上海交通大學

SHANGHAI JIAO TONG UNIVERSITY

课程报告

COURSE PROJECT

课程：计算机图形学

学生姓名：杨海辰
学生学号：122034910134
专　　业：电子与通信工程
任课教师：马利庄，易冉
学院(系)：电子信息与电气工程学院

目 录

一、 基本原理 1

1. 主要工作 1

2. 网格和几何体 1

3. 加速结构 3

4. 着色模型和材质 3

5. 全局光照算法 5

二、 实现 7

1. 代码简述 7

2. 文件说明 8

三、 实验结果 9

四、 总结 11

一、 基本原理

1. 主要工作

大作业的主题为 Project1 中的全局光照模型。

大作业的主要工作是基于路径追踪算法实现了一个简单的渲染器，支持球体、圆柱体、圆锥和三角形网格，其中三角形网格可以从.obj 文件导入，并支持纹理映射、法线贴图、置换贴图。路径追踪算法使用 BVH 作为光线与物体求交的加速结构。光照模型方面，实现了 blinn-phong 模型、基于微表面模型的 PBR 材质以及简单的透射模型，且全部实现了重要性采样。

大作业为 C++编写，使用了 OpenCV 来导入纹理以及输出图像。

2. 网格和几何体

渲染器支持球体、圆柱、圆锥以及三角形网格。在这一部分将逐一介绍以上几何体的描述方法，以及光线如何与这些几何体求交，并获取交点的法线、纹理坐标等信息。

（1）球体

渲染器中的球体使用中心点和半径来描述，方程为：

$$\|\mathbf{p}-\mathbf{c}\|=r \tag{1.1}$$

对于光线

$$\mathbf{p}=\mathbf{o}+t\mathbf{d} \tag{1.2}$$

它与球体的交点可通过求解方程

$$t^2+2t(\mathbf{d}\cdot(\mathbf{o}-\mathbf{c}))+(\mathbf{o}-\mathbf{c})\cdot(\mathbf{o}-\mathbf{c})-r^2=0 \tag{1.3}$$

得到，而交点的法线为

$$\mathbf{n}=\frac{\mathbf{p}-\mathbf{c}}{\|\mathbf{p}-\mathbf{c}\|} \tag{1.4}$$

（2）圆柱

圆柱的属性包含中心、半径和长度。将圆柱的侧面定义为：

$$\begin{aligned}(x-x_c)^2+(z-z_c)^2&=r^2\\ y_c-\frac{h}{2}\le y&\le y_c+\frac{h}{2}\end{aligned} \tag{1.5}$$

底面定义为：

$$\begin{aligned}(x-x_c)^2+(z-z_c)^2&\le r^2\\ y&=y_c\pm\frac{h}{2}\end{aligned} \tag{1.6}$$

这里只定义了地面与 y 轴垂直的圆柱。如果要将圆柱进行旋转，可以通过对光线进行对应旋转来实现。

光线与圆柱求交的方法为逐一计算可能产生的交点，并取距离最小的可用交点作为结果。为了便于计算，首先将圆柱和光线进行平移，使圆柱中心位于原点。

首先考虑圆柱的侧面，将光线带入方程得到：

$$(o_x+td_x)^2+(o_z+td_z)^2=r^2 \tag{1.7}$$

所以求解方程

$$(d_x^2+d_z^2)t^2+2(d_xo_x+d_zo_z)t+o_x^2+o_z^2-r^2=0 \tag{1.8}$$

就能得到光线与侧面的交点。而对于上下底面的交点，只需计算光线与上下底面所在平面的交点，然后验证该交点是否在圆柱表面。

当最终确定的交点在圆柱侧面时，法线为

$$\mathbf{n}=normalize(x-x_c,\ 0,\ z-z_c)^{\mathrm{T}} \tag{1.9}$$

当交点在底面时，法线与 y 轴平行。

（3）圆锥

圆锥的属性包含中心、高度和斜率。同样地，也只定义了底面与 y 轴垂直的情况。这里的中心代表横截面半径为零处，高度代表底面相对中心在 y 轴上的位移，而斜率满足如下关系：

$$r=k|y-y_c| \tag{1.10}$$

圆锥侧面的表达式为：

$$\begin{gathered}(x-x_c)^2+(z-z_c)^2=k^2(y-y_c)^2\\ \begin{cases} y_c+h\le y\le y_c & h<0\\ y_c\le y\le y_c+h & h>0 \end{cases}\end{gathered} \tag{1.11}$$

