**蒙特卡洛路径追踪算法的实现**

**编程环境**：

c++， visual studio 2017，win10 x86，cpu = Intel core 12700K，gpu = RTX 3080ti

**依赖库：**

c++标准库、GLFW/glut（仅使用窗口创建和调用、主渲染循环等基础函数）、glm（仅使用向量和矩阵运算等函数）、FreeImage（仅使用读取和保存图片等函数）

**运行方法：**

1. 在./release/文件夹中选择要绘制的对象（cornell-box: mc\_cb.exe, bedroom: mc\_bd.exe, veach-mis: mc\_vm.exe, my cornell-box: mc\_my.exe）。
2. 运行exe，命令行会显示实时的绘制进度。
3. 每一轮绘制结束后可以在./result/文件夹中查看。若不主动操作，绘制会一直进行下去，但是每一轮绘制结束后会在./result/中更新结果。

**目录结构：**

**实验结果：**

1. Cornell-box

spp = 50的绘制结果如下

与老师给定的sample较为相似，物体的固有色、漫反射和材质贴图等信息基本正确。

1. Bedroom

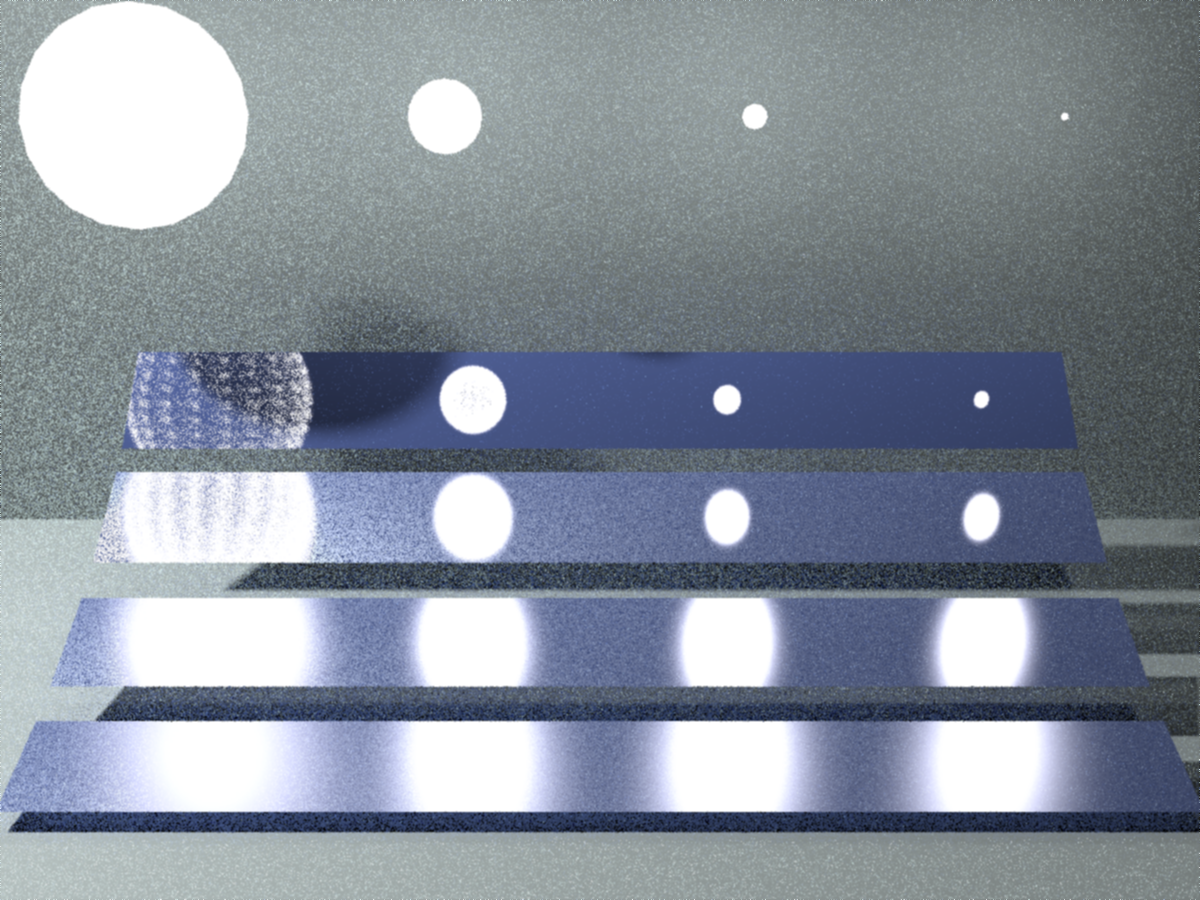
spp = 50的绘制结果如下



整体视觉效果尚可，但玻璃材质的透射光表现有些欠缺。

1. veach-mis

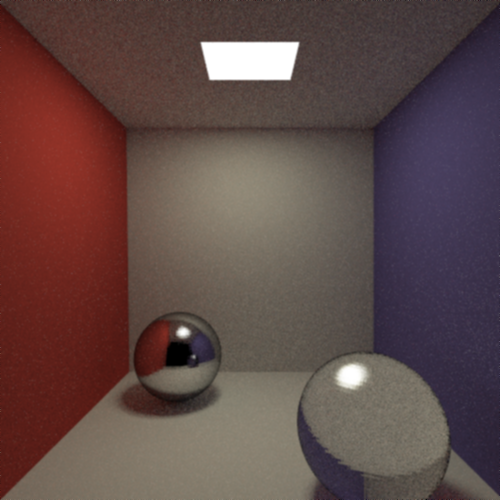
spp = 10的绘制结果如下

****

本图光源数量多，绘制速度很慢。图像整体出现了一定的过曝现象，反射光的采样数量不够，出现了斑驳的现象。

1. my cornell-box

