎无限地发射光子来模拟

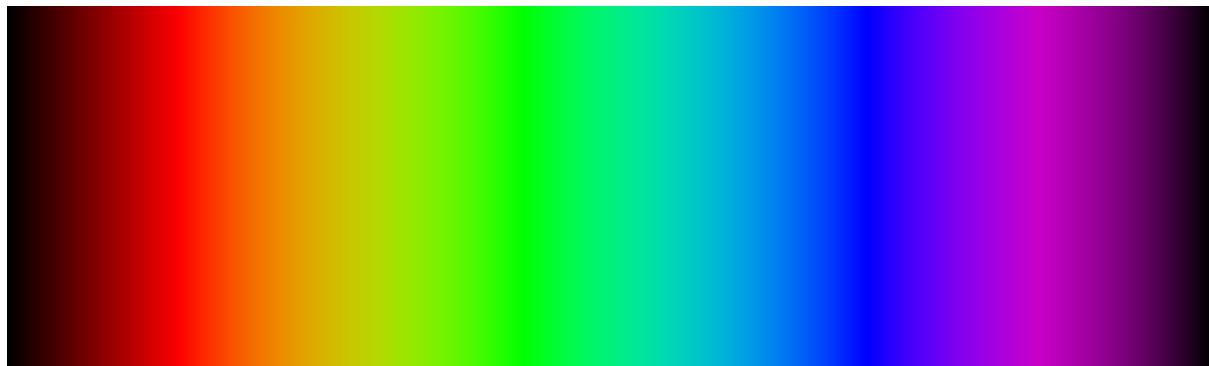
这样做的另一个优点是，产生了足够多光子信息后，摄像步骤实际上是很慢的。因此，对于一个场景可以快速地生成不同位置和角度的影响，以生成一个动画

色散

在 PM 的基础上，如果定义每个发射的光子的颜色，然后根据不同的颜色确定不同的折射率，就可以达成色散效果

光子的颜色生成

我使用了分段函数来模拟光谱，其效果如下：



其中颜色过渡较为平滑，而且对整段光谱进行积分的和恰为白光
光源发出的白光即使在频谱中均匀选择颜色

物体对颜色的吸收

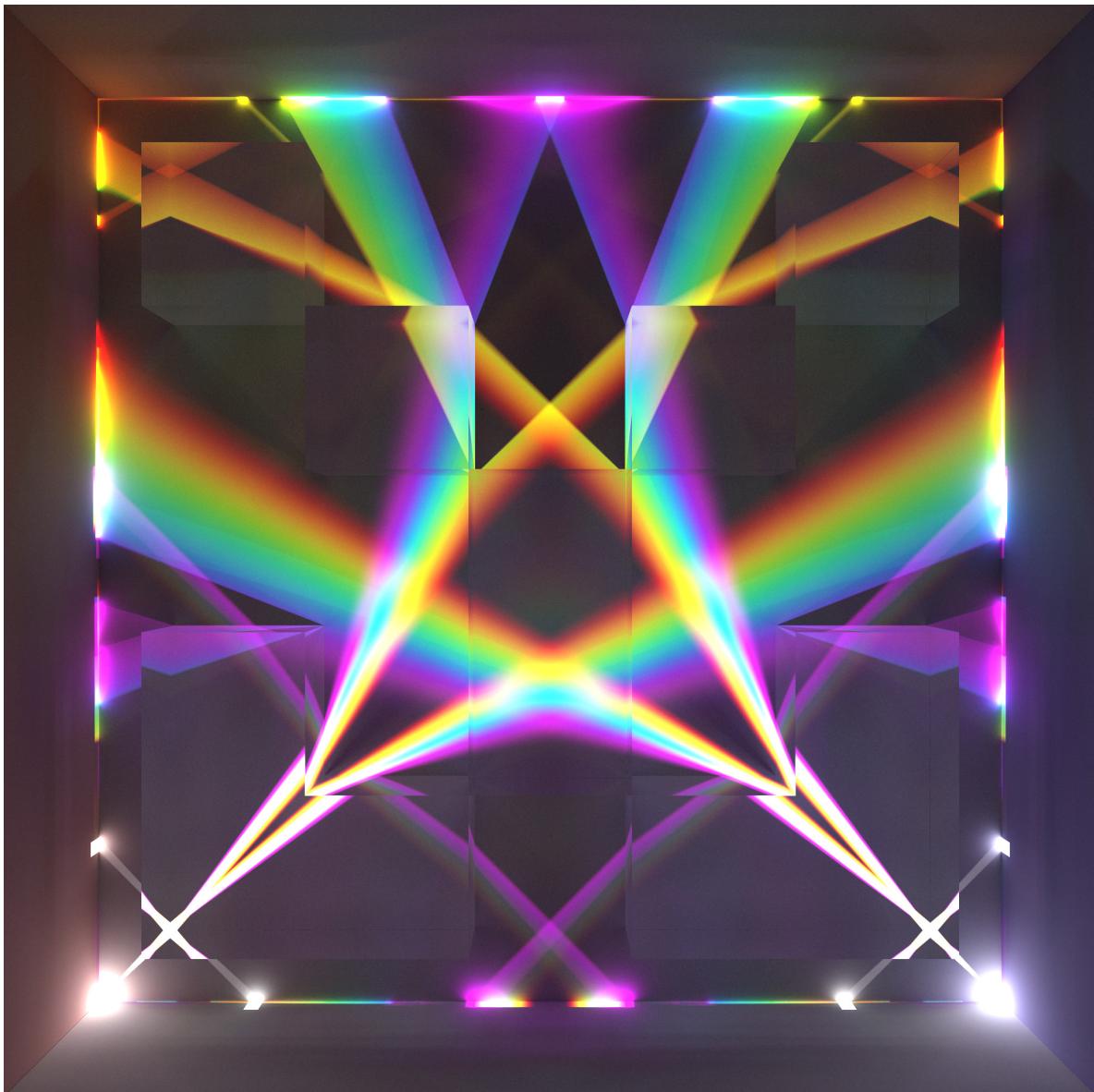
此时物体吸收光子的颜色不再是颜色分量相乘，因为物体只能吸收光子能量而不能改变光子频率。因此我将物体颜色分为两部分：

$$\text{rgb}(a, b, c) = \text{rgb}(w, w, w) + \text{rgb}(x, y, z), \text{ 其中 } \min(x, y, z) = 0$$

这样， (x, y, z) 相当于物体的颜色，而 (w, w, w) 部分则是灰色，直接乘以 w 反射光子

物体与光子的颜色差别越大，吸收越多。因此我将颜色先映射到与 $(1, 1, 1)$ 相垂直的平面上，然后单位化做点乘作为反射率

色散效果



Spectrum.png
2048x2048 分辨率
10,000,000,000 光子
~200 CPU小时

左下和右下为光源

为了实现更加绚丽的效果，此渲染中夸大了色散的程度，真实物体中折射率没有如此大差异

另有此场景中移动摄像头的动画，为 Spectrum1.mp4 和
Spectrum2.mp4