底面为：

$$\begin{gathered}(x-x_c)^2+(z-z_c)^2\le k^2(y-y_c)^2\\ y=y_c+h\end{gathered} \tag{1.12}$$

计算光线与圆锥交点时，首先也将圆锥和光线进行平移，使圆锥中心位于原点，然后逐个验证光线与圆锥可能产生的交点。

对于侧面，代入光线：

$$(o_x+td_x)^2+(o_z+td_z)^2=k^2(o_y+td_y)^2 \tag{1.13}$$

于是求解方程

$$(d_x^2+d_z^2-k^2d_y^2)t^2+2(o_xd_x+o_zd_z-k^2o_yd_y)t+o_x^2+o_z^2-k^2o_y^2=0 \tag{1.14}$$

就能得到侧面的交点。对于底面，只需计算光线与底面所在平面的交点，然后验证该交点是否在圆锥表面。

当最终确定的交点在圆锥侧面时，法线为

$$n=normalize\big(normalize(x-x_c,\ 0,\ z-z_c)^{\mathrm{T}}-\mathrm{sgn}(h)\,(0,\ |k|,\ 0)^{\mathrm{T}}\big) \tag{1.15}$$

当交点在底面时，法线与 y 轴平行。

（4）网格

渲染器中的三角形网格被存储为一个三角形列表。每个三角形中都包含顶点坐标、纹理坐标、顶点法线和切线数据。

计算光线和三角形交点的方法为：

$$\begin{bmatrix} t \\ b_1 \\ b_2 \end{bmatrix} = \frac{1}{\mathbf{S_1} \cdot \mathbf{E_1}} \begin{bmatrix} \mathbf{S_2} \cdot \mathbf{E_2} \\ \mathbf{S_1} \cdot \mathbf{S} \\ \mathbf{S_2} \cdot \mathbf{d} \end{bmatrix} \tag{1.16}$$

其中

$$\begin{aligned} \mathbf{E_1} &= \mathbf{p_1} - \mathbf{p_0} \\ \mathbf{E_2} &= \mathbf{p_2} - \mathbf{p_0} \\ \mathbf{S} &= \mathbf{o} - \mathbf{p_0} \\ \mathbf{S_1} &= \mathbf{d} \times \mathbf{E_2} \\ \mathbf{S_2} &= \mathbf{S} \times \mathbf{E_1} \end{aligned} \tag{1.17}$$

通过公式(1.5)的结果可以得到光线与三角形的交点以及交点在三角形内的重心坐标。只有当重心坐标的值全为正时，交点位于三角形内部。而重心坐标同时也用于对三角形的顶点法线、纹理坐标进行线性插值以实现平滑着色，即：

$$\mathbf{a} = b_1 \mathbf{a}_1 + b_2 \mathbf{a}_2 + b_3 \mathbf{a}_3 \tag{1.18}$$

为了支持法线贴图，在初始化三角形时也会根据顶点的纹理坐标计算切线，计算方法为：

$$\mathbf{T} = normalize\left(\frac{(v_2 - v_1)(\mathbf{p_2} - \mathbf{p_1}) - (v_3 - v_1)(\mathbf{p_3} - \mathbf{p_1})}{(v_2 - v_1)(u_3 - u_1) - (v_3 - v_1)(u_2 - u_1)} \right) \tag{1.19}$$

其中 (u, v) 表示顶点的纹理坐标。在光线与三角形进行求交之后，如果检测到材质中包含法线贴图，则利用交点法线和三角形切线建立局部坐标系，再根据从法线贴图中采样得到的切线空间法线计算得到该点的法线。

3. 加速结构

在路径追踪算法中，光线与场景求交的计算量较大。为了加速该操作，渲染器使用 BVH 来存储场景中的图元。当所有模型加载完成后，会根据各个图元的包围盒构建 BVH，在渲染过程中使用。

简单介绍构建 BVH 的方法。对于一个已知的 BVH 节点，首先根据各个图元的包围盒在某一坐标分量上的最小值来对图元进行空间上的排序，然后就可以将图元按数量平均分为两部分，分别作为该 BVH 节点的两个子节点。重复上述操作，直到每个 BVH 叶子节点只对应一个图元。计算三角形、球形、圆柱和圆锥包围盒的方法较为简单，这里不赘述。

