计算机图形学大作业

实验报告

王　昭　软件32班　2013013330
陈彤宇　软件32班　2013013325
周立旺　软件32班　2013013326

目录

1 实验环境

- CPU：Intel i5-6500 @ 3.20GHz * 4
- Memory：16GB
- OS：Windows 10 教育版
- Visual Studio：2013

2 实验目标

- 在第二次小作业《光线追踪》的基础上进行修改及扩展
- 实现蒙特卡洛路径追踪算法，并实现漫反射、Phong两种BRDF模型。
- 实现了一篇关于通过神经网络来实现对固定场景全局光照进行渲染的文章[Ren et al., 2013]，并在其基础之上额外支持了一些新特性。

3 实现内容

3.1 光线追踪

这一部分在第二次小作业中已经基本实现，在本次实验中，对之前的光线追踪渲染器进行进一步修改及扩展。

3.1.1 多线程下随机数生成

在之前的作业中使用了C的srand方法种下种子，并调用rand方法来获取伪随机数。在多线程加速的情况下，会导致多个线程产生相同随机数的情况，但这并不是我们想要的结果。

在本次实验中，实现了RNGenerator类，以支持对多线程随机数的生成。RNGenerator类封装了C++11的随机数生成引擎，并且使用梅森旋转算法[1] 来进行随机数的生成。当然，最主要的目标还是要保证不同线程的种子不同。在对RNGenerator类进行实例化时，将会使用当时的时间及当前线程ID做异或操作，用上述结果作为随机数种子，从而保证了多线程随机数生成的随机性。

[1] https://en.wikipedia.org/wiki/Mersenne_Twister

3.1.2 额外渲染对象的支持

实现了对四边形的支持，使用了[Lagae and Dutré, 2005]所述的光线与四边形快速求交的算法。以此为基础，即可实现对康奈尔盒模型的渲染。

3.2 蒙特卡洛路径追踪

路径追踪[2]是一种对真实世界下全局光照的蒙特卡洛渲染方法，笼统来讲，该算法利用蒙特卡洛方法对渲染方程进行近似，从而达到对真实情况的逼近。在本次实验中，路径追踪渲染器用来作为直接/非直接光照的数据生成器及用来对比的Ground-Truth 数据生成器。

3.2.1 渲染方程

渲染方程[3] 于1986年被引入计算机图形学，在不考虑波长λ及时间t的情况下，其形式如下：

$$L_o(\boldsymbol{x},\omega_o) = L_e(\boldsymbol{x},\omega_0) + \int_\Omega f_r(\boldsymbol{x},\omega_i,\omega_o) L_i(\boldsymbol{x},\omega_i)(\omega_i \cdot \boldsymbol{n}) d\omega_i$$

其中

- $L_o(\boldsymbol{x},\omega_o)$ 代表在位置$\boldsymbol{x}$以及角度ω_o的出射光
- $L_e(\boldsymbol{x},\omega_o)$ 代表上述位置及方向发出的光
- $\int_\Omega ... d\omega_i$ 代表入射方向半球的无穷小累加和
- $f_r(\boldsymbol{x},\omega_i,\omega_o)$ 是BRDF，也即在该点从入射方向到出射方向光的反射比例
- $L_i(\boldsymbol{x},\omega_i)$ 代表该点的入射光位置及方向ω_i

3.2.2 重要性采样

在蒙特卡洛方法模拟积分时，采用重要性采样方法会使蒙特卡洛方法得到结果的方差减小，从而使算法更好更快地收敛至正确结果。

在路径追踪的过程中，当追踪光线与某物体存在一个交点时，需要随机确定一个反射方向从而继续追踪，此时就需要使用重要性采样来对不同的BRDF进行方向的选择。方向向量ω可以由天顶角θ及方位角ϕ确定，接下来介绍两种BRDF的重要性采样方法。

[2] https://en.wikipedia.org/wiki/Path_tracing
[3] https://en.wikipedia.org/wiki/Rendering_equation

如果该物体的BRDF是Lambertian，也即$f_r(\boldsymbol{x},\omega_i,\omega_o)=\frac{\rho}{\pi}$。假设$u,v$是两个属于[0,1]且均匀分布的随机数，那么

$$(\theta,\phi)=(arccos(\sqrt{1-u}),2\pi v)$$

概率分布函数为

$$pdf(\omega_i)=\frac{cos(\theta)}{\pi}$$

如果该物体的BRDF是Phong，也即$f_r(\boldsymbol{x},\omega_i,\omega_o)=\frac{k_d}{\pi}+k_s\frac{(n+2)}{2\pi}cos^n(\alpha)$。由于表达式的前半部分与Lambertian一致，我们只讨论后半部分的情况。u,v的定义同上，则

$$(\theta,\phi)=(arccos(u^{\frac{1}{n+1}}),2\pi v)$$

概率分布函数为

$$pdf(\omega_i)=\frac{n+1}{2\pi}cos^n(\alpha)$$

其中α为ω_o与$\boldsymbol{n}$所成的夹角。

3.2.3 直接光照& 非直接光照

由于在之后的实验中，会对场景中每个物体的直接光照及非直接光照进行训练，所以需要将当前的渲染方法拆成直接光照值和非直接光照值。

当追踪光线与某物体存在一个交点时，每次只需递归跟踪一条路径，现在追踪两条：一条指向光源ω_{direct}，另一条与之前相同，仍然通过重要性采样随机进行跟踪$\omega_{indirect}$。其中，ω_{direct}路径无需继续递归，只计算光源的发射值，$\omega_{indirect}$路径则继续递归跟踪下去。计算直接光照的伪代码见算法 1：这里提到了对光源的采样以及概率

Algorithm 1 directRadianceEst(x,ω_r)

ω_i, pdf =luminaireSample$(x,\boldsymbol{n})$

y =traceRay(x,ω_i)

return brdf$(x,\omega_i,\omega_r)\cdot$emittedRadiance$(y,-\omega_i)/pdf$

密度分布函数问题，这个问题将在下一小节说明。计算非直接光照的伪代码见算法 2：

References

Lagae, Ares and Philip Dutré (2005). “An efficient ray-quadrilateral intersection test”. In: *journal of graphics, gpu, and game tools* 10.4, pp. 23–32.

Ren, Peiran et al. (2013). “Global illumination with radiance regression functions”. In: *ACM Transactions on Graphics (TOG)* 32.4, p. 130.

Algorithm 2 indirectRadianceEst(x, ω_r)

if random()<survivalProbability **then**
  ω_i, pdf =brdfSample($x, \boldsymbol{n}$)
  y =traceRay(x, ω_i)
  return brdf(x, ω_i, ω_r)·reflectedRadianceEst($y, -\omega_i$)/(pdf·survivalProbability)
else
  return 0
end if