计算机图形学 —— 蒙特卡罗光线跟踪算法

11821095 葛林林

2019 年 3 月 31 日

1 预备知识

1.1 光照模型

（1） **环境光**：环境光无处不在，无论表面的法向如何，明暗程度都是一致的。

（2） **点光源**：光源来自某个点，且向四面八方辐射。

（3） **平行光**：又称为镜面光，这种光是互相平行的。从手电筒或者太阳出来的光都可以被看做平行光。

（4） **聚光灯**：这种光源的光线从一个锥体中射出，在被照射的物体上产生聚光的效果。使用这种光源需要指定光的射出方向以及锥体的顶角α。

1.1.1 Lambertian光照模型（漫反射效果的仿真）

1.1.2 BlinnPhong光照模型（镜面反射效果的仿真）

（1） **发射光**

$$C_e = M_e$$

C_e：发射颜色
M_e：发射材质颜色

（2） **漫反射**
定义漫反射因子α如下：

$$\alpha = max\{0, N \cdot L\}$$

其中L是光线，N是定点法线向量。

$$C_d = L_d M_d \alpha$$

1.2 obj文件介绍

obj文件并不考虑物体的大小，所以不同的物体读入的坐标范围可能变化很大，因此为了显示的方便需将其转为当前绘制坐标系中。

（1） 库相关

- ***mtllib*** xxx
  材料库。
- ***usemtl*** xxx
  代表使用xxx类型的材质。

（2） 组相关

- ***g*** xxx
  表示组，将xxx标签之后的多边形组成一个整体。
- ***s*** xxx
  光滑组:加入光滑组之后能够让在同一组的多边形之间连接更为光滑，其中“s off”代表关闭光滑组。

（3） 坐标相关

- ***vt*** tu tv
  代表纹理坐标。
- ***vn*** nx ny nz
  法向量的表示。
- ***f*** $v/vt/vn$ $v/vt/vn$ $v/vt/vn$
  表示多边形，格式为“f 顶点索引/ 纹理坐标索引/ 顶点法向量索引”。
- ***v*** x y z
  顶点以v开头后面跟着该顶点的x, y, z三轴坐标。

1.3 mtl文件介绍

mtl文件是用来描述文件材质的一个文件，描述的是物体的材质信息，ASCII存储，任何文本编辑器可以将其打开和编辑。一个.mtl文件可以包含一个或多个材质定义，对于每个材质都有其颜色，纹理和反射贴图的描述，应用于物体的表面和顶点。描述的是物体的材质信息，ASCII存储，任何文本编辑器可以将其打开和编辑。一个.mtl文件可以包含一个或多个材质定义，对于每个材质都有其颜色，纹理和反射贴图的描述，应用于物体的表面和顶点。：

（1） **格式：*Ka*** r g b
示例：*Ks* 0.588 0.588 0.588
描述：环境反射，用RGB颜色值来表示，g和b两参数是可选的，如果只指定了r的值，则g和b的值都等于r的值。三个参数一般取值范围为[0.0,1.0]，在此范围外的值则相应的增加或减少反射率;

（2） **格式：*Kd*** r g b
示例：*Kd* 0.65 0.65 0.65
描述：漫反射，rgb代表了RGB值，范围为[0,1]。其中g和b是可选的，如果未设置这两个值，则g, b的值与r的值相同。

（3） **格式：*Ks*** r g b
示例：*Ks* 0.65 0.65 0.65
描述：镜面反射，rgb代表了RGB值，范围为[0,1]。其中g和b是可选的，如果未设置这两个值，则g, b的值与r的值相同。

（4） **格式：*Tf*** r g b

示例：*Tf* 0.65 0.65 0.65

描述：代表了透射滤波，任何光线穿透该物体时可以利用该参数进行透射滤波，该参数只让指定颜色的光线穿透物体。例如Tf 0 1 0，只允许所有的绿色光线穿透，而所有的红色和绿色光线则不能够穿透。rgb代表了RGB值，范围为[0,1]。其中g和b是可选的，如果未设置这两个值，则g, b的值与r的值相同。