BVH 构建完毕之后，在渲染阶段，光线与场景求交的过程就转换为光线与 BVH 根节点求交的过程。而当光线与 BVH 节点求交时，若该节点不是叶子节点，则分别计算光线与该节点的两个子节点的包围盒的求交结果，如果光线与某个子节点的包围盒存在交点，则接着计算光线与该子节点的求交结果。由于当光线与某个子节点的包围盒无交点时，该分支会直接被丢弃，所以使用 BVH 可以有效地降低光线与场景求交的开销。

4. 着色模型和材质

渲染器实现了三种着色模型，分别为 Lambert 模型、blinn-phong 模型和 GGX 微表面模型。其中重点为 GGX 微表面模型，它可以直接用于描述金属材质，也可以结合 Lambert 模

型来描述非金属材质。为了增加表面粗糙度较小时渲染结果的收敛速率，对于以上三种模型都实现了重要性采样，其中 Lambert 模型使用 cos-weighted 采样，其他两种模型则通过采样半程向量来实现重要性采样。另外，也实现了考虑折射的透明材质。

接下来将逐一介绍渲染器中使用的着色模型，以及实现的材质。

（1）Lambert 模型

Lambert 模型通常用于描述 diffuse 表面，它假设光的反射强度不随相机的观察角度变化而变化，而只与光线的入射角度有关。

考虑渲染方程

$$L_o = L_e + \int L(\mathbf{l}) f(\mathbf{l},\ \mathbf{v}) \cos\theta \,\mathrm{d}\mathbf{l} \tag{1.20}$$

Lambert 模型的 BRDF 为

$$f_{diffuse}(\mathbf{l},\ \mathbf{v}) = \frac{\mathbf{c}}{\pi} \tag{1.21}$$

由于 Lambert 模型的 BRDF 为常数，所以当假设入射光强度在上半球面为均匀分布时，使用 cos-weighted 采样方法能得到最好的结果，其 PDF 为：

$$p(\mathbf{h}) = \frac{\cos\theta}{\pi} \tag{1.22}$$

随机生成 cos-weighted 采样结果的方式为：

$$\begin{aligned} &\xi_1,\ \xi_2 \sim U(0,\ 1) \\ &\theta = \arccos\sqrt{1-\xi_1} \\ &\phi = 2\pi\xi_2 \end{aligned} \tag{1.23}$$

利用 Lambert 模型可以直接描述 diffuse 材质。

（2）Blinn-phong 模型

Blinn-phong 模型是一种比较简单的光照模型，BRDF 为

$$f_{blinn\_phong}(\mathbf{l},\ \mathbf{v}) = k_d \frac{1}{\pi} + k_s \frac{n+2}{2\pi} \cos^n \alpha \tag{1.24}$$

其中 k_d 和 k_s 用于控制模型的 diffuse 和 specular 系数，n 用于控制材质粗糙度，而 α 为半程向量 h 和法线的夹角。

相比 Lambert 模型，Blinn-phong 模型为材质增加了 specular 成分。而在进行重要性采样时，也可以利用此特点，以 $k_d/(k_d+k_s)$ 的概率采样 diffuse 部分，否则采样 specular 部分。而 specular 部分的重要性采样可以通过采样 **h** 得到，方法为：

$$\begin{aligned} &\xi_1,\ \xi_2 \sim U(0,\ 1) \\ &\theta = \arccos \xi_1^{\frac{1}{n+1}} \\ &\phi = 2\pi\xi_2 \end{aligned} \tag{1.25}$$

而 PDF 可以通过

$$p(\mathbf{h}) = \frac{n+1}{2\pi} (\mathbf{n} \cdot \mathbf{h})^n \tag{1.26}$$

得到。采样到 **h** 后，就能根据 **v** 和 **h** 计算 **l**。如果得到的 **l** 与法线方向相反，则将这次采样得到的光照结果设为 0。

