Progressive Photon Mapping

吴克文 梁家硕

2017 年 6 月

Section 1

概述

PPM vs PM

Photon Mapping 作为全局光照领域的主流算法，以其高效率，能处理多种光照效果等特点，一直受到广泛的关注。

PPM vs PM

Photon Mapping 作为全局光照领域的主流算法，以其高效率，能处理多种光照效果等特点，一直受到广泛的关注。然而，Photon Mapping 算法的一个主要问题在于，使用光子进行光能估计的过程引入了偏差。理论上，要完全消除偏差，需要存储无穷的光子，这从计算机存储角度来看是不可接受的。

PPM vs PM

Photon Mapping 作为全局光照领域的主流算法，以其高效率，能处理多种光照效果等特点，一直受到广泛的关注。
然而，Photon Mapping 算法的一个主要问题在于，使用光子进行光能估计的过程引入了偏差。理论上，要完全消除偏差，需要存储无穷的光子，这从计算机存储角度来看是不可接受的。
为此，Toshiya Hachisuka 提出了 Progressive Photon Mapping(又称渐进式光子映射)，采用多遍的绘制流程，通过不断向场景中发射光子达到不断减小偏差的目的，亦解决了 Photon Mapping 的存储问题。

What's new

- 支持 Linux，后推出 Windows 版本
- 使用扩展版 KD-Tree 维护场景
- 利用 KD-Tree 加速邻近点查找
- 多线程
- 同时支持 BRDF 参数和 Phong 方程
- 支持贴图
- 支持真球体

