光线追踪大作业 实验报告

叶舟桐 2020010940

一、实现功能总览

算法选型：路径追踪（Path Tracing, PT）

实现的功能：反射、折射、阴影；B 样条曲线旋转面；三角网格模型导入、法向插值；软阴影（PT 自带）、景深、运动模糊；材质贴图、法向扰动凹凸贴图；色散；焦散（PT 自带）；Color Bleeding（PT 自带）；kd-tree 加速、OpenMP 加速；抗锯齿；参与性介质（体积光）。

二、算法主体

本次大作业采用的算法为路径追踪（Path Tracing）。路径追踪通过在同一像素处多次采样，采样时将漫反射表面当作 Lambert 反射体处理，进行随机得到反射光线方向。每次采样终止于两种情况：或是击中光源，或是被俄罗斯轮盘赌（Russian Roulette）策略终止。后者的终止概率与漫反射体对光线的吸收率相关：吸收率越高，终止概率越大。最终通过将多次采样的结果进行平均，得到此像素的 RGB 值，相当于用蒙特卡洛法近似求解亮度积分。

路径追踪算法的采样过程符合物理规律，故其收敛结果为真实感很强的图像，属于无偏的渲染方法。因此，除了基础的反射、折射和阴影之外，路径追踪还自带了软阴影、焦散和 Color Bleeding 的效果。

本次作业中路径追踪的主体逻辑在 src/main.cpp 的 radiance()

函数中实现。处理折射体表面的反射现象时采用了菲涅尔项的Schlick 近似。

三、参数曲面

本次作业中实现了 B 样条曲线的旋转面的导入。求交时采用了牛顿迭代法，具体实现参考了教材 107 页的推导。实践中发现牛顿迭代法求得的交点不一定是该曲面离光源最近（t 最小）的交点。通过采用对迭代起始点进行多次采样并取最近的交点，这一问题得到了解决。代码在 include/curve.hpp 及 include/revsurface.hpp 中。

四、三角网格与法向插值

本次作业中实现了三角网格模型的导入和基于法向插值的平滑处理。法向插值的具体方法如下：对于每一个顶点，定义其顶点法向量为其所在的所有三角面片的法向量之和的单位化结果；对于光线与三角面片的交点，其法向量为交点所在三角面片的顶点法向量根据顶点与交点距离倒数的加权和的单位化结果。三角网格和法向插值的实现详见 src/mesh.cpp 和 include/triangle.hpp。具体实现时为了减少浮点数误差，采用了连续 2 次在直线上插值的方法来计算交点在三角面片上的插值，具体推导见 triangle.hpp 中的注释。

五、景深与运动模糊

景深的实现参考了相机和人眼的成像原理，在采样时模拟了光圈和焦距的效果。具体而言，在一次采样的光线生成以后，首先算出该光线在与相机距离为焦距长度处的点 P。之后根据光圈大小，对

光线的原点进行扰动。以从新原点出发、经过 P 点的光线采样，并将结果计入原像素（而非新原点对应的像素位置）。这一功能在 src/main.cpp 的 shift()函数中实现。

运动模糊通过在求交时随机物体在运动轨迹上的位置实现。具体方法是：记录带有运动模糊效果的物体在时刻 0 到时刻 1 之间的位移。在生成光线进行采样时，给光线赋予一个 0 到 1 之间的随机数作为光线的“时间”属性。求交时根据时间和总位移算出物体当前位置，并据此求交。相关功能在 src/main.cpp 中的 main()函数、include/ray.hpp 及 include/sphere.hpp 中实现。

六、纹理贴图及凹凸贴图

贴图通过将交点坐标转化为参数坐标实现。凹凸贴图的实现采用了法向扰动：额外导入纹理图片对应的 normal map 图片，在求交时以 normal map 上的值变换到场景坐标后（normal map 图片一般将参数坐标下的法向量作为该点的颜色值）作为交点处的法向量。法向扰动可以使被贴在平面上的纹理体现出凹凸效果（尤其是在垂直观察时）。具体代码在 include/texture.hpp、include/plane.hpp 的 Rectangle 类及 include/sphere.hpp 中。

七、色散

色散的物理本质是同一材质对不同波长光线折射率不同。因此实现色散需要在对光线原点、光线时间采样的同时也对光线的波长进行采样。一种采样的方式是对 RGB 三通道分别采样并将结果融合，但这种方法的效果生硬，色散区域明显为的红、绿、蓝三个色斑的