利用 Blinn-phong 模型可以描述带有光泽的非金属材质，类似塑料。

（3）GGX 模型

GGX 模型是一个比较常用的微表面模型。微表面模型使用法线分布来描述 specular 表面，同时考虑了菲涅尔效应和微观的遮挡情况。Cook-Torrance 微表面模型 BRDF 为：

$$f_{microfacet}(\mathbf{l},\ \mathbf{v}) = \frac{D(\mathbf{h})F(\mathbf{v},\ \mathbf{h})G(\mathbf{l},\ \mathbf{v},\ \mathbf{h})}{4(\mathbf{n}\cdot\mathbf{l})(\mathbf{n}\cdot\mathbf{v})} \tag{1.27}$$

其中 D 为法线 NDF 项，F 为菲涅尔项，G 为几何项。

GGX 法线分布为

$$D(\mathbf{h}) = \frac{\alpha^2}{\pi\left((\mathbf{n}\cdot\mathbf{h})^2(\alpha^2-1)+1\right)^2} \tag{1.28}$$

其中$\alpha = Roughness^2$。对于菲涅尔项，这里采用了 Schlick's approximation，即

$$F(v,\ h) = F_0 + (1-F_0)(1-\mathbf{v}\cdot\mathbf{h})^5 \tag{1.29}$$

这里的 F_0 是法线方向的 specular 反射系数，如果是金属材质，这一项就是其 base color。对于几何项，这里使用了 GGX 模型的 Smith 近似，假设入射光和反射光的遮挡情况互相独立，即

$$\begin{gathered} k = 0.125(Roughness+1)^2 \\ G_1(\mathbf{v}) = \frac{\mathbf{n}\cdot\mathbf{v}}{(\mathbf{n}\cdot\mathbf{v})(1-k)+k} \\ G(\mathbf{l},\ \mathbf{v},\ \mathbf{h}) = G_1(\mathbf{l})G_1(\mathbf{v}) \end{gathered} \tag{1.30}$$

接下来考虑 GGX 模型的重要性采样。由于 G 和 F 项的值都较为平坦，所以可以利用 NDF 对半程向量进行重要性采样，使其 PDF 为：

$$p(\mathbf{h}) \sim D(\mathbf{h})\cos\theta = \frac{\alpha^2\cos\theta}{\pi\left((\alpha^2-1)\cos^2\theta+1\right)^2} \tag{1.31}$$

于是采样 **h** 的方法为：

$$\begin{gathered} \xi_1,\ \xi_2 \sim U(0,\ 1) \\ \theta = \arctan\left(\alpha\sqrt{\frac{\xi_1}{1-\xi_1}}\right) \\ \phi = 2\pi\xi_2 \end{gathered} \tag{1.32}$$

GGX 模型可直接用于描述金属材质。同时，它也可以作为非金属的 specular 项，可取非金属的菲涅尔系数为 0.04，那么

$$f_{nonmetal}(\mathbf{l},\ \mathbf{v}) = 0.04f_{GGX}(\mathbf{l},\ \mathbf{v}) + 0.96f_{diffuse}(\mathbf{l},\ \mathbf{v}) \tag{1.33}$$

（4）透明材质

渲染器实现了最基本的透明材质，这里的方法是根据折射定律

$$n_1\sin\theta_1 = n_2\sin\theta_2 \tag{1.34}$$

来计算折射后的光线。同样地，使用反射模型和折射模型的加权和来描述透明材质。

5. 全局光照算法

全局光照采用路径追踪算法以及蒙特卡洛积分进行计算。对于一个 shading point，它在

方向 $\mathbf{v}$ 的出射 radiance 使用蒙特卡洛积分可以表示为：

$$L_o = L_e + \frac{1}{N}\sum_{k=1}^{N}\frac{L_i(\mathbf{l}_k)f(\mathbf{l}_k,\ \mathbf{v})\cos\theta_{\mathbf{l}_k}}{p(\mathbf{l}_k,\ \mathbf{v})} \tag{1.35}$$

其中 $\mathbf{l}_k$ 通过随机采样生成，$p(\mathbf{l}_k,\mathbf{v})$ 为其 PDF。想要得到较为准确的 shading 结果，需要进行多次采样，此时如果同时计算光的多次反射，那么会导致采样光线数量过大。所以这里才用的路径追踪方法中，每次计算蒙特卡洛积分时只进行一次采样，但是对于每一个像素可以多次计算路径追踪结果从而实现多次采样。

由于 BRDF 以及入射光都并不是均匀分布，所以如果不使用重要性采样会导致计算结果的噪声非常大，特别是在材质非常光滑的情况下。于是，分别根据材质的 BRDF 以及场景中光源进行了重要性采样，前者已经包含在着色模型和材质这一节中，而接下来将介绍如何实现对面光源的重要性采样。

