计算机图形学作业5:光线跟踪

本作业中，通过加入一些新特性，前面编写的光线投射的渲染能力将大大提升。首先，通过递归产生反射光和阴影光来改进渲染模型。其次，你需要增加过程实体纹理，最后，实现超采样来解决走样问题。

本文档余下内容组织如下：

1. 开始编程
2. 需求概述
3. 初始代码
4. 实现注意事项
5. 测试用例
6. 提示
7. 附加分
8. 作业提交

## 1 开始编程

本作业通常需要做几个星期，请尽早开始。

本作业初始代码中包含了命令行分析器的更新，示例解答程序及新的场景文件。运行示例解答a5soln如下:

./a5soln -input scene12\_vase.txt -size 300 300 -output 10.bmp -shadows

这将生成名为10.bmp的图像文件。同样，试下下面的命令

./a5soln -input scene12\_vase.txt -size 300 300 -output 12.bmp -shadows -jitter -filter

后面我们描述命令行参数余下的内容。在程序编写完成后，你将能渲染此场景及下面给出的其他场景。

## 2 需求概述

如前所述，你将对光线投射增加新特性来改进真实性和性能。 新特性

* 阴影和反射的递归光线跟踪
* 过程实