Section 2

效果展示

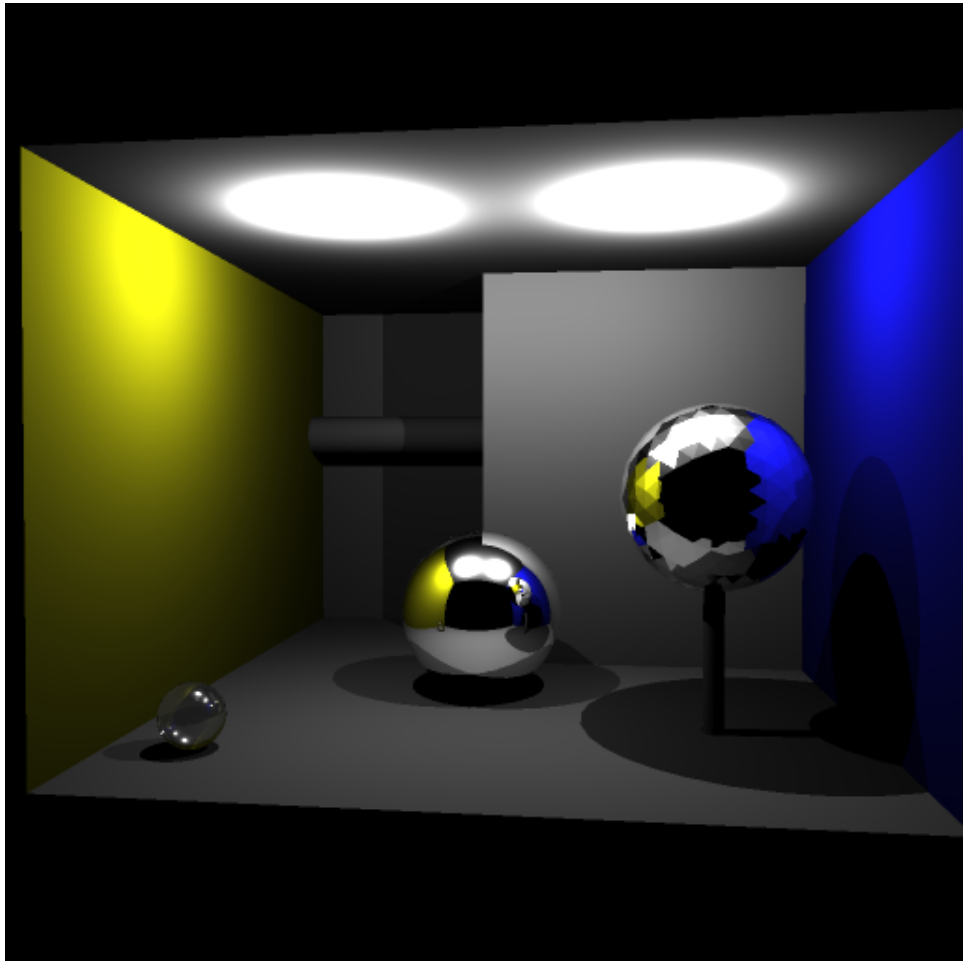


图: PPM 原作者的场景 0

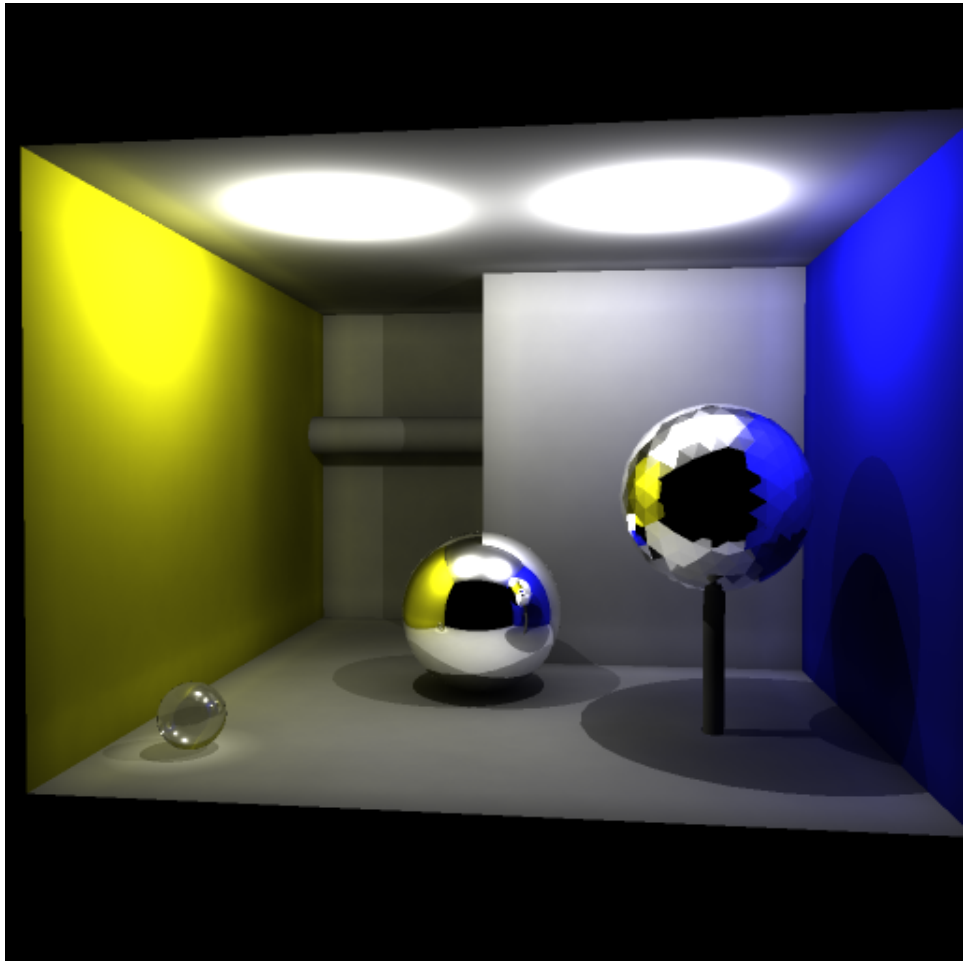


图: PPM 原作者的场景 1

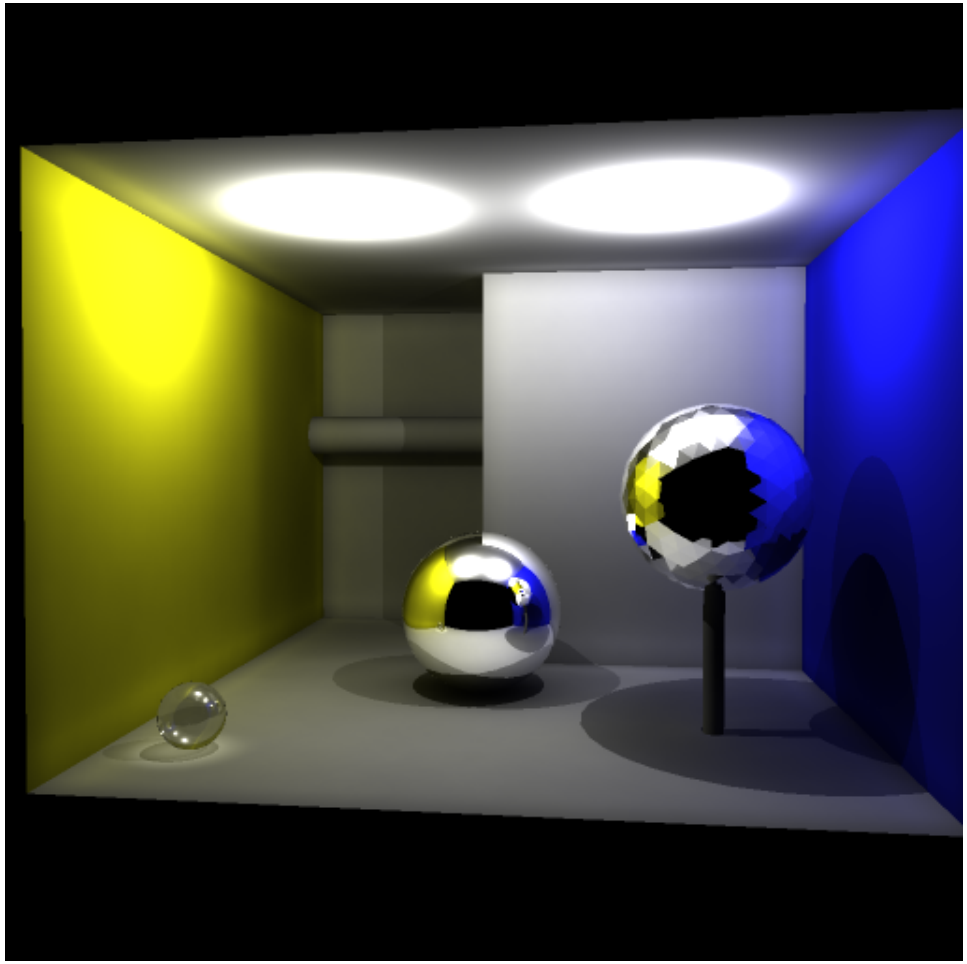