（5） **格式：*illum*** $number$

示例：*illum* 2

描述：指定了光照模型。

（6） ***Ns*** **10.000000**

指定材质的反射指数，定义了反射高光度。exponent是反射指数值，该值越高则高光越密集，一般取值范围在$[0, 1000]$。

（7） ***Ni*** **1.500000**

指定材质表面的光密度（即折射值），取值范围为$[0.001, 10]$。若取值为1.0则光在通过物体的时候不发生弯曲。玻璃的折射率为1.5。取值小于1.0的时候可能会产生奇怪的结果不推荐。

（8） ***d*** **1.000000**

表示物体融入背景的数量，取值范围为$[0.0, 1.0]$，取值为1.0表示完全不透明，取值为0.0时表示完全透明。

1.4 空间包围盒的介绍

空间包围盒是一个物体的凸包，是为了物体与光线求交的方便而引入的概念。人们提出了很多种可用的空间包围盒，其中AABB包围盒最为简单和常用。AABB包围盒的所有边与坐标轴平行，其示意图如下所示，起哄蓝色的为物体，绿色表示的时该物体的AABB包围盒：

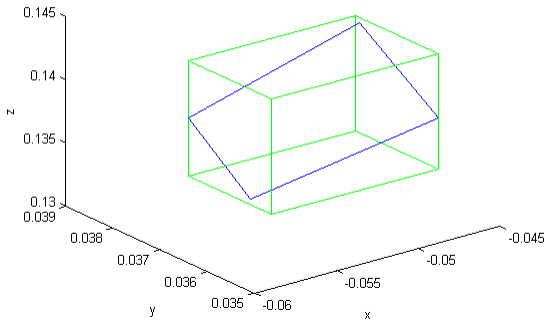


Figure 1: AABB包围盒示意图

1.5 Kd-Tree的介绍

Kd-tree（K-dimension tree）是一种拥有多维空间的快速最近邻查找技术，它是一种将BST扩展到多维的升级。在使用Kd-tree的过程中极其重要的两个操作是创建和查找。

1.5.1 Kd-Tree的构造

（1） **划分维度的选择**

kd-tree在举例时都会按照第一维、第二维、第三维的循环顺序进行构建，然而在实际使用当中最主要的目的是加速，也就是尽快减少检索范围。每一层使用更为分散的维度能够更好的将数据进行分离。最大方差法就是利用方差最大的维度对数据进行划分。

（2） **构造截止条件**

kd-tree在构造时满足一下任意条件则截止：

- 在该空间已经不存在数据点；
- 超过一定的深度则划定为叶子节点，该深度的阈值可以由自己来规定。

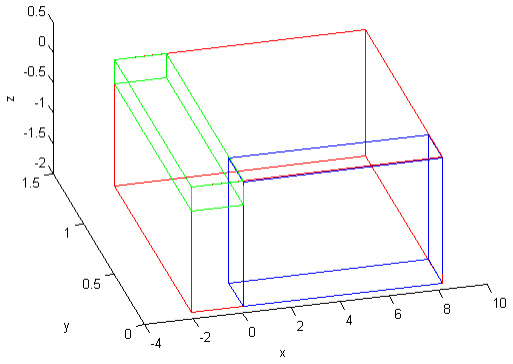


Figure 2: Kd-tree与AABB包围盒相结合的示意图

1.5.2 Kd-Tree的最近表面搜索策略

本文中用到的最近表面的搜索策略是在广度优先搜索的基础上，分层的进行搜索，其伪代码如下所示：

Algorithm 1 Nearest Surface Search by Level

```
B_s = Get current buffer size
for i = 0 to B_s do
    if It is leaf node then
        if This node is more close then
            Store this node
        end if
    else
        Insert two children
    end if
    Remove this first B_s number of node
end for
```

