**Introduction to Physically Based Rendering(PBR) and Physically Based Ray Tracing(PBRT)**

练孙鸿

Notes of PBR 3rd Edition – chapter1

**Introduction**

渲染，是从3D场景生成图片的过程。这样的描述显然是过于宽泛的，因为还有很多种方法去完成这个任务。而基于物理(Physically Based)的方法会基于物理定律对光与物质之间的交互进行建模来模拟现实世界。虽然基于物理的方法看起来是一种非常直观的方法，但由于机器性能等原因，直到最近十来年才逐渐被广泛使用。1.7会介绍Physically Based Rendering(PBR)的简要发展历史。

这本书介绍了pbrt，也就是基于光线跟踪(RT, ray tracing)实现的基于物理的渲染系统(physically based rendering system)。

* 1. **Literate Programming**

(略)

* 1. **真实感渲染与光线跟踪算法**

真实感渲染(photo-realistic rendering)的目标是创造出一张让人难以分辨真假的3D场景的照片级图像（虽然书中也提到了“难以分辨真假”(indistinguishable)是一个不太准确的词，毕竟这个词包含着人类观察者的主观评价）。所以我们只会尽我们所能地去模拟光、光和物理的交互的物理过程，并努力给人类观察者献上品质最好的渲染图片。

几乎所有的PBR系统都是基于光线跟踪算法的。光线跟踪从原理和流程来说是比较简单易懂的：**它跟踪着光线在场景中的传播路径，并考察它是如何和周围环境交互然后反弹离开**。但无论要怎么写一个基于ray tracing的渲染系统，都起码需要模拟如下的对象或者现象：

* **摄像机、成像器(Camera)**：首先我们需要对摄像机进行建模，这决定了场景的光线会怎么被“观察者”通过成像器记录下来。非常多的渲染系统都从摄像机开始发射光线，然后跟踪到场景里面（如Path Tracer）。(pbrt 1.2.1)一般情况下我们最常用的就是针孔摄像机(pinhole camera)模型，场景物体一般会透视投影(perspective projection)到成像平面上。
* **射线与物体的相交(Ray-Object intersection)**：给定射线/光线，我们必须要知道他们在哪与物体相交，并且我们还需要交点处的一些几何信息如surface normal、材质等。大部分的ray tracing渲染系统可以测试一根射线跟多个物体的相交，一般情况下返回离射线起点最近的交点。如果我们把条射线和场景物体逐对求交的话，复杂度会是二次的，但如果我们用的代价建立加速结构之后，以后的Ray-Object intersection的复杂度可以优化成。
* **光源(Light Sources)**：光源是渲染里面非常重要的一个组成元素，如果没有了光源，渲染一个场景就变得毫无意义了。渲染系统必须对光在场景中的分布、光在空间中的能量分布进行建模。(pbrt 1.2.3)渲染系统一般需要场景中光照的几何(geometric)与辐射度学(radiometric)的分布，然后才能考察与计算到达某点的光线总能量。在渲染的时候，渲染系统会经常想要知道Ray-Object intersection交点处微元面积(differential area)接收并储存的能量。
* **可见性(Visibility)**：为了确定光在空间某一点上储存的能量是多少，我们必须要知道点到光源处是否有不被遮挡(un-interrupted)的路径。(pbrt 1.2.4)只有在光源和有不被遮挡(un-obstructed)的路径，才能被着色。只要光源还有一部分抵达点，那么它的影响就还需要考虑进去。
* **表面散射(Surface Scattering)**：每个物体都需要有一个表面的几何描述，无论是用解析形式还是网格描述；然后我们还需要知道光线与物体表面交互的方式（如反射、折射、散射），以及被再次辐射(re-radiated)(或者说被散射的)的光线的属性。其中[1]散射(scatter)广义上的含义是：某种形式的辐射（如光、声、运动粒子）因为空间中一些局部的突变而被迫偏离原来直线传播的轨道，从而向其他一个或多个方向继续传播。而日常生活当中说的“光线散射”现象如[2]丁达尔效应（Tynall Effect）则是散射的一种例子。(pbrt 1.2.5)渲染系统需要知道Ray-Object intersection交点处的材质(material)，一般用双向反射分布函数BRDF(Bi-directional Reflectance Distrution Function)来描述。BRDF是渲染方程的积分里面的一项，用来描述入射到物体在某点处**、**入射方向为的光线经过反射/散射在出射方向上的能量，一般记为。那么计算反射回摄像机的光量伪代码就是：

|  |
| --- |
| For each light:  If light is not blocked:  Incident\_light = light.***L***()  Amount\_reflected = surface.BRDF(***, ,* )**  ***L***+= Amount\_reflected \* Incident\_light |

* **间接光传输(indirect light transport)**：从物体表面反弹出射的光线也可能会落在其它物体的表面上，这些光线也要考虑到。(pbrt 1.2.6) [Turner80]指出用ray tracing求解global illumination问题的递归(recursive)特性。而我们随时都可以向下递归并加上indirect light的影响。（然后书中稍微提了一下light transport equation/the rendering equation）
* **光线/射线传播(Ray Propagation)**：如果光线穿过的是真空，那么它在直线传播路径上的能量不会有变化。虽然真空在地球上并不常见，但是作为效率和效果的平衡，一般渲染就当光线是在真空中传播。在一些更加复杂精巧的渲染模型里面，光线可以穿过雾、烟、云、地球大气层等非真空介质（可能会产生散射与能量损失）。这就是体渲染(Volume Rendering)的由来，体渲染需要用数值方法求解体光照传输方程(volumn light transport equation)，这个会在pbrt chap11和chap15继续讲。
  1. **pbrt系统概览**

pbrt 3rd edition 在讲解ray tracing render system的时候会放上很多C style pseudo-code，这本书似乎也有配套的pbrt的开源系统。它的设计

引用

[1] <https://en.wikipedia.org/wiki/Scattering>

[2] <https://en.wikipedia.org/wiki/Tyndall_effect>

[3] Turner Whitted. An Improved Illumination Model for Shaded Display[C] Communication of the ACM, Volume 23 Issue 6, June 1980 Pages 343-349