图: PPM 原作者的场景 5

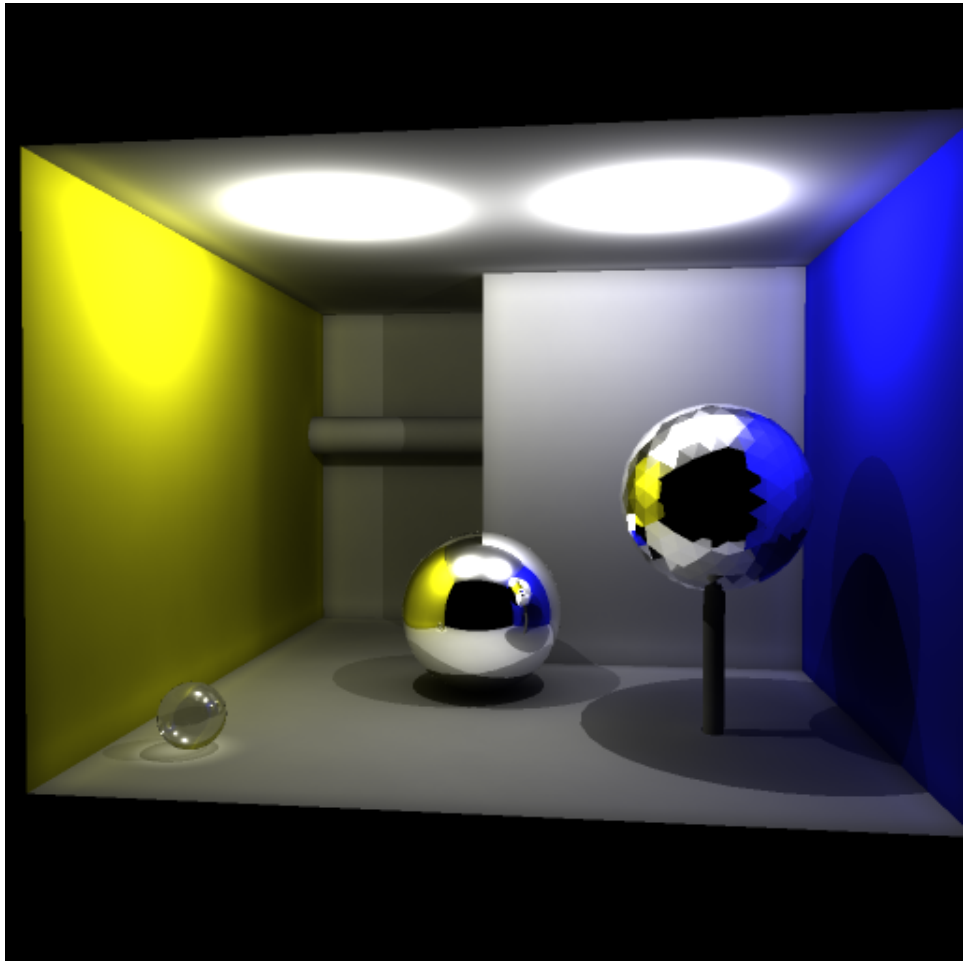


图: PPM 原作者的场景 10

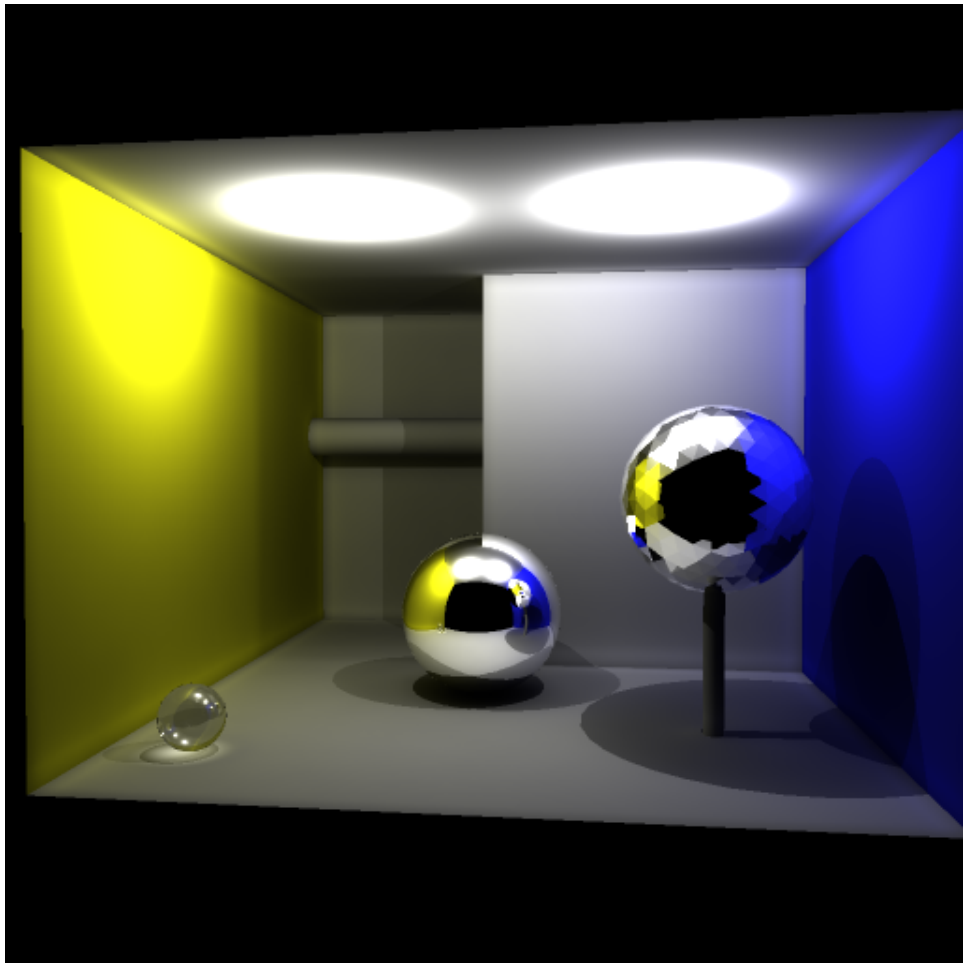


图: PPM 原作者的场景 100 (33'13")