1.6 相机光线的产生

由于从相机出发逆向跟踪光路能够避免计算无法被看到的部分，从而大大降低计算量。下面介绍了如何从相机出发产生初始的逆向光线。假设相机的位置为$\boldsymbol{P}_{camera}$，Lookat的位置为$\boldsymbol{T}_0$，up的单位方向为D_{up}，则相机的单位前向方向$\boldsymbol{D}_{forward}$为：

$$\boldsymbol{D}_{forward} = \frac{\boldsymbol{T}_0 - \boldsymbol{P}_{camera}}{||\boldsymbol{T}_0 - \boldsymbol{P}_{camera}||}$$

除此之外，实际过程中所给的up方向并不是实际的相机朝上方向，如下图所示Supplied 'up' vector一般是所给的方向，而我们要去计算实际的Actual 'up' vector：

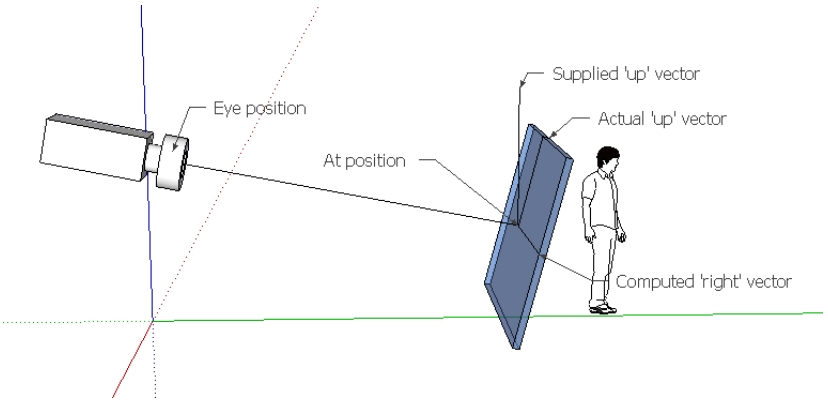


Figure 3: up方向的修正示意图

计算公式如下所示，注意保证$\boldsymbol{D}_{up}$和$\boldsymbol{D}_{forward}$保证是单位向量：

$$\theta = acos(\boldsymbol{D}_{up} \cdot \boldsymbol{D}_{forward})$$

则真正的$\boldsymbol{D}_{up}$可以有如下公式算出：

$$\boldsymbol{D}_{up} = \begin{cases} \boldsymbol{D}_{up}cos(\theta) & \theta > 0 \\ -\boldsymbol{D}_{up}cos(\theta) & \theta \leq 0 \end{cases}$$

则指向像平面右侧的单位方向为$\boldsymbol{D}_{right}$：

$$\boldsymbol{D}_{right} = \boldsymbol{D}_{forward} \times \boldsymbol{D}_{up}$$

假设单个像素的实际宽度为W，宽度像素的实际个数为N_w，单个像素的实际高度为H，高度方向是像素个数为N_h，相机的焦距为L，则像平面左上角像素在世界坐标系下的位置$P_{0,0}$为：

$$P_{0,0} = \boldsymbol{P}_{camera} + L\boldsymbol{D}_{forward} - N_w W D_{right} + N_h H D_{up}$$

则有则第i行第j列像素的实际位置可以表示如下所示：

$$\boldsymbol{P}_{i,j} = \boldsymbol{P}_{0,0} + W * j + H * i$$

1.7 求交检测

1.7.1 AABB包围盒与光线求交检测

1.7.2 凸多边形与光线求交检测

判断光线与凸多边形求交检测的步骤如下：

（1） **求取方程**

求取凸多边形对应的平面方程和光线所在直线方程，假设平面方程为：

$$Ax + By + Cz + D = 0$$

直线方程为:

$$\begin{cases} x = a_1 t + b_1 \\ y = a_2 t + b_2 \\ z = a_3 t + b_3 \end{cases}$$

