VCL-Final Project:Photon Mapping

xTryer

2025/1/15

1 Introduction

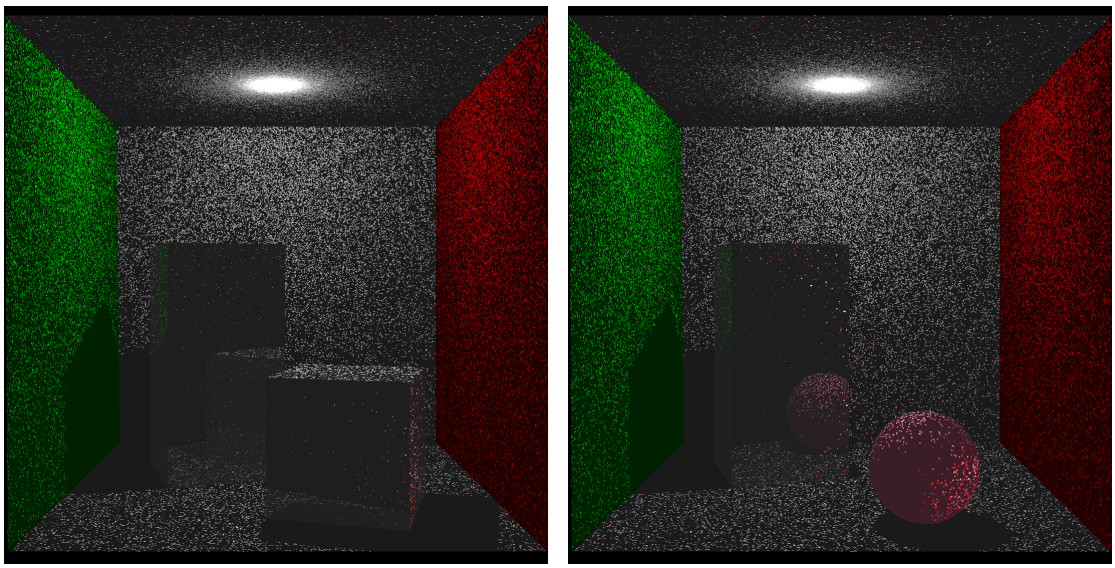
Photon Mapping(光子映射) 是一种用于实现全局光照的渲染技术，它通过追踪光子的路径来模拟光在场景中的传播，特别适合处理复杂的光照效果, 相对于传统 RayTracing 能够获得较柔和精致的渲染效果。

2 Methods

Photon Mapping 的实现主要分为两个部分: 1. 追踪光子，创建光子贴图 (PhotonMapping), 2. 追踪路径，收集邻近的光子以进行渲染.

2.1 Photon Tracing Pass

在这一阶段，我们实现光子的追踪，并创建光子贴图以供在下一阶段中进行渲染。我们采用的主要思想是：对于场景中的每一个光源，利用其光强 (Intensity) 代表其发光功率，每个光源发射出 photons_num 个光子，每个光子平分该光源的功率。产生的光子初始方向随机分布，在场景中不断发生折射、反射，每次折射或反射都会产生一个新的 photon, 这里我们规定——对于单个光子而言，迭代上限为 bounce_max。然后依据材质属性，按照 BSDF 模型进行新 photon 的计算。记录场景中所有散落的光子，存在一个全局 photons_vec 中，以便在阶段 2 中进行光子的收集和渲染。(这一部分在 PhotonMapping.h-PhontonMapping-Init_TracePhotons 中实现)