图: 镜面球体 0

图: 镜面球体 1

图: 镜面球体 5

图: 镜面球体 10

图: 镜面球体 100 (6'56")

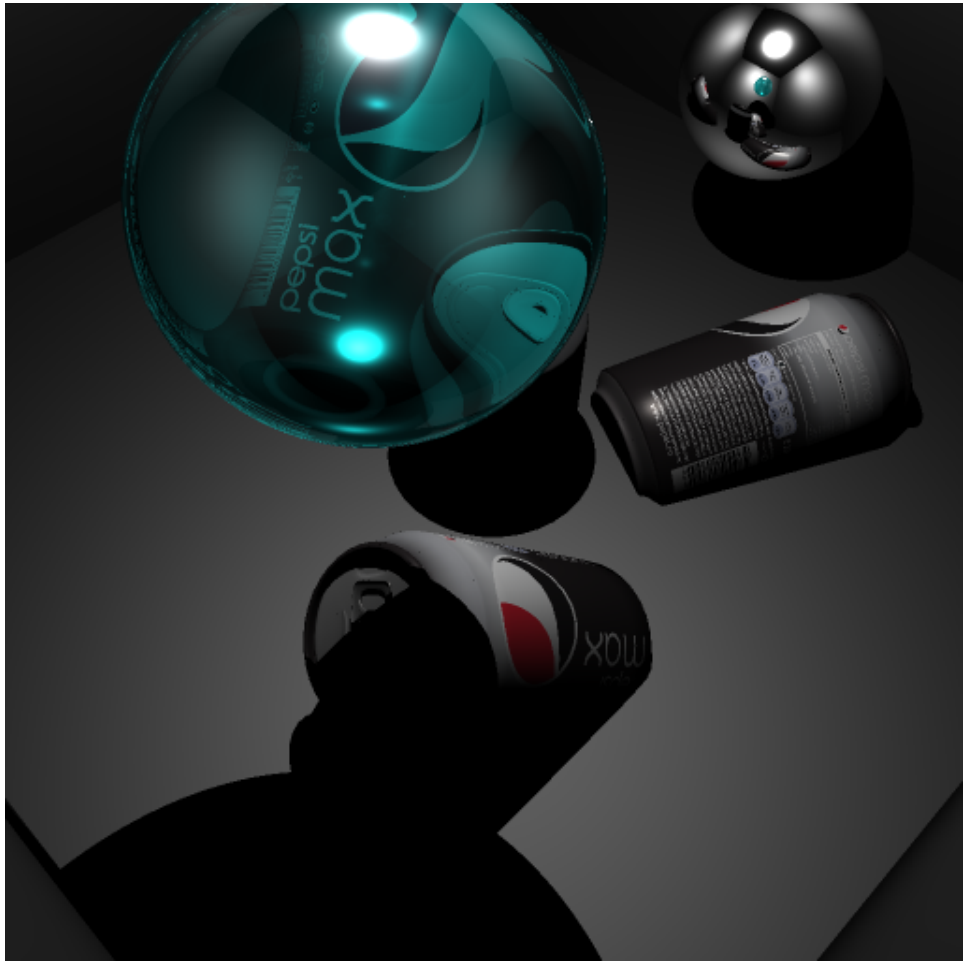


图: 透明球体 0

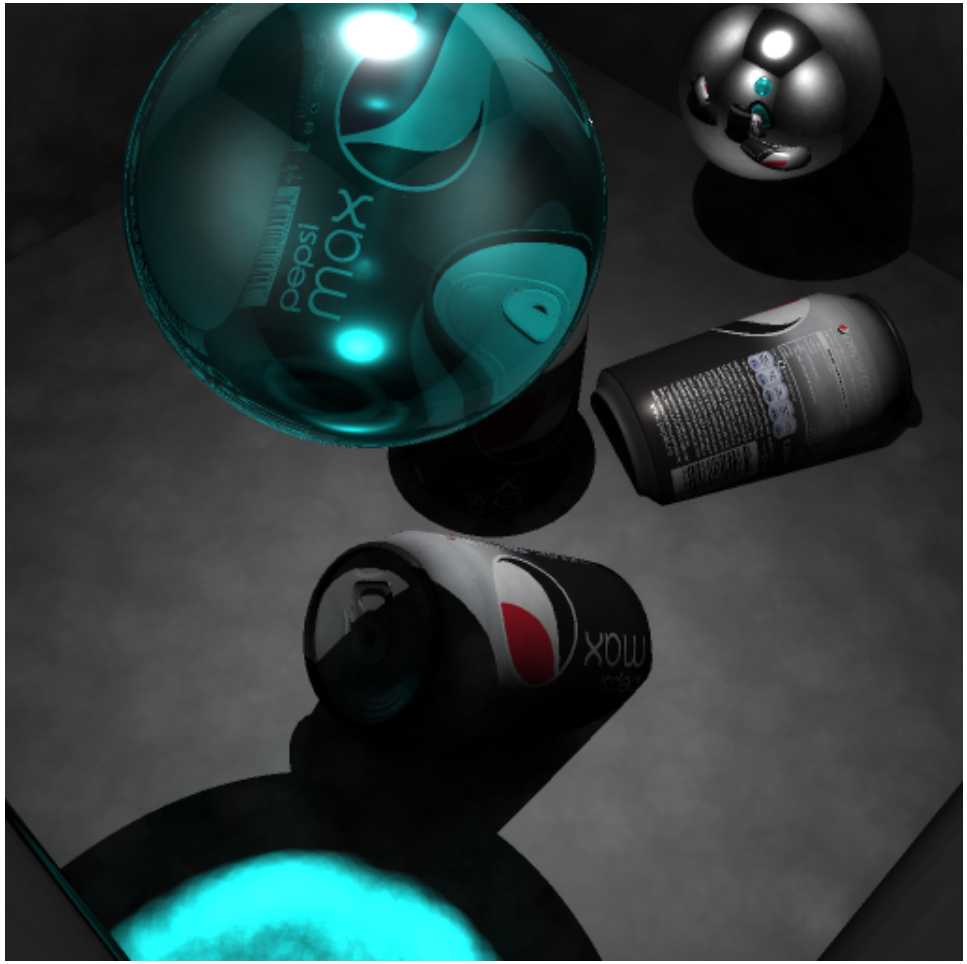


图: 透明球体 1