（2） **求取交点**

计算凸多边形所在平面和光线所在直线方程的交点

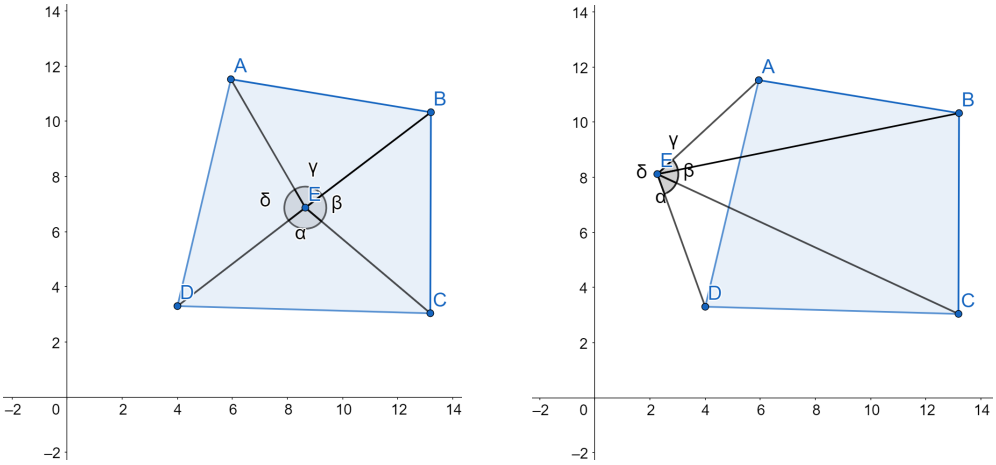
（3） **判断方向是否一致**

需要判断从光线原点出发到交点的射线方向是否与光线方向一致，假设光线原点为$\boldsymbol{r}_o$，光线方向为$\boldsymbol{r}_d$，交点坐标为$\boldsymbol{I}$，则有如下结论：

$$\boldsymbol{r}_d(\boldsymbol{I} - \boldsymbol{r}_o) = \begin{cases} < 0 & no \quad intersection \\ > 0 & not \quad sure \end{cases}$$

