作业 7：路径追踪
GAMES101, 2020 年春季

教授：闫令琪

计算机图形学与混合现实研讨会

GAMES：Graphics And Mixed Environment Seminar

发布日期为北京时间 2020 年 4 月 10 日 (星期六) 中午**12：00**

截止日期为北京时间 2020 年 4 月 21 日 (星期二) 晚上**23：59**

注意：

- 任何更新或更正都将发布在论坛上，因此请偶尔检查一下。
- 论坛链接：`http://games-cn.org/forums/forum/graphics-intro/`。
- 你必须独立完成自己的作业。
- 你可以在论坛上发布帖子求助，但是发布问题之前，请仔细阅读本文档。
- 在截止时间之前将你的作业提交到 SmartChair 上。

---

1 总览

在之前的练习中，我们实现了 Whitted-Style Ray Tracing 算法，并且用 BVH 等加速结构对于求交过程进行了加速。在本次实验中，我们将在上一次实验的基础上实现完整的 Path Tracing 算法。至此，我们已经来到了光线追踪版块的最后一节内容。

请认真阅读本文档，按照本文档指示的流程完成本次实验。

2 调通框架

2.1 修改的内容

相比上一次实验，本次实验对框架的修改较大，主要在以下几方面：

- 修改了 main.cpp，以适应本次实验的测试模型 CornellBox
- 修改了 Render，以适应 CornellBox 并且支持 Path Tracing 需要的同一 Pixel 多次 Sample
- 修改了 Object，Sphere，Triangle，TriangleMesh，BVH，添加了 area 属性与 Sample 方法，以实现对光源按面积采样，并在 Scene 中添加了采样光源的接口 sampleLight
- 修改了 Material 并在其中实现了 sample, eval, pdf 三个方法用于 Path Tracing 变量的辅助计算

2.2 你需要迁移的内容

你需要从上一次编程练习中直接拷贝以下函数到对应位置：

- ~~`Triangle::getIntersection` in Triangle.hpp: 将你的光线-三角形相交函数粘贴到此处，请直接将上次实验中实现的内容粘贴在此。~~
- ~~`IntersectP(const Ray& ray, const Vector3f& invDir, const std::array<int, 3>& dirIsNeg)`~~ in the Bounds3.hpp: 这个函数的

作用是判断包围盒 BoundingBox 与光线是否相交，请直接将上次实验中实现的内容粘贴在此处，并且注意检查 t_enter = t_exit 的时候的判断是否正确。

- **~~getIntersection~~(BVHBuildNode* node, const Ray ray)**in BVH.cpp: BVH 查找过程，请直接将上次实验中实现的内容粘贴在此处.

2.3 编译运行

基础代码只依赖于 CMake，下载基础代码后，执行下列命令，就可以编译这个项目：

```
$ mkdir build
$ cd ./build
$ cmake ..
$ make
```

在此之后，你就可以通过 **./Raytracing** 来执行程序。请务必确保程序可以正常编译之后，再进入下一节的内容。

3 开始实现

3.1 代码框架

在本次实验中，你只需要修改这一个函数:

- **castRay(const Ray ray, int depth)**in Scene.cpp: 在其中实现 Path Tracing 算法

可能用到的函数有：

- **intersect(const Ray ray)**in Scene.cpp: 求一条光线与场景的交点
- **sampleLight(Intersection pos, float pdf)** in Scene.cpp: 在场景的所有光源上按面积 uniform 地 sample 一个点，并计算该 sample 的概率密度

- **`sample(const Vector3f wi, const Vector3f N)`** in Material.cpp: 按照该材质的性质，给定入射方向与法向量，用某种分布采样一个出射方向
- **`pdf(const Vector3f wi, const Vector3f wo, const Vector3f N)`** in Material.cpp: 给定一对入射、出射方向与法向量，计算 sample 方法得到该出射方向的概率密度
- **`eval(const Vector3f wi, const Vector3f wo, const Vector3f N)`** in Material.cpp: 给定一对入射、出射方向与法向量，计算这种情况下的 f_r 值