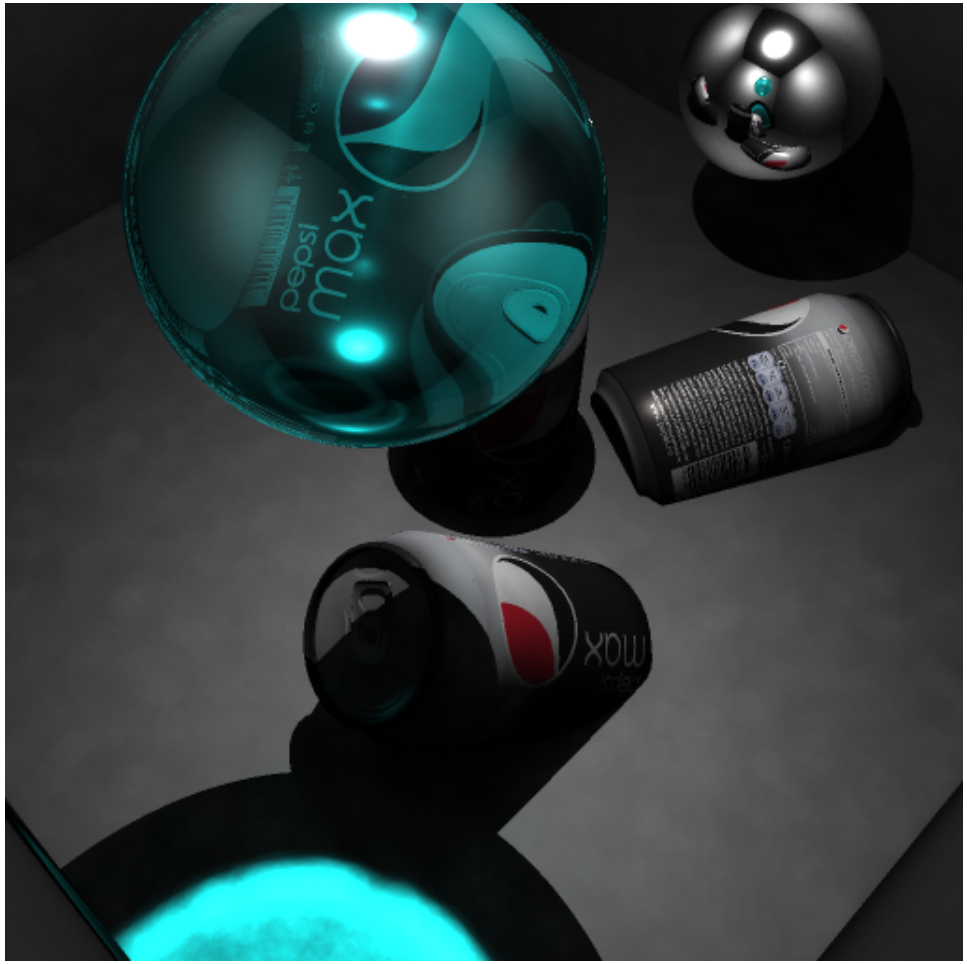


图: 透明球体 5

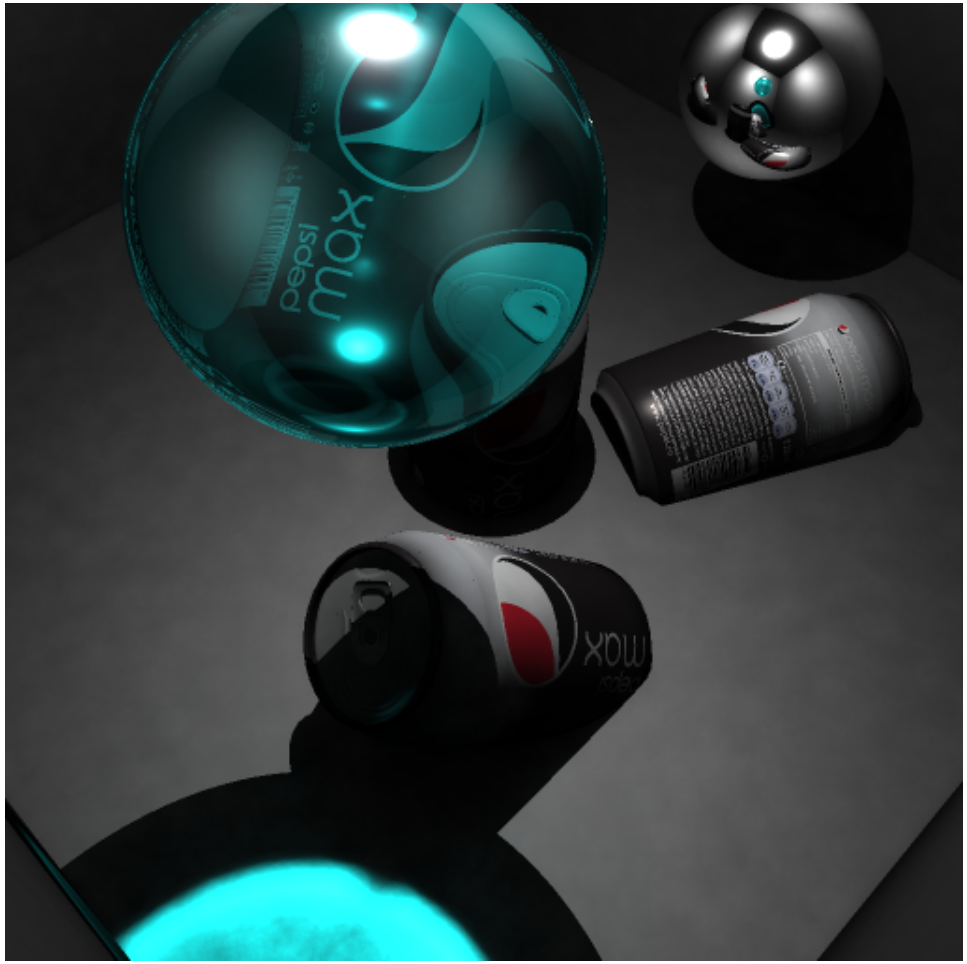


图: 透明球体 10

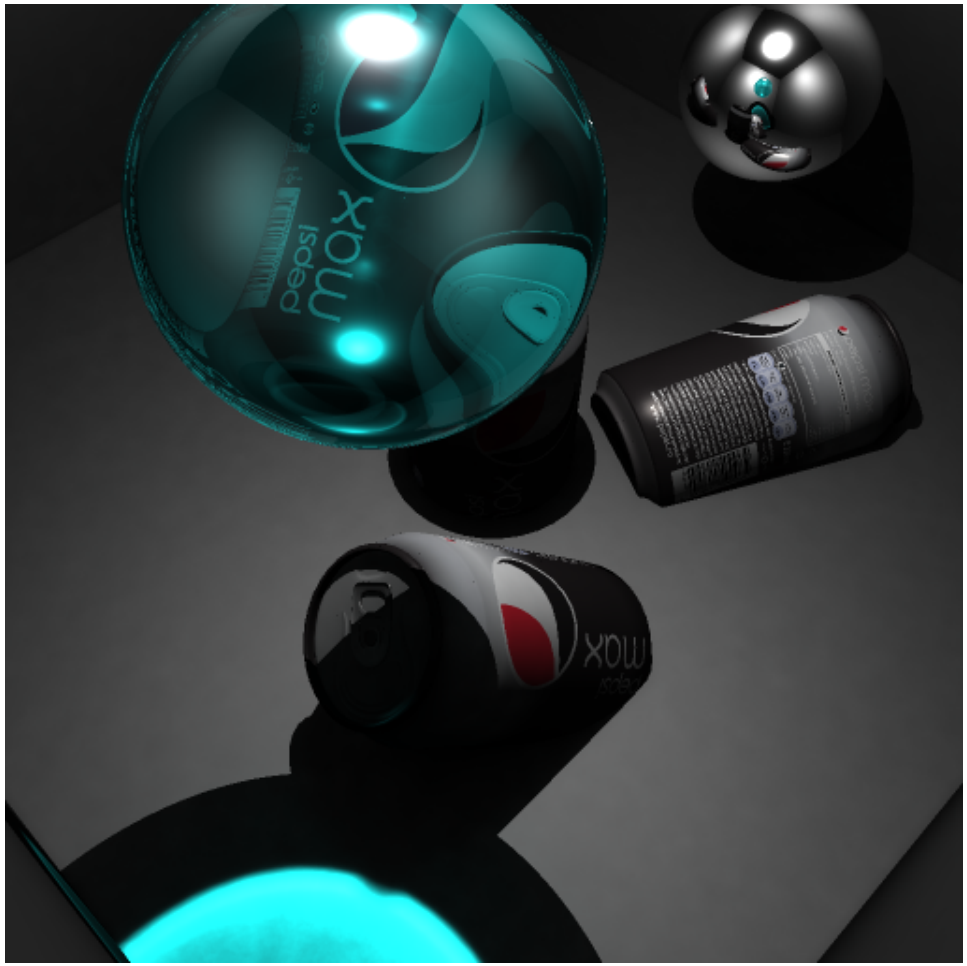


图: 透明球体 100 (9'13")