（4） **判断点和凸多边形关系**

如下图所示是交点与凸多边形的关系示意图



(a) 内部　　(b) 外部

Figure 4: 交点与凸多边形关系图

从上图中可以看出，内部的情况需要满足如下公式：

$$\alpha + \beta + \gamma + \sigma = 2\pi$$

当在外部的情况下满足如下关系：

$$\alpha + \beta + \gamma + \sigma < 2\pi$$

1.8 光线和球求交

如下图所示是光线和球求交的示意图：

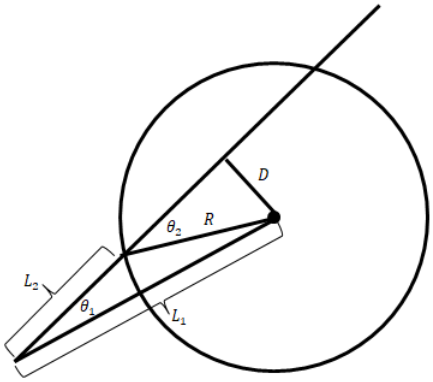


Figure 5: 光线和球求交检测示意图

则图中得到L的公式为：

$$\theta_1 = arcsin(\frac{D}{L_1})$$

$$\theta_2 = arsin(\frac{D}{R})$$

$$L_2 = \frac{D}{tan(\theta_1)} - \frac{D}{tan(\theta_2)}$$

2 步骤

如下图所示是整个光线跟踪算法的处理流程图：

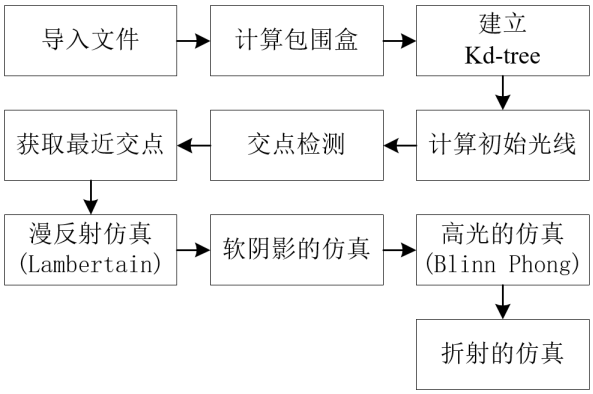


Figure 6: 整体流程图

3 实验结果

3.1 环境参数

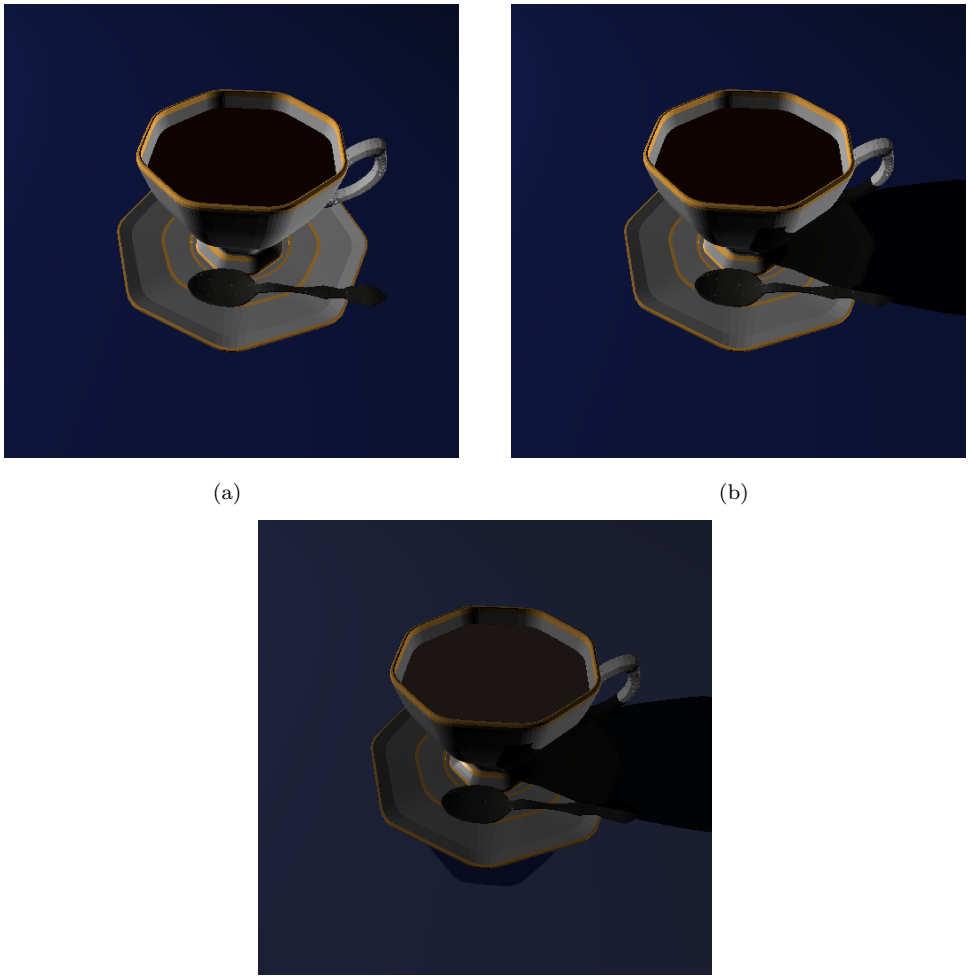
本报告中实验所使用的开发环境如下所示：

Table 1: 环境参数

参数	描述
System	Windows 10 64bit
CPU	Intel(R) Core(TM) i5-2410M CPU @2.30GHz 2.3GHz（4核）
RAM	6GB
IDE	Visual Studio 2017