相交结果，真实感欠缺。

为了真正实现真实的色散，我在采样时考虑了太阳光谱可见光部分的波长分布，并在生成采样光线时近似于此分布在 380nm-750nm 的波长区间（即可见光波段）进行了采样。在遇到折射面时，我采用了柯西等式(Cauchy's transmission equation)计算当前光线波长对应的折射率。如果本次光线追踪终止于光源，那么在最终计算采样结果时需要将得到的亮度 RGB 值和采样波长的 RGB 值进行向量的按位乘，之后再计入该像素的采样结果。波长到 RGB 的换算关系采用了基于 gamma 值的换算方法。

在实际渲染中，此方法产生的色散现象表现为连续的光谱而非三个色斑的叠加，效果非常自然。不过由于在随机采样过程加入了波长这一维度，此方法对采样率的要求较高。在低采样率时往往会出现彩色噪点，影响观感。具体代码见 src/main.cpp 的 radiance()函数和 main()函数，以及 include/wavelength.hpp。

八、加速

为了节省时间成本，本次作业采用了 kd-tree 来存储网格面片，极大地减少了求交用时（渲染同一张测试图片，不做任何求交优化需要 130 秒，用包围盒+包围球需要 75 秒，用 kd-tree 需要 40 秒）。在具体实现中，kd-tree 节点将管辖区域（一个包围盒）分为三块：在包围盒内、在包围盒上（与边界相交）及在包围盒外。包围盒内的部分用于建立左子树，包围盒外的部分用于建立右子树，而包围和上的部分在搜索时用线性查找的方式遍历。建树时采用多

次随机+估值函数的方法选取最优分隔面。kd-tree 的实现见 include/kdtree.hpp。实现 kd-tree 需要对场景进行预处理，相关代码见 include/group.hpp。

除了 kd-tree 之外，本次作业也使用了 OpenMP 进行硬件加速，实现了对 CPU 的充分利用，也显著地加速了渲染。

九、抗锯齿

为了实现抗锯齿并减少噪点，我在给每一个像素采样时并没有只以像素中心为样本光线的起点，而是在像素中心周围的一个区域内进行采样。具体而言，我以像素中心为基准，利用一个 tent filter 进行采样达到了较好的抗锯齿效果。具体的代码实现见 src/main.cpp 的 main()函数。

十、参与性介质（体积光）

PT 直接实现体积光比较困难，但是可以一种比较简单的方法进行近似处理。体积光的本质是光线与参与性介质（participating media）的交互。参与性介质包括雾、霾、灰尘等，对光线有散射及吸收的作用。通过在路径追踪时模拟参与性介质对光线传播的影响，我实现了体积光的部分效果。

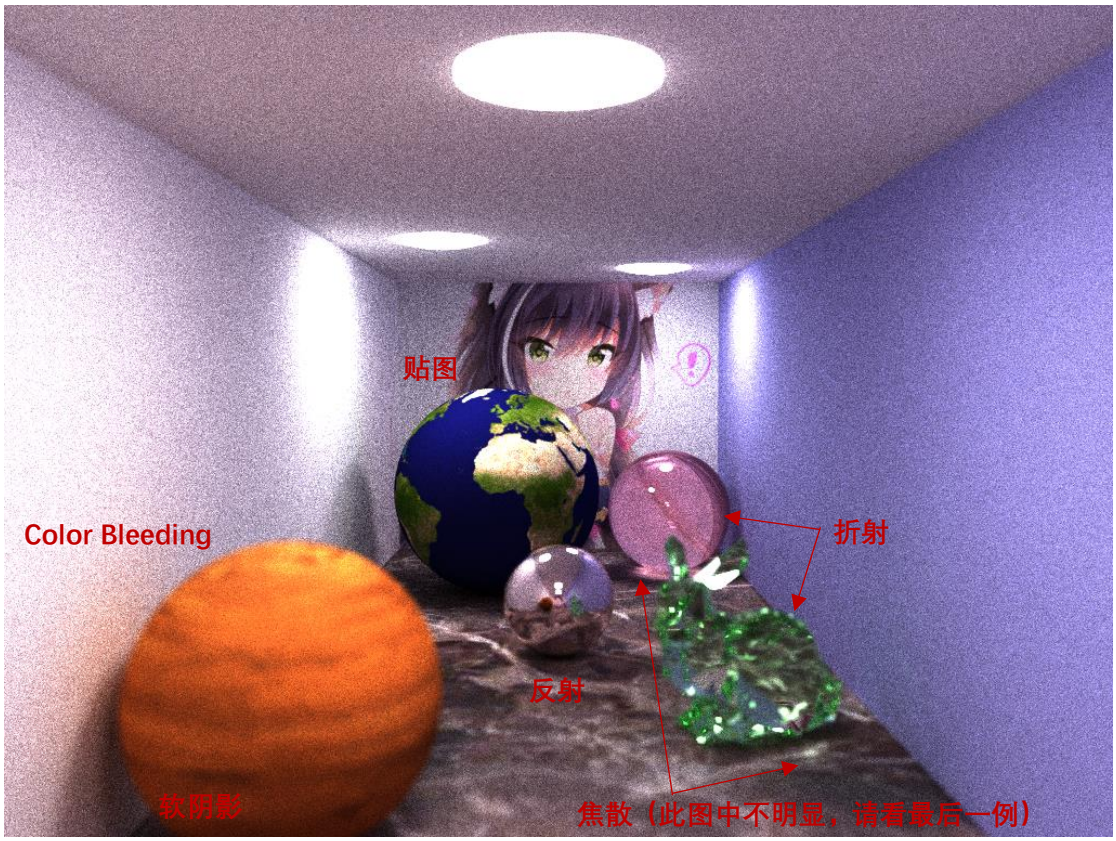
具体而言，我引入了三个参数：单位长度介质中光线不被吸收而通过的概率 P_{ne}、光线通过单位长度介质而不发生散射的概率 P_{ns}、光线发生散射时“反向散射”（back scattering）的强度 B。radiance()函数中算出击中点后，可以算出当前光线从起点到击中点传播距离 t 的过程中，被吸收的概率为 $1-P_{ne}^{t}$，被散射的概率