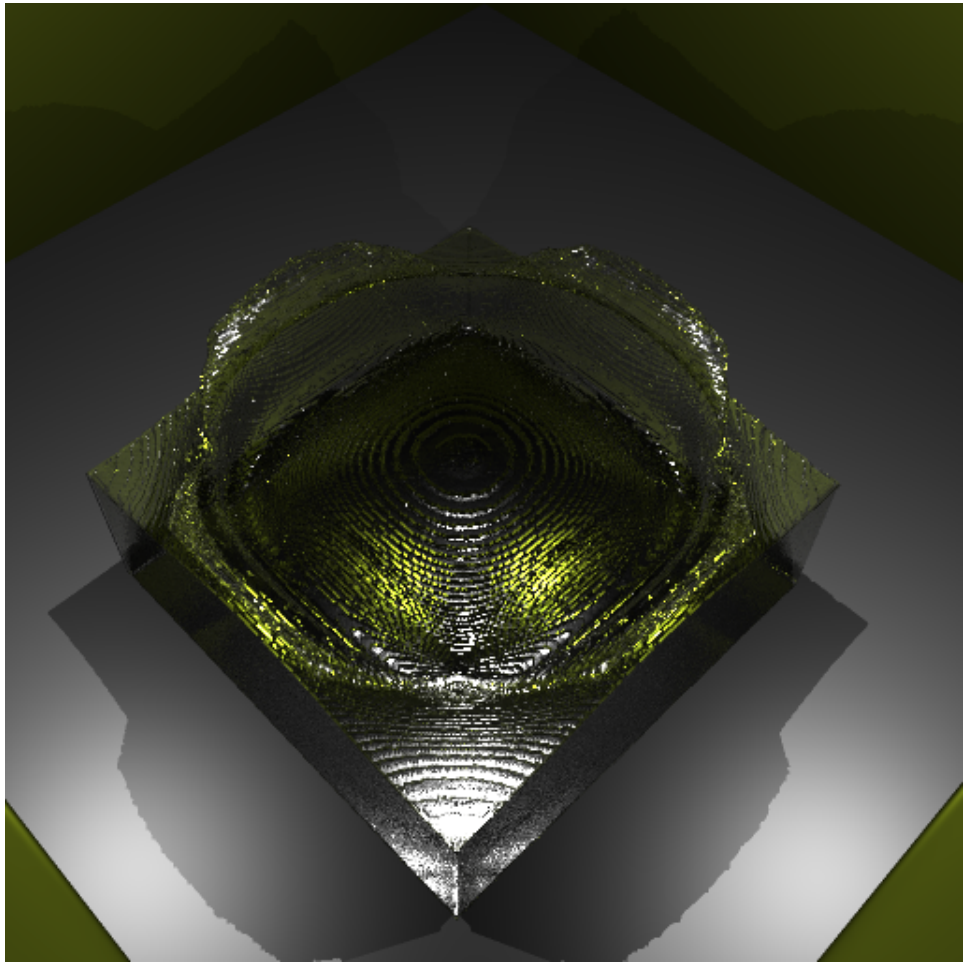


图: 水 0

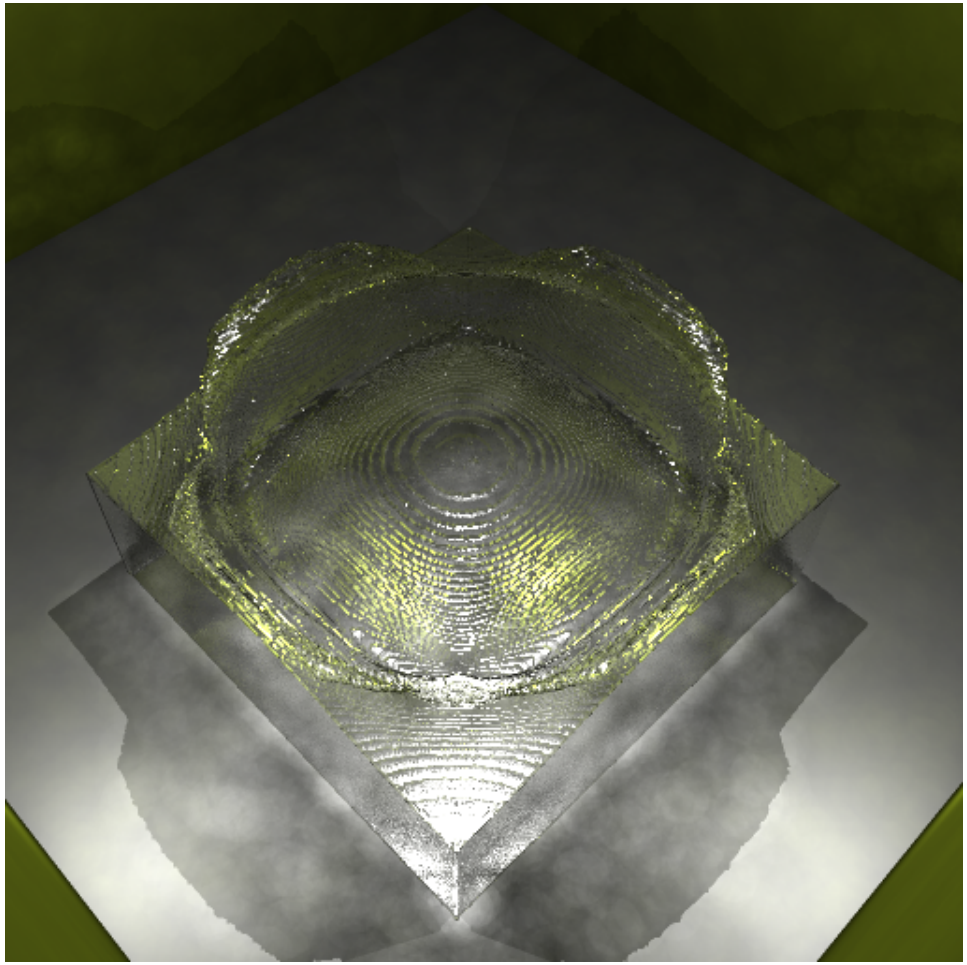


图: 水 1

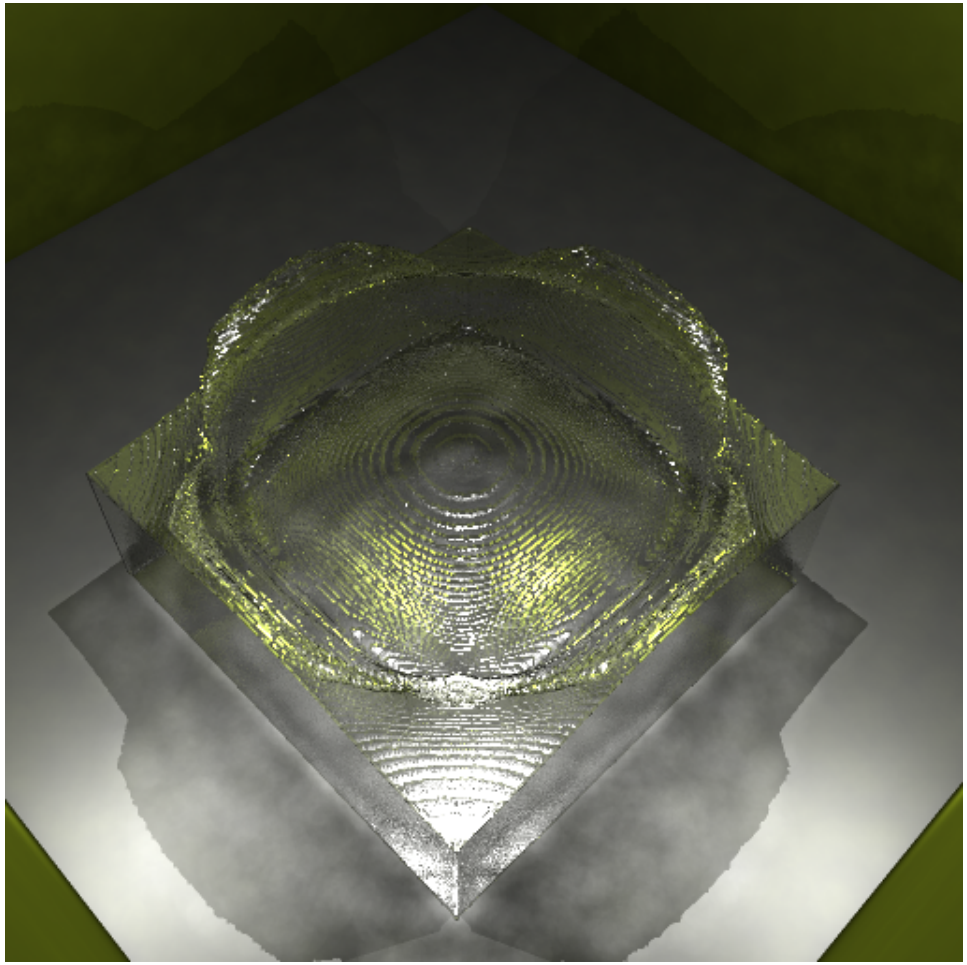


图: 水 5

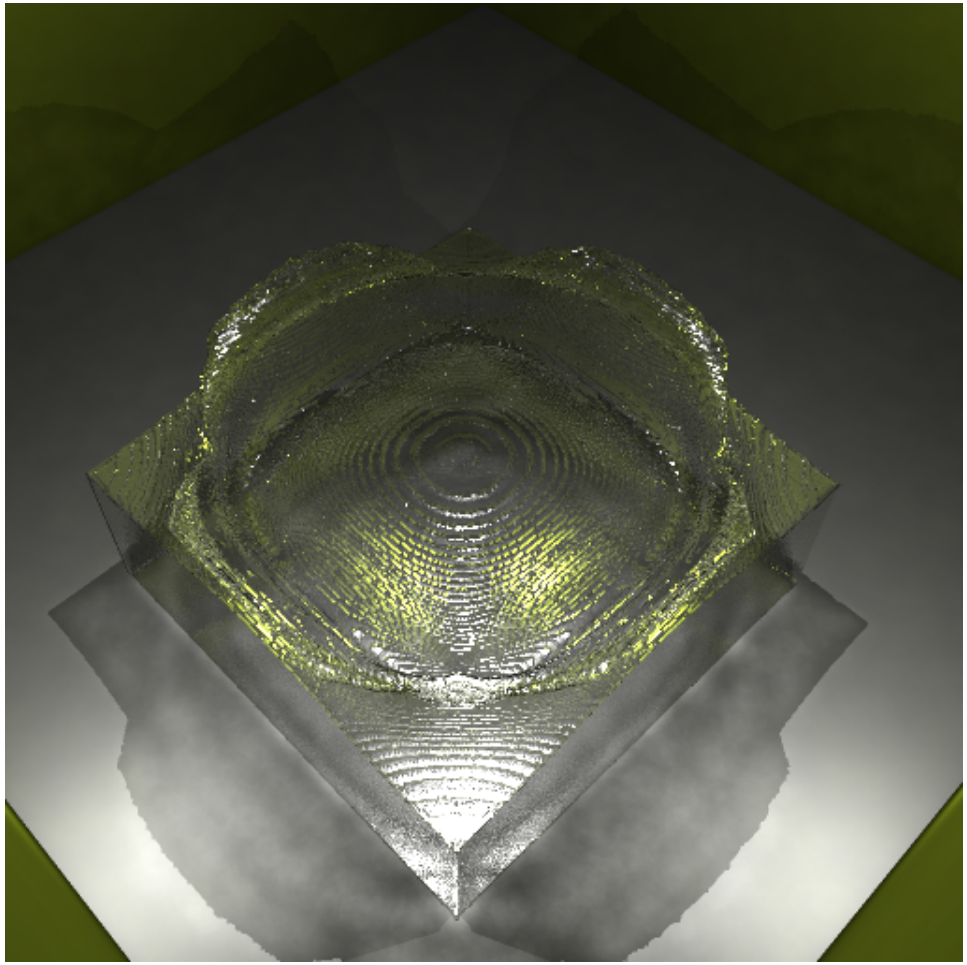


图: 水 10

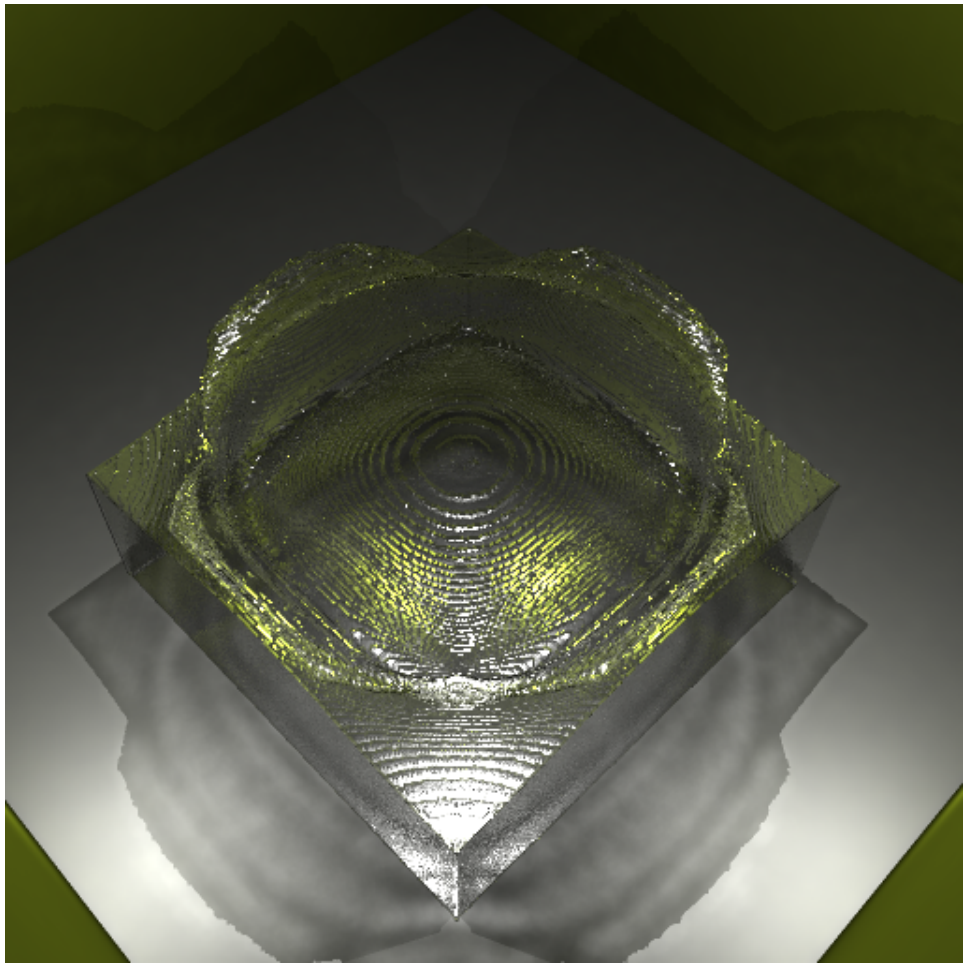


图: 水 100 (23'05")