3.2 效果图片

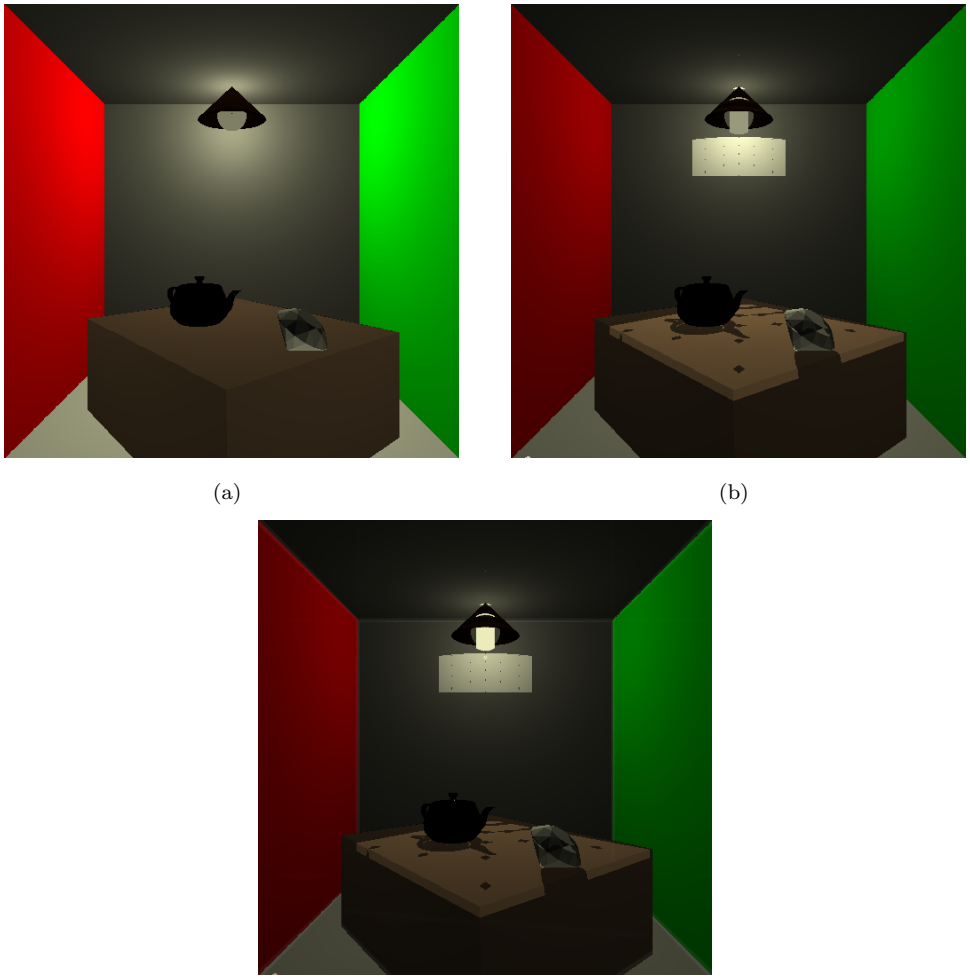
本实验中分别展示了利用Lambertain算法的漫反射仿真结果，以及利用阴影测试线得到的阴影仿真结果，以及利用Blinn Phong算法得到的高光仿真结果图。由于场景obj文件中包含了光源的三角片，因此场景中出现多余的阴影。



(a) (b)

(c)