(a) 200000 photons per light-showcase1　　(b) 200000 photons per light-showcase2

图 1: Photon Tracing

2.2 Rendering Pass

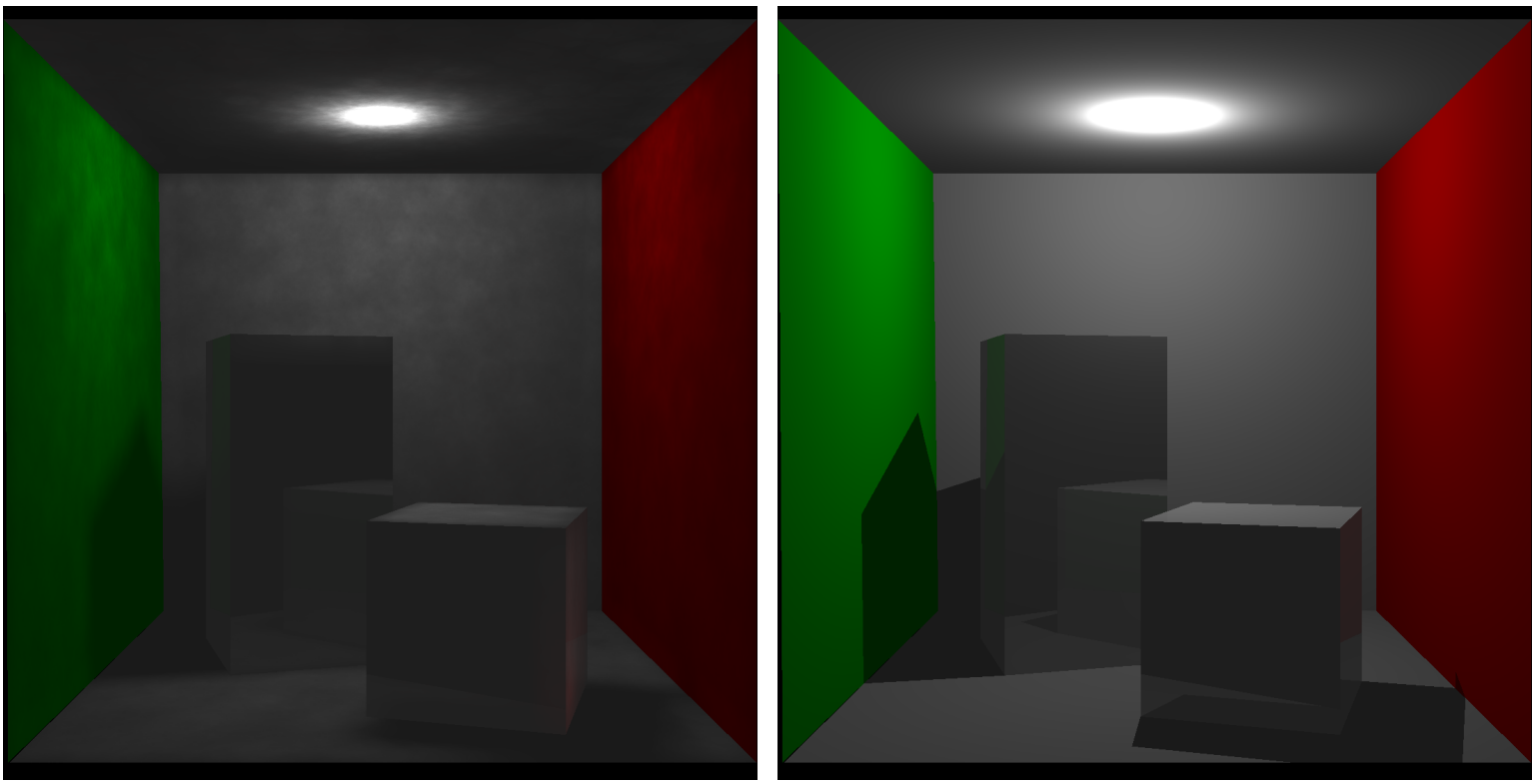
在这一阶段，我们追踪路径，对于相机上每个像素发出的光线进行追踪，收集邻近的 near_Phonton_num 个光子进行渲染。具体而言，对于相机光线与场景中的每一个交点，我们查询在一定半径范围内与之邻近的光子，然后利用这些光子的功率和发射方向按照 Blinn-Phong 模型计算出它们的着色贡献，最后通过密度估计归一化处理得到最后的辐射值。(这一部分主要在 PhotonMapping.cpp-RayTraceWithPhotonMapping 与 PhotonMapping.h-PhotonMapping-calculate_photon_radiance 中实现)