Section 3

环境与使用

编译与运行环境

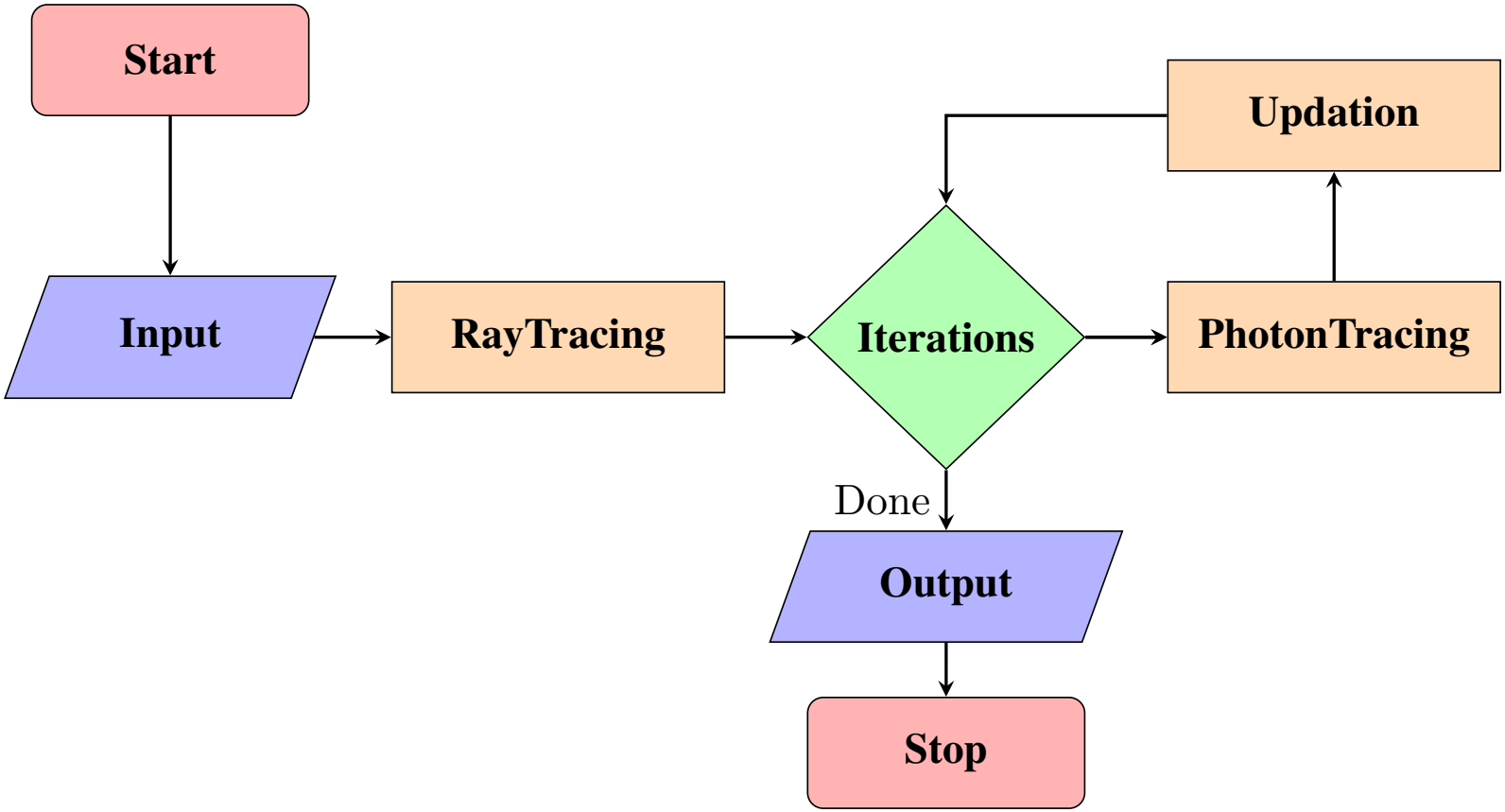
Linux/Windows
支持 c++11 的编译器
GNU toolchain
cmake >= 2.8

使用

```
$ raytracing <directory>
$ photontracing <directory>
$ updation <directory> <picture_name>
```

Section 4

总流程



Section 5

核心算法

Subsection 1

Lighting Equation

BRDF

BRDF(双向反射分布函数), 全称为 Bidirectional Reflectance Distribution Function，用来定义给定入射方向上的辐射照度如何影响给定出射方向上的辐射率。更笼统地说，它描述了入射光线经过某个表面反射后在各个出射方向上的分布效果。

光照方程

$$L_o(\mathbf{x}, \omega_o) = L_e(\mathbf{x}, \omega_o) + \int_\Omega BRDF(\mathbf{x}, \omega_i, \omega_o) L_i(\mathbf{x}, \omega_i)(\omega_i \cdot \mathbf{n})\, d\omega_i \quad (1)$$