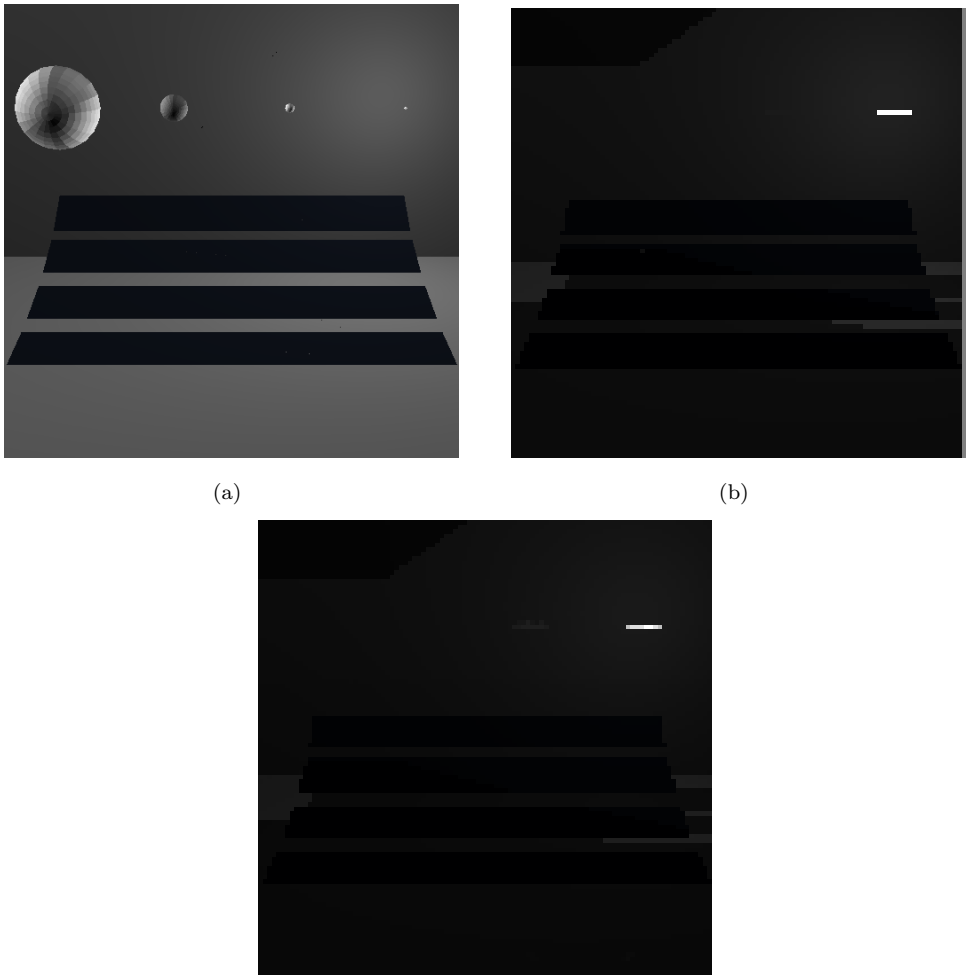
Figure 7: （a）cup场景漫反射仿真结果图（b）cup场景阴影仿真结果图（c）cup场景高光仿真结果图



(a) (b)

(c)

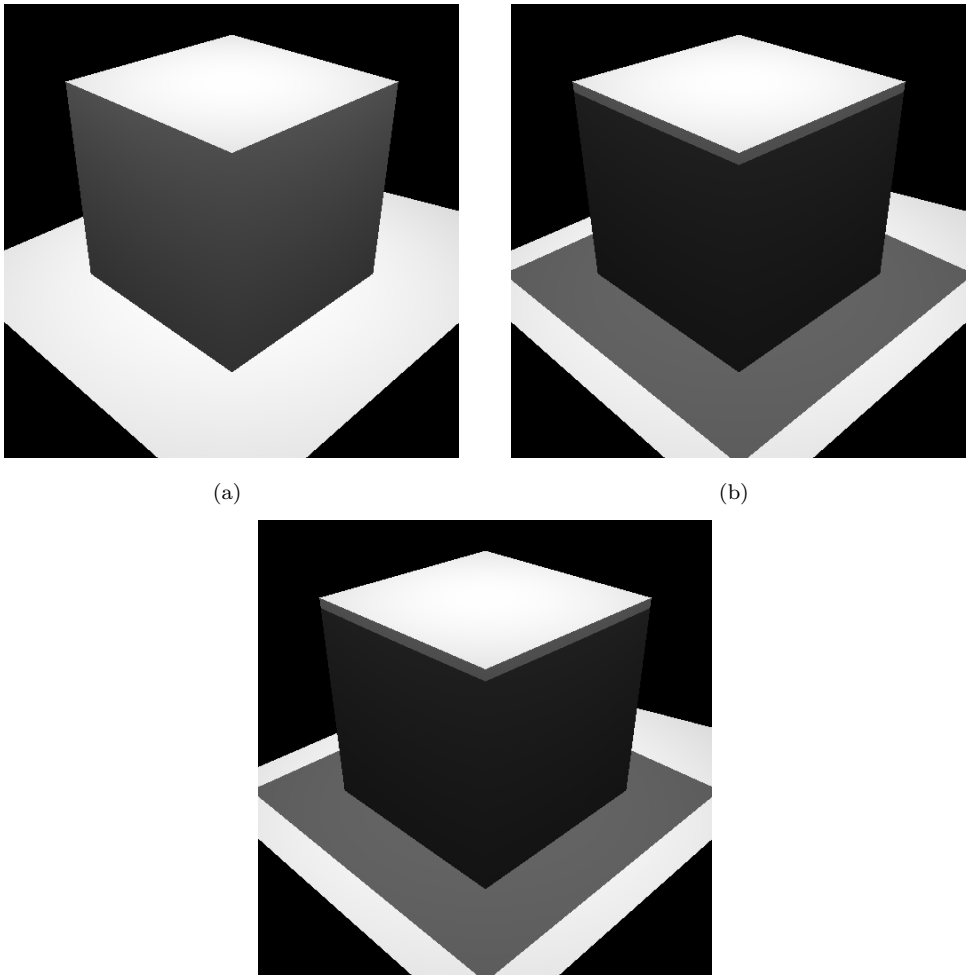
Figure 8: （a）room场景漫反射仿真结果图（b）room场景阴影仿真结果图（c）room场景高光仿真结果图