$1-P_{ns}^{t}$。如果随机到被吸收的情况则返回黑色，如果随机到被散射的情况则根据散射点的分布函数进行采样，取到一个散射点。在散射点，以 B*(-ray.getDirection())加上一个随机值，然后进行单位化的结果为散射方向，进行下一次光线追踪。这一方法是对散射相关的物理规律的近似，通过较多的采样次数可以更复杂的 in-scattering 和 out-scattering 现象。但由于在采样时引入了散射这一维度的随机性，对采样数要求较高。

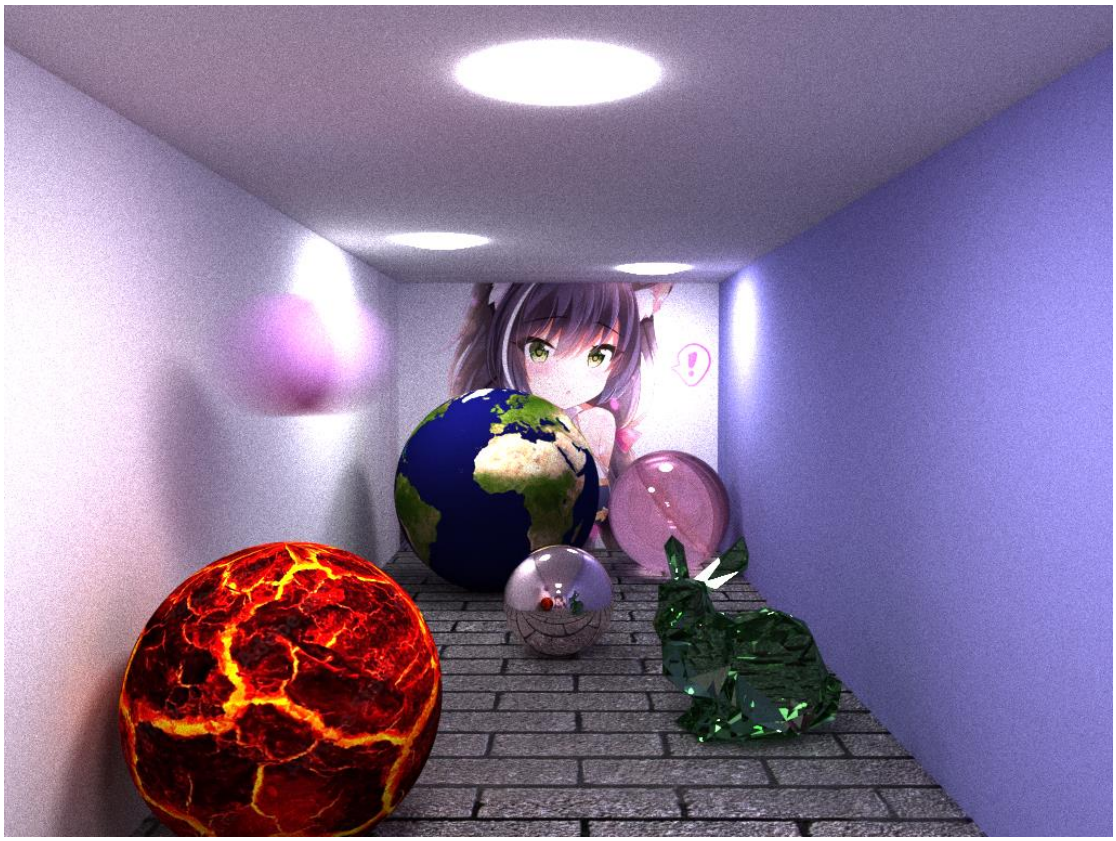
利用此方法，在实际渲染中观察到了参与性介质中出现了明显的丁达尔效应（Tyndall effect），调整好介质参数后可以得到体积光的效果。代码见 src/main.cpp 的 radiance()函数。

