蒙特卡洛光线追踪

1 编程环境

VS2012+OpenGL（使用了 glut 和 FreeImage 两个第三库）

项目托管地址：https://github.com/feiqian/PathTracing

2 运行

直接运行程序即可。程序启动时会自动扫描程序根目录下的 data 目录（**如果想测试其他的场景文件，请将其拷贝至该目录下**），找到所有子文件夹后会提示用户输入要显示的场景（**相机和光源信息由.scene 文件定义。模型和材质信息由.obj 文件定义**）。输入正确编号后，程序就会渲染出结果，并同时显示当前迭代次数（spp 默认为 100）,全部迭代结束后，程序会将结果写入到 result 目录。另外程序暂不支持鼠标和键盘操作。

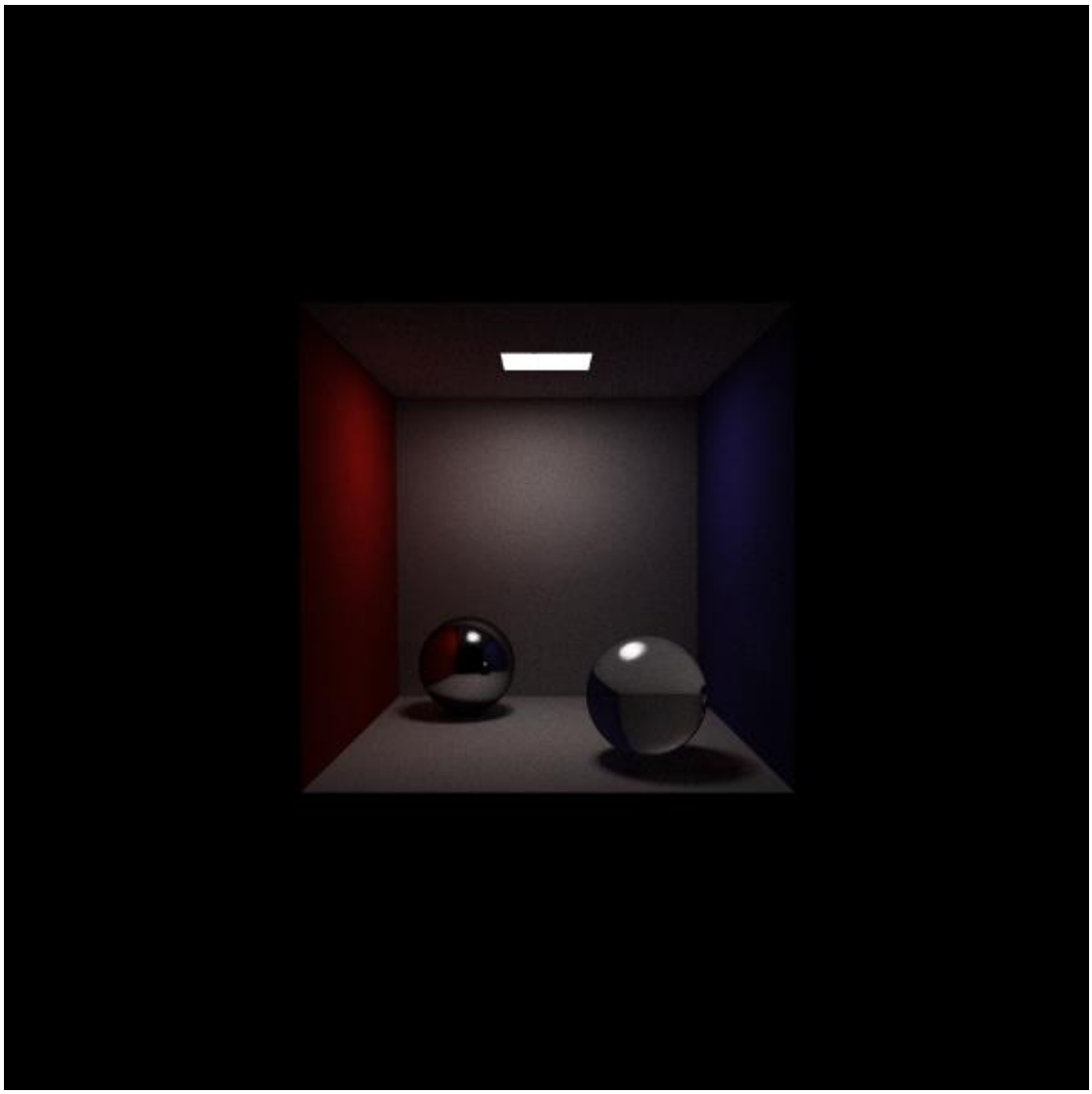
3 特性

1. 反走样：对每个像素点进行子采样（默认值为 1）。
2. 软阴影：在面光源上随机撒点计算阴影（默认采样值为 3）。
3. 图元：实现了立方体，球，平面和三角网格四种基本图元。
4. 光源：实现了环境光源，点光源和区域光源。
5. 纹理贴图：遵循 obj 格式
6. 折射：支持 transmissive materials。使用菲涅尔系数计算反射和折射比率，并根据俄罗斯赌盘的结果决定选择反射还是折射光线进行递归。
7. 重要度采样：使用重要度采样而不是均匀采样半球内的光线，并使用直接光照加速收敛过程。
8. 加速结构：使用 AABB 层次包围盒（BVH）对场景求交进行加速。另外使用 OpenMP 进行并行加速。