其中，L_e 为直接光照，$\mathbf{x}$ 为空间坐标，$\mathbf{n}$ 为平面法向量，ω_o, ω_i 分别为出射和入射方向。

Subsection 2

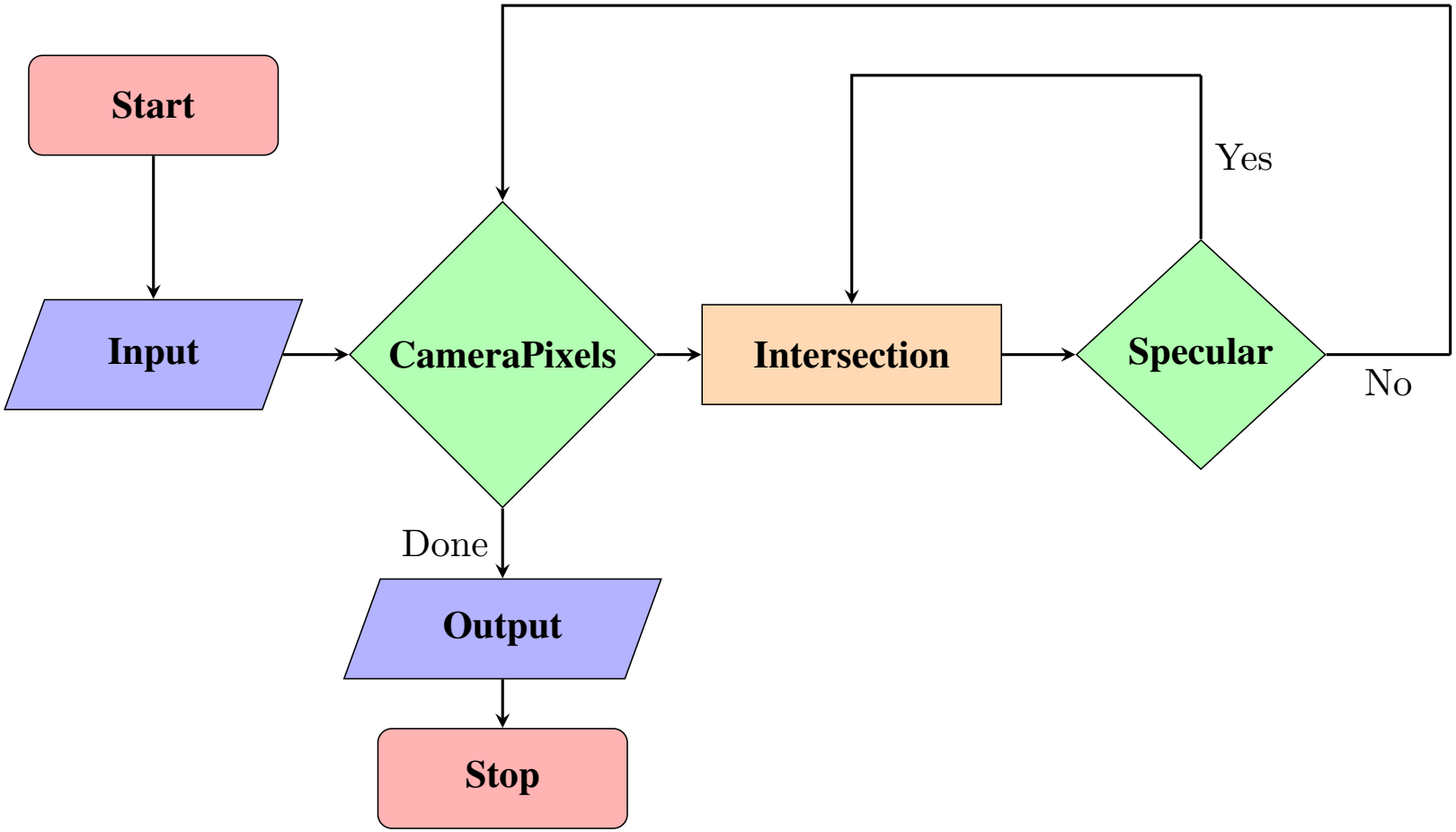
Ray Tracing Pass

光线追踪

从观察点出发，通过光线追踪来获得可见点 (hitpoints)，同时计算直接光照的贡献。

注

在镜面较多的场景中可用反（折）射次数作为阈值强制结束 Ray Tracing Pass。



Subsection 3

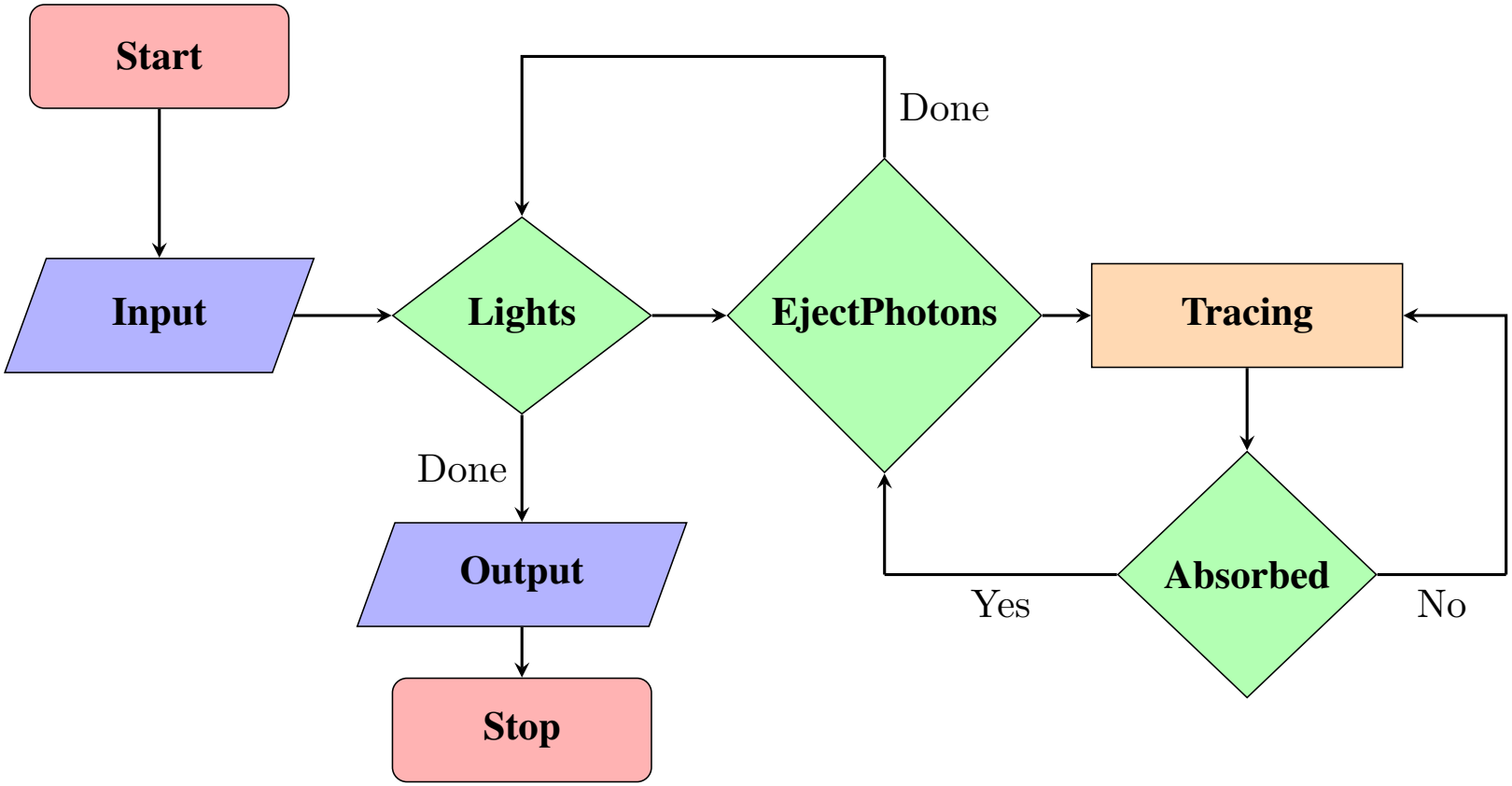
Photon Tracing Pass

光子追踪

每轮 Photon Tracing Pass，从光源随机方向发射一批光子，追踪每个光子的运动轨迹，考虑到效率，将光子能量的衰减用随机被物体表面吸收（或达到折反射阈值）来控制，这样每个光子的能量即为定值，折反射仅改变其颜色向量（通过 BRDF 计算）。

注

由于直接光源已在 Ray Tracing Pass 计算过，故每个光子与场景的第一个交点不必计入 photon map。



Subsection 4

Progressive Updation

更新模型

结束 Photon Tracing Pass 后，需要枚举每个 hitpoint，同时统计其半径 R 内光子对其亮度影响。

推导

记 $N(\mathbf{x})$ 为上轮后在 hitpoint $\mathbf{x}$ 半径 $R(\mathbf{x})$ 内的光子数，$M(\mathbf{x})$ 为本次新增光子数，同时 $\hat{N}(\mathbf{x}), \hat{R}(\mathbf{x})$ 分别为新累计光子数和半径，则有如下更新，

$$\hat{N}(\mathbf{x}) = N(\mathbf{x}) + \alpha M(\mathbf{x}) \tag{2}$$

$$\hat{R}(\mathbf{x}) = R(\mathbf{x}) \sqrt{\frac{N(\mathbf{x}) + \alpha M(\mathbf{x})}{N(\mathbf{x}) + M(\mathbf{x})}} \tag{3}$$

推导

记 $\tau_N(\mathbf{x},\omega)$ 和 $\tau_M(\mathbf{x},\omega)$ 为在 $\mathbf{x}$ 处，入射光方向为 ω 的前光强和新增光强（未乘 BRDF 系数），则有

$$\tau_{\hat{N}}(\mathbf{x},\omega) = (\tau_N(\mathbf{x},\omega) + \tau_M(\mathbf{x},\omega))\frac{N(\mathbf{x}) + \alpha M(\mathbf{x})}{N(\mathbf{x}) + M(\mathbf{x})} \tag{4}$$

其中，$\alpha \in (0,1)$ 是一常数。
再记总发射光子数为 $N_{emitted}$，ϕ 为光子光强，则最终辐照率表达式为，

$$L(\mathbf{x},\omega) \approx \frac{1}{\pi R(\mathbf{x})^2}\frac{\tau(\mathbf{x},\omega)}{N_{emitted}} \tag{5}$$


Toshiya Hachisuka, Shinji Ogaki, and Henrik Wann Jensen.
Progressive photon mapping.
ACM Transactions on Graphics (TOG), 27(5):130, 2008.


Henrik Wann Jensen.
Realistic image synthesis using photon mapping, volume 364.
Ak Peters Natick, 2001.


Ben Spencer and Mark W Jones.
Progressive photon relaxation.
ACM Transactions on Graphics (TOG), 32(1):7, 2013.


李睿, 陈彦云, and 刘学慧.
基于自适应光子发射的渐进式光子映射.
计算机工程与设计, 33(1):219–223, 2012.

其它参考

BRDF 参数及代码来自网站 http://www.merl.com/brdf/
png 图片相关代码来自 http://lodev.org/lodepng/

更多

更多技术细节详见 Equestrotopia.pdf 和 src 文件夹中的代码以及 GitHub 仓库
https://github.com/liangjs/Equestrotopia

Thanks!