面光源重要性采样的思路就是将面光源上的面积微元转换为 shading point 上半球面的立体角微元，关系式为：

$$d\mathbf{l} = \frac{dA\cos\theta'}{\|\mathbf{p}'-\mathbf{p}\|^2} \tag{1.36}$$

其中 θ' 为面光源法线与光线方向的夹角，而 $\|\mathbf{p}'-\mathbf{p}\|$ 是面光源采样点到 shading point 的距离。所以蒙特卡洛积分形式的单个光源对 shading point 的光照贡献为：

$$L_o = \frac{1}{N}\sum_{k=1}^{N}\frac{L_i(\mathbf{l}_k)f(\mathbf{l}_k,\ \mathbf{v})\cos\theta_{\mathbf{l}_k}\cos\theta'_{\mathbf{l}_k}}{p(\mathbf{l}_k,\mathbf{v})\|\mathbf{p}'-\mathbf{p}\|^2 p_{light}(\mathbf{l}_k)} \tag{1.37}$$

所以，计算光照步骤可以分为计算直接光贡献和间接光贡献两部分：首先在每个面光源上进行采样计算直接光照，然后随机采样入射光与场景求交计算间接光照，二者之和加上表面自发光就是最终的光照结果。这里计算直接光的时候应考虑 shading point 到光源采样点之间是否存在遮挡的问题，所以需要进行一次光线与场景求交的操作。

渲染器实现了三角形面光源和圆形面光源。在三角形面光源上均匀采样的方法为：

$$\begin{gathered}\xi_1,\ \xi_2 \sim U(0,\ 1)\\ (u,\ v) = \begin{cases}(\xi_1,\ \xi_2) & \xi_1+\xi_2 \le 1\\ (1-\xi_1,\ 1-\xi_2) & \xi_1+\xi_2 > 1\end{cases}\end{gathered} \tag{1.38}$$

在圆形面光源上均匀采样的方法为：

$$\begin{gathered}\xi_1,\ \xi_2 \sim U(0,\ 1)\\ r = \sqrt{\xi_1}R\\ \phi = 2\pi\xi_2\end{gathered} \tag{1.39}$$

二、 实现

1. 代码简述

渲染器主要包含以下模块：

(a) ray：描述光线。
(b) camera：描述相机。
(c) hittable：所有参与光线求交的图元的基类，triangle、sphere、cylinder、cone 均继承该类型。
(d) bvh_node：用于构建 bvh。
(e) texture：所有纹理的基类，图像纹理、法线贴图和置换贴图均继承该类型。
(f) material：所有材质的基类，其子类包括 diffuse 材质、blinn-phong 材质、金属材质、非金属材质、透明材质等。
(g) light：所有光源的基类，其子类包括圆形面光源和三角形面光源。
(h) mesh_triangle：配合开源的.obj 文件读取脚本 OBJ_Loader.h，可以将 mesh 以及其材质信息存储为 triangle 数组并加载到场景中。
(i) renderer：包括主要的路径追踪逻辑。
(j) main：主函数。

接下来将简单介绍各个模块的实现方法。

(a) ray

ray 类存储了光线的起点、方向和介质信息。不过介质信息目前并没有使用，在透明材质中使用的是 material 中存储的内外表面介质信息来获取介质的折射率。

(b) camera

camera 类存储了相机的位置、fov 和焦距信息。给出画布上的 uv 坐标就能使用 get_ray()函数获取路径追踪所使用的光线。不过这里并没有实现相机的旋转，相机的朝向为-z 方向，且+y 方向在上。

(c) bvh_node

每个 bvh_node 都包含一个包围盒和子节点的指针。在构造函数中需要传入场景中的 hittable 数组以及该 node 负责的 hittable 索引范围用于构建 bvh。

(d) hittable

作为所有参与光线求交的图元的基类，包含两个函数，其中 bounds()返回其包围盒，而 hit()则根据给定的光线计算相交情况，如果光线与该图元存在交点，则返回一个 hit_record。而 hit_record 包含了交点的位置、法线、材质纹理坐标、材质等信息，用于计算光照结果。