4 测试结果

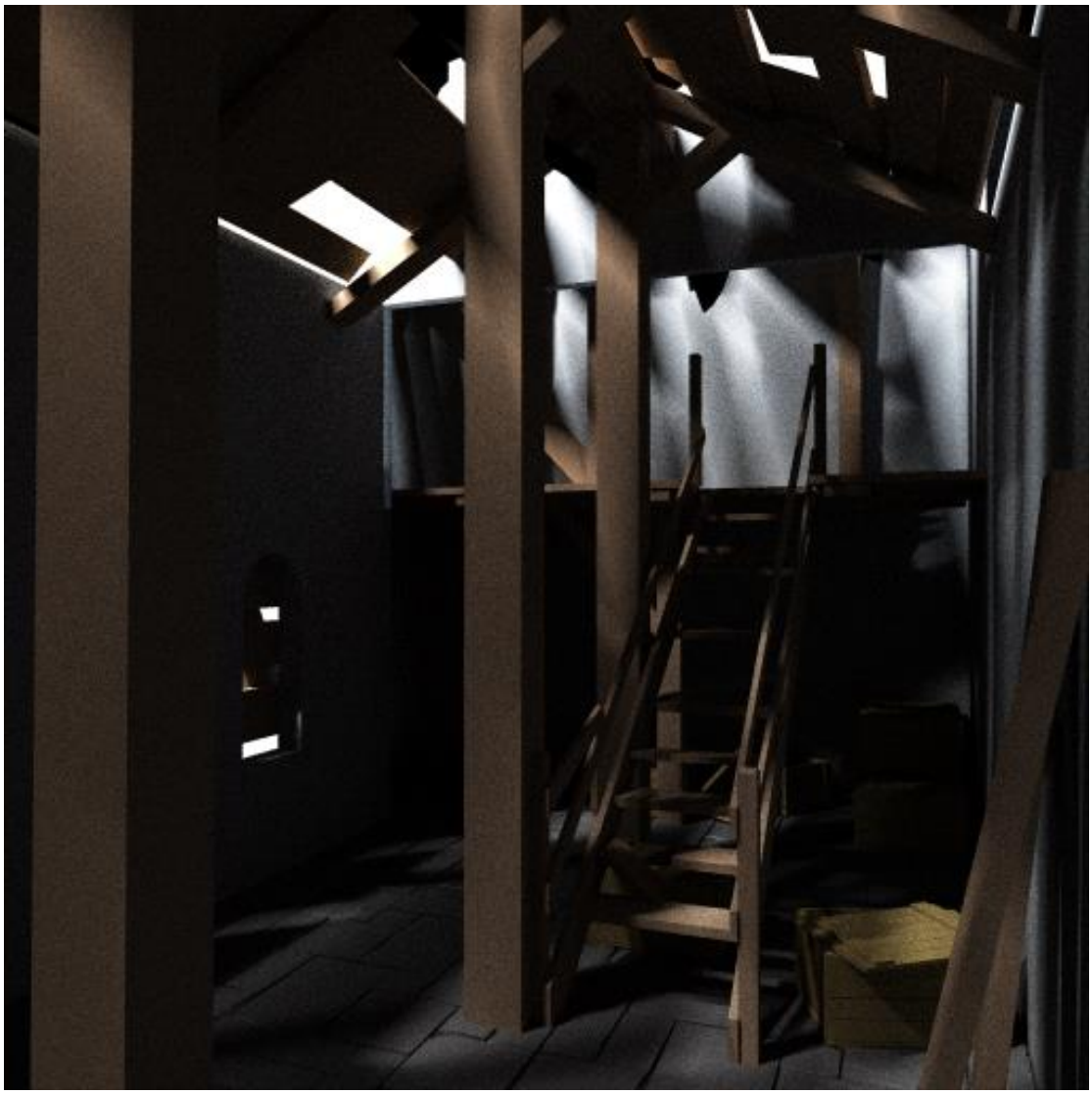
测试环境：win10 64bit 笔记本 + i5-3230M CPU(2.60GHZ) + 4G 内存

场景 1：cornell box



效果：SPP 为默认值 100。每帧渲染时间大约为 1s。左边球（mirror）和右边球（glass）的效果基本上都得到了较好的渲染。不过有一点不知道是什么原因（可能是场景文件？），右边球折射后的焦散效果与标准程序相比不是很好。

场景 2：ruins



效果：SPP 为默认值 100。每帧渲染时间大约为 10s。这个场景基本上都是 diffuse 材质的物体，总体效果还比较 OK。

5 TODO

1. 景深
2. 其他优化（CUDA 加速）和改进

6 总结和感想

总的来说，通过这次蒙特卡洛光线追踪算法的实现，自己还是收获颇多。一方面弥补了当时课堂上对蒙特卡洛算法的的一知半解，加深了我对这个真实感渲染算法的认识（不过还有一点东西的原理自己还没有弄懂，希望后面能把它搞懂）。另一方面同时除了两个第三库库以外，整个程序都是自己一点一点搭建起来的。在各种修 bug，调效果的过程中锻炼了自己的编码能力。