十一、渲染结果展示

基本效果（带景深，对焦在地球上，无注释原图见压缩包）



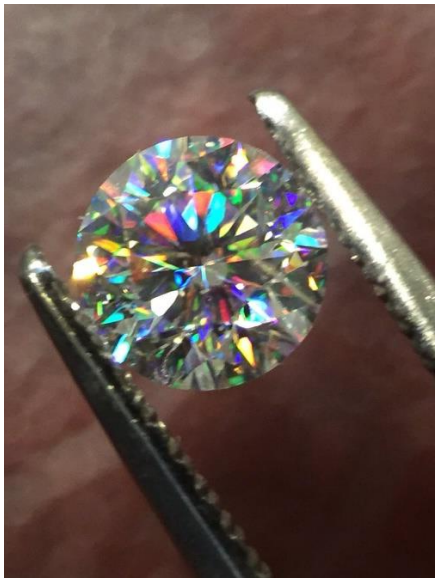
运动模糊（紫球）



法向插值（黑兔）、参数曲面（酒杯）与凹凸贴图（砖墙、木地板）

色散、焦散

这张图片中兔子使用的是普通玻璃的折射数据，兔子脚下有明显的焦散现象。可以和下面使用钻石数据的图片进行对比。（两张图光源颜色和位置不同）



用于对比的真实钻石照片

（非渲染所得!!）

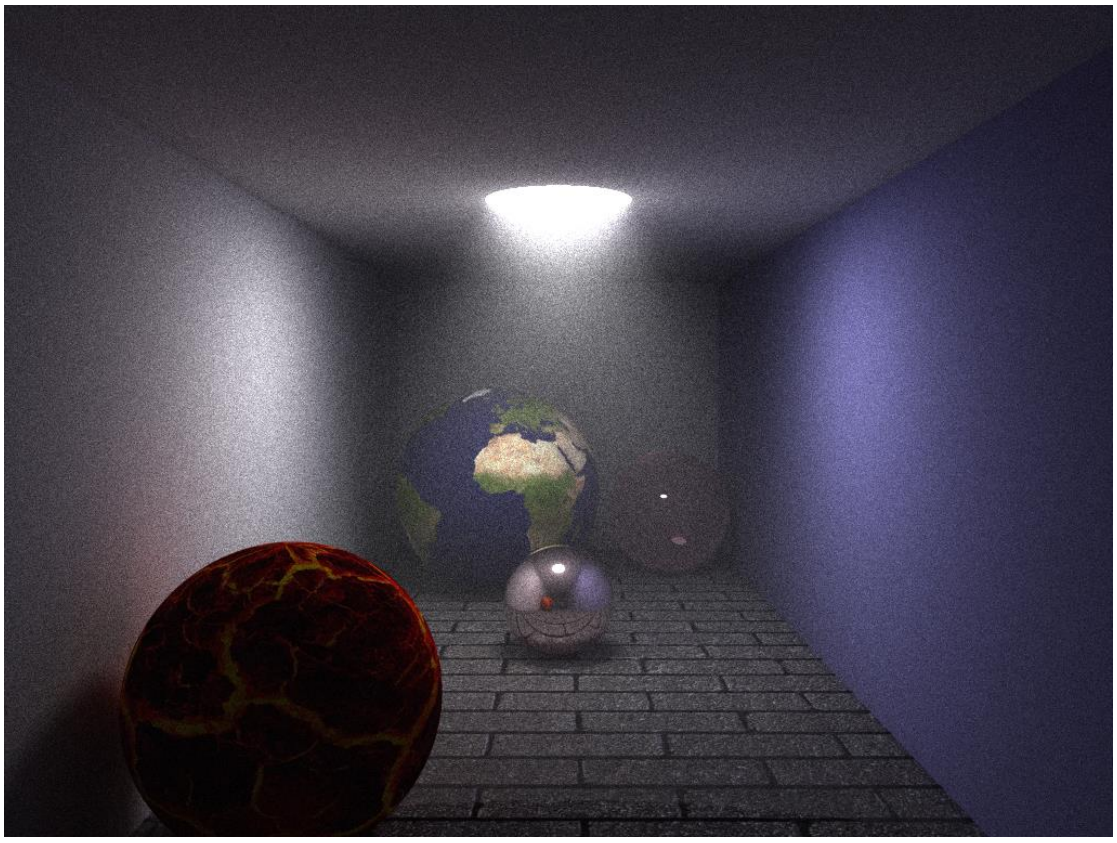
这张图中兔子的材质使用的是钻石的折射数据，经过折射和色散也确实呈现出了钻石“火彩”（fire）的效果。维基百科关于火彩的介绍如下