(a) (b)

(c)

Figure 9: （a）Veach MIS场景漫反射仿真结果图（b）Veach MIS场景阴影仿真结果图（c）Veach MIS场景高光仿真结果图



(a) (b)

(c)

Figure 10: （a）Veach MIS场景漫反射仿真结果图（b）Veach MIS场景阴影仿真结果图（c）Veach MIS场景高光仿真结果图

3.3 仿真结果总结

本文中涉及的三个场景，其中cup场景效果相对较好，但是整体场景的结果图整体偏暗有些参数有待调整。room场景中阴影效果较好，然而由于多余面片的存在因此桌面会有额外的阴影。三个场景中Veach MIS场景效果最差运行耗时很长，因此结果显示的是低分辨率的结果。Figure 10图片的仿真结果是自己设计的一个显示正方体场景，效果一般。

4 问题汇总

4.1 关于光线的疑问

问题描述：在编程时光线如何存储？光线的数学表达如下所示：

$$\boldsymbol{R}(t) = \boldsymbol{O} + t\boldsymbol{D}$$

其中$\boldsymbol{O}$为原点，$\boldsymbol{D}$为光线的方向（默认为单位向量），而t为距离。

4.2 光照信息的存储

4.3 蒙特卡洛用在哪里？

光线跟踪的交叉点的光照值通过RDBF模型来计算，RDBF模型中假设了采样点的光度值都是由来自各个方向的半球光线汇集得到的RGB值，如下图所示：

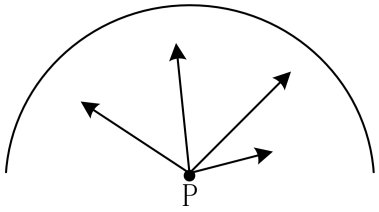


Figure 11: 蒙特卡洛采样示意图

由于来自该半球方向的光线理论上有无线多条，因此采用蒙特卡洛方法随机的对方向进行采样。蒙特卡洛的结果可以看成是该半球光线积分的估计值。

4.4 Kd-tree面对三角片如何构建？

由于kd-tree分割的对象是点坐标，因此对于三角片可以用其中点代替三角片本身的位置，这样就可以很方便的确定kd-tree中心轴的位置。

4.5 Kd-tree是如何与包围盒结合使用的？

kd-tree的每一个节点都包含一个包围盒，该包围盒包括了该节点下的所有子节点空间。