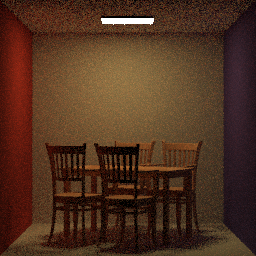
spp = 50的绘制结果如下



更明显的展示了玻璃材质，尝试了视野更大的透视效果，所以物体有些畸变。

**算法优化：**

1. 使用八叉树结构（Octree）和轴对齐包围盒（AABB）来加速光线和面片的求交运算，加速效果已经在秋学期的zBuffer作业中得到证明。
2. 使用俄罗斯轮盘赌（RR）来决定是否继续递归，而没有通过设置最大递归深度（实际上，递归深度设置了一个非常大的值，在该深度下几乎没有光线能够存活），这样可以使得结果接近无偏。通过控制每次存活的概率（survival）来均衡质量和效率的问题。以下分别为survival = 1 至survival = 0.3的绘制结果。



Survival = 0.8

Survival = 1.0

Survival = 0.3

Survival = 0.5

可以看出survival越低，物体表面越难展现附近其他物体的颜色，整体颜色更暗淡一些，但是绘制速度有很大的提升。所以权衡效率和质量，最终选定了survival = 0.8。

1. 对最后绘制的结果图像和纹理采样使用了高斯滤波和二元线性插值，得到的图样更加平滑。
2. 在release模式下开启了openMP支持，利用了多线程的并行性加速计算。

**不足和改进：**

1. 开启直接光照时，对多光源场景的绘制变得非常慢，因为对同一个光源的多个面（比如球形光源由很多面组成）分别进行了采样，优化采样策略，减少采样次数就能对绘制效率有很大提升。
2. 透射光和镜面反射光采样时似乎还有很多地方没采到（bedroom的玻璃灯），应该再优化一下透射光和镜面反射光的算法。
3. 函数封装做的不够好，类与类之间的耦合性可以降低，继承关系可以再优化（比如点光源和面光源都继承自光源基类）。