可能用到的变量有：

- **RussianRoulette** in Scene.cpp: P_RR, Russian Roulette 的概率

3.2 Path Tracing 的实现说明

课程中介绍的 Path Tracing 伪代码如下 (为了与之前框架保持一致，wo 定义与课程介绍相反)：

```
shade(p, wo)
  Uniformly sample the light at xx (pdf_light = 1 / A)
  Shoot a ray from p to x
  If the ray is not blocked in the middle
    L_dir = L_i * f_r * cos_theta * cos_theta_x / |x-p|^2
  / pdf_light

  L_indir = 0.0
  Test Russian Roulette with probability P_RR
  Uniformly sample the hemisphere toward wi (pdf_hemi = 1
  / 2pi)
  Trace a ray r(p, wi)
  If ray r hit a non-emitting object at q
```

```
    L_indir = shade(q, wi) * f_r * cos_theta / pdf_hemi / P_RR

  Return L_dir + L_indir
```

按照本次实验给出的框架，我们进一步可以将伪代码改写为：

```
shade(p, wo)
  sampleLight(inter, pdf_light)
  Get x, ws, NN, emit from inter
  Shoot a ray from p to x
  If the ray is not blocked in the middle
    L_dir = emit * eval(wo, ws, N) * dot(ws, N) * dot(ws, NN) / |x-p|^2 / pdf_light

  L_indir = 0.0
  Test Russian Roulette with probability RussianRoulette
  wi = sample(wo, N)
  Trace a ray r(p, wi)
  If ray r hit a non-emitting object at q
    L_indir = shade(q, wi) * eval(wo, wi, N) * dot(wi, N) / pdf(wo, wi, N) / RussianRoulette

  Return L_dir + L_indir
```

请确保你已经清晰地理解 Path Tracing 的实现方式，再进入下一个环节的讨论。

4 结果与分析

本章节讨论得到结果与调试过程中需要特别注意的一些问题。

4.1 注意事项

1. 本次实验代码的运行非常慢，建议调试时调整 main.cpp 中的场景大小或 Render.cpp 中的 SPP 数以加快运行速度；此外，还可以实现多线程来进一步加快运算。

2. 注意数值精度问题，尤其注意 pdf 接近零的情况，以及 sampleLight 时判断光线是否被挡的边界情况。这些情况往往会造成渲染结果噪点过多，或出现黑色横向条纹。

4.2 参考结果

如果严格按照上述算法实现，你会发现渲染结果中光源区域为纯黑。请分析这一现象的原因，并且修改 Path Tracing 算法使光源可见。最终结果如下：

4.3 材质的拓展

目前的框架中拆分 sample, eval, pdf, 实现了最基础的 Diffuse 材质。请在不破坏这三个函数定义方式的情况下修改这三个函数，实现 Microfacet 模型。本任务不要求你实现复杂的采样手段，因此你依然可以沿用 Diffuse 材质采用的 sample 与 pdf 计算。

Microfacet 相关知识见第十七讲 Slides `https://sites.cs.ucsb.edu/~lingqi/teaching/resources/GAMES101_Lecture_17.pdf`.

5 提交与评分

评分：

- [5 points] 提交格式正确，包含所有需要的文件；代码可以在虚拟机下正确编译运行。
- [45 points] Path Tracing：正确实现 Path Tracing 算法，并提交分辨率不小于 512*512，采样数不小于 8 的渲染结果图片。
- [加分项 10 points] 多线程：将多线程应用在 Ray Generation 上，注意实现时可能涉及的冲突。
- [加分项 10 points] Microfacet：正确实现 Microfacet 材质，并提交可体现 Microfacet 性质的渲染结果。
- [-5 points] 未提交 README.md，README.md 中未说明需要说明的信息，未提交 CMakeLists.txt，未提交结果图片，未完整提交代码，提交包中多余文件（比如 /build，/.vs）未清除。

提交：

- 当你完成作业后，请清理你的项目，记得在你的文件夹中包含 CMakeLists.txt 和所有的程序文件（无论是否修改）；

- 同时，请新建一个 /images 目录，将所有实验结果图片保存在该目录下；
- 再添加一个 README.md 文件写清楚自己完成了上述得分点中的哪几点，并说明提交结果的分辨率与采样数、计算时间；如果实现了扩展功能，你还需要简要描述你在各个函数中实现的功能；
- 最后,将上述内容打包,并用“姓名_Homework7.zip”的命名方式提交到 SmartChair 平台。