There are three light properties which are described in relation to light performance: brilliance, fire, and scintillation… Fire refers to the spectral colors which are produced as a result of the diamond dispersing the white light… A diamond that is cut and polished to produce a high level of these qualities is said to be high in light performance.

Wikipedia – Diamond (gemstone)

钻石的火彩效果作为典型的色散现象，其正确渲染也证明了本次实验中色散实现方法的合理性。

参与性介质与体积光



在加入散射之后，中间产生了体积光的效果。

十二、项目依赖与运行方法

本次作业没有使用 C++标准库及 PA1 vecmath 库之外其他的库。项目需要 C++11 才能够正常编译。在/code 目录下运行 run_all.sh 可以对项目进行编译，运行 gen_bunny.sh 可以生成样例。

十三、参考资料与致谢

感谢胡老师和助教们一学期的细心指导。

本次作业采用了 PA1 的代码框架实现，充分利用了之前的代码。

感谢以下公开资料（资源）作者们的分享精神：

1. smallpt

   smallpt 帮助我快速了解了路径追踪算法的逻辑，并在折射

的处理、采样的模式两方面起到了重要的参考作用。

链接：

https://www.kevinbeason.com/smallpt/

2. 关于分离轴定理的教程，解决了包围盒求交问题

链接：

https://gdbooks.gitbooks.io/3dcollisions/content/Chapter4/aabb-triangle.html

3. BMP 转换器，用于制作纹理贴图

链接：

https://image.online-convert.com/convert-to-bmp

4. 法向插值教程

链接：

https://www.flipcode.com/archives/Interpolating_Normals_For_Ray-Tracing.shtml

5. opengl-tutorial 网站的法向扰动教程

链接：

http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-13-normal-mapping/

6. 从波长到 RGB 值的转换公式

链接：

https://gist.github.com/friendly/67a7df339aa999e2bcfcfec88311abfc

7. 在线 B 样条曲线编辑器

链接：

https://natevm.github.io/B-Spline-Curve-Editor/

8. 维基百科，了解算法和物理公式的重要资源

部分参考词条：*Bounding Sphere, Newton's Method, Normal Mapping, List of Refractive Indices, Cauchy's Equation, Fresnel Equations, Schlick's Approximation, Supersampling, Scattering.*

链接：

https://en.wikipedia.org/wiki/Main_Page

9. 学长的 git 仓库，提供了部分材质和模型资源

链接：

https://github.com/hyz317/Computational-Graphics-THU-2020
https://github.com/Guangxuan-Xiao/THU-Computer-Graphics-2020

10. Depth of Field in Path Tracing，解决了景深问题

链接：

https://medium.com/@elope139/depth-of-field-in-path-tracing-e61180417027

11. Pete Shirley's Graphics Blog，解决了运动模糊

链接：

http://psgraphics.blogspot.com/2016/02/motion-blur-in-ray-tracer.html

12. godRay 项目的 Github 主页，了解了散射相关的物理知识

链接：

https://noahpitts.github.io/godRay/

最后特别感谢袁桢淏同学。我们在遇到算法和物理公式方面的困难时经常一起动手推导，互补了数学和物理知识上的短板。他在找到有用的资料时和我分享，也让我受益匪浅。