(e) texture

包含了 get_value()和 get_alpha()函数。

get_value()函数用于在纹理上采样，输入为纹理坐标，如果是图像纹理则返回采样点的

RGB 值，如果是法线贴图则返回采样点的切线空间法线，如果是置换贴图则返回采样点的位移量。这里实现图像纹理时需要首先将 RGB 值从非线性空间转换到线性空间。此外，采样方法只实现了双线性采样。

get_alpha()函数返回图像纹理的透明度。

(f) material

material 类完整地描述了物体表面的材质信息，包含了一系列非常重要的方法。

首先，material 类的子类必须包含 get_color_map_ptr()、get_normal_map_ptr()和 get_displacement_map_ptr()函数用于返回其图像纹理、法线贴图和置换贴图的指针。另外 get_medium_outside_ptr()和 get_medium_inside_ptr()函数返回其内侧和外侧的介质信息。get_radiance()则返回材质的自发光强度。hit_record 结合以上函数可以得到 shading point 的完整表面信息用于计算光照结果。

sample_wi()和 sample_positioni()函数用于采样光线的入射方向和入射位置来实现基于光照模型的重要性采样。对于反射模型只需要利用 sample_wi()就能实现重要性采样，而设计 sample_positioni()函数是为了支持基于 BSSRDF 的次表面散射材质，但该材质目前还没有正确实现。

最后，bsdf()函数用于计算 BRDF（反射模型使用）或 BTDF（透射模型使用）的值。

(g) light

light 类中只需包含一个 sample_p()函数，用于在面光源上进行重要性采样，返回采样点的位置、radiance、法线和 PDF。这里只实现了常数光强的面光源。

(h) mesh_triangle

这里首先利用 OBJ_Loader.h 从.obj 文件中读取三角形网格中的顶点位置、顶点法线、顶点纹理坐标、三角形顶点索引和材质信息，然后根据以上数据组装三角形，可返回一个 triangle 数组。

(i) renderer

基于路径追踪算法进行渲染。这里在每一个画布上的像素中都随机生成采样点来投射光线，从而实现抗锯齿。另外，没有采用俄罗斯轮盘赌方法，而是限制了光线的最大弹射次数。

(j) main

主要包含设置相机、设置场景、设置光源、设置渲染线程、图片文件写入等操作。

2. 文件说明

源码在 SimpleRT/code/src 文件夹中，模型和贴图在 SimpleRT/exec/obj 文件夹中。

在 SimpleRT/exec 文件夹中使用命令“.\SimpleRT.exe samples_per_pixel”渲染默认场景，分辨率为 1024*1024，结果输出到 SimpleRT/exec/out 文件夹，其中 samples_per_pixel 为采样率，默认值为 1。如果需要设置其他参数，可以在 main 函数中进行修改。

三、 实验结果

设置输出分辨率为 1280*1280，采样率为 2048，光线最大弹射次数为 4，渲染得到的结果如图 1 所示。

图 1 渲染结果 1

场景中共包含两个正方形面光源，他们都由两个三角形面光源组合而成。

几何体包括球形、圆柱和圆锥，也包含网格，其中网格包括数个模型，在 blender 中编辑它们的摆放方式并导出为.obj 文件后再导入渲染器。场景中共包含约 80 万个三角形，生成了 1048575 个 BVH 节点。

场景中圆锥和球体采用了基于微表面模型的金属材质，兔子使用了透明材质并设置其折射率为 1.5，其余物体都采用非金属材质。

这里椅子的坐垫使用了置换贴图，墙壁、地面和盆栽使用了法线贴图。
从图 2 中可以更加直观地看到法线贴图的效果。这里正方体采用的是金属材质。

图 2 渲染结果 2

四、 总结

大作业基于路径追踪算法实现了一个简单的离线全局光照渲染器，支持球体、圆柱体、圆锥和三角形网格，并支持纹理映射、法线贴图、置换贴图。路径追踪算法使用 BVH 作为光线与物体求交的加速结构。光照模型方面，实现了 blinn-phong 模型、基于微表面模型的 PBR 材质以及简单的透射模型，且全部实现了重要性采样。

目前该渲染器能够得到较好的渲染结果，但依旧存在很多不足，比如不支持次表面散射、透射模型过于简单、不支持各向异性材质、不支持体积等等，而且在性能上没有做很多优化，也没有使用图像降噪方法改善渲染结果，在后续工作中可以加以完善。