2.3 KD-Tree to Accelerate

在实际渲染过程中，我发现在渲染阶段查询与交点相邻的 k 个光子的时间开销较大，对此我使用 K-Dimensional Tree(KD-Tree) 的数据结构来实现场景中光子的高效存储和查询，实测发现在光子数目较大时，使用 KD-Tree 对于渲染过程有显著的加速效果。具体来说，我利用先前存储光子的 photons_vec 来构建 KD-Tree，KD-Tree 通过空间划分和剪枝的策略来缩小检索范围，从而提高查询效率。此外，我还根据光子密度动态调整了搜索半径，以确保每个查询点周围有足够的光子用于计算。(这一部分主要在 PhotonMapping.h-KDTree 与 PhotonMapping.h-PhotonMapping-calculate_photon_radiance 中实现)

3 Visualized Results

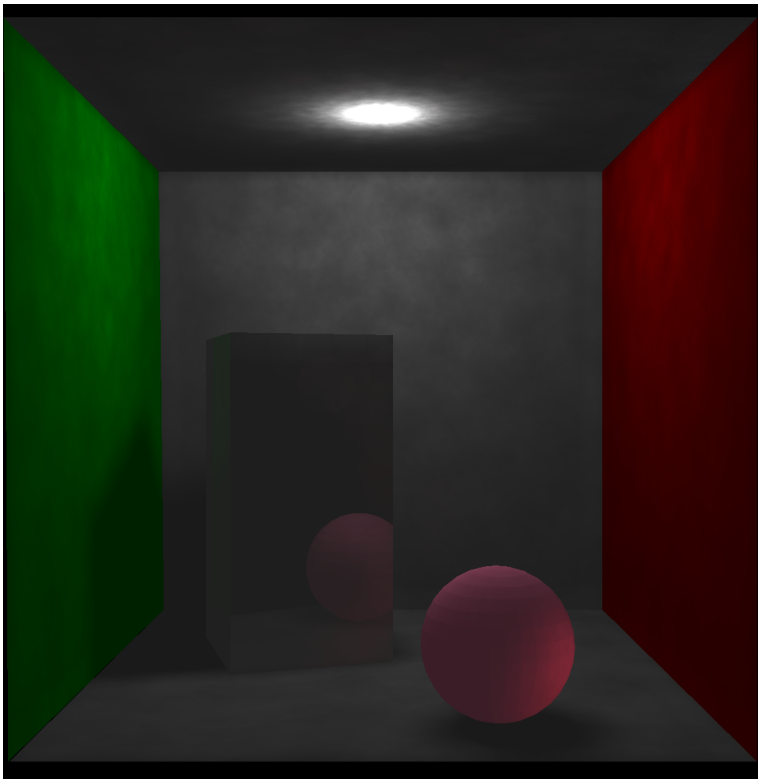
我们将参数设置为每个光源发射出 200000 个光子，光子最多折射、反射 5 次，每个查询点检索邻近的 140 个光子，检索初始半径为 30，同时，我们利用传统 RayTracing 的方法进行渲染，最终得到渲染结果对比图如下:



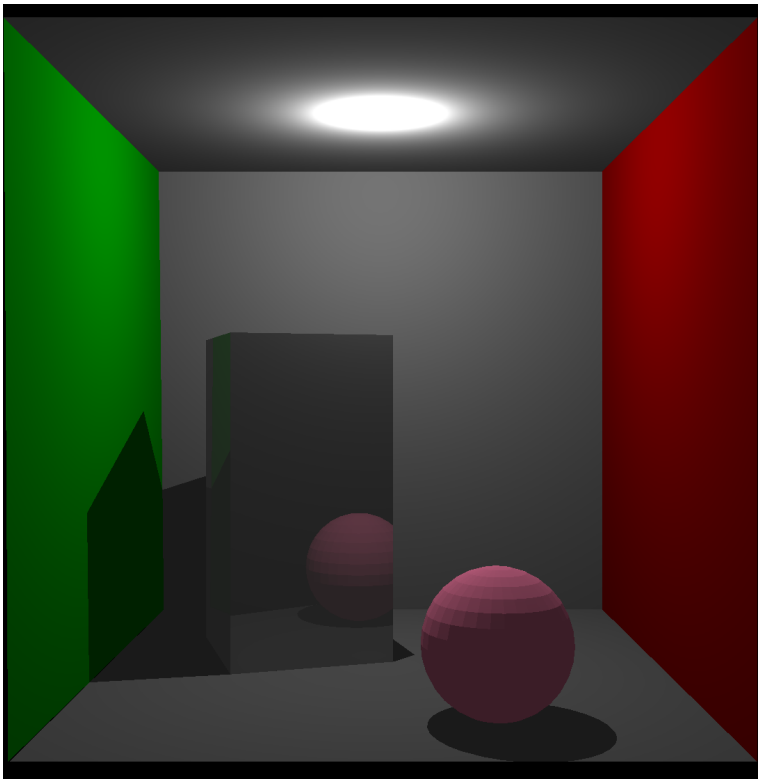
(a) Photon Rendering-showcase1

(b) RayTracing Rendering-showcase1

图 2: Photon Rendering vs RayTracing Rendering



(a) Photon Rendering-showcase2



(b) RayTracing Rendering-showcase2

图 3: Photon Rendering vs RayTracing Rendering

通过对比可以发现传统 RayTracing 渲染得到的是硬阴影，边缘清晰，渲染结果缺乏真实感，而 Photon-Mapping 渲染得到的是软阴影，边缘柔和，且能精确模拟光线多次反射，呈现出来的效果更具实感。

4 可视化页面说明

在 Lab 可视化框架的基础上，我主要调整了可视化的参数面版，如图所示，各项的功能分别为: 1-切换场景,2-采样率,3-光线追踪深度,4-每个光源发射出 photon 数量,5-每个查询点检索的相邻光子数 6-每个 photon 反射/透射次数, 7-每个查询点检索光子时的范围, "KD-Tree Accelerate"-是否启用 KD-Tree 加速渲染。

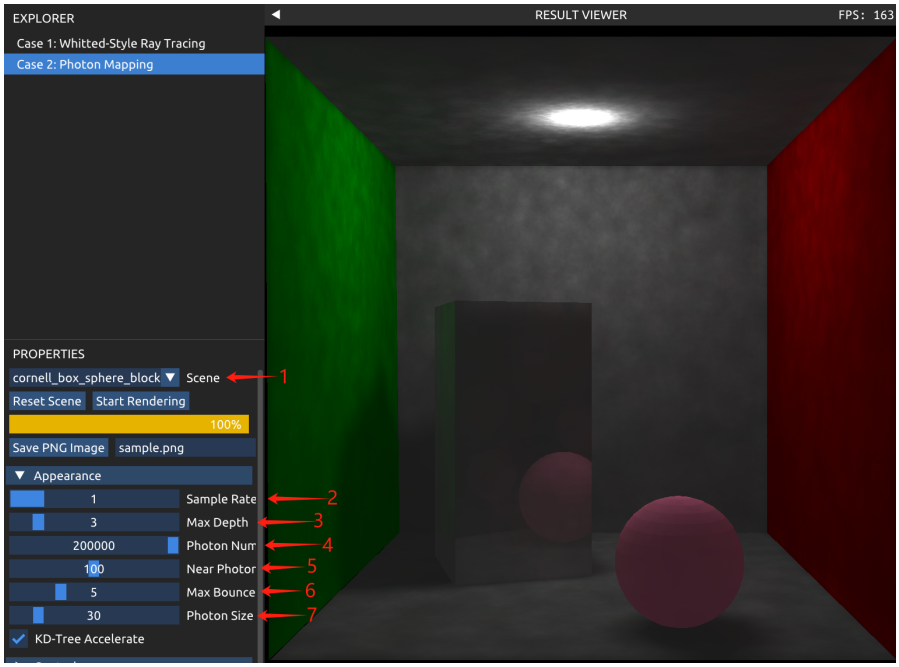


图 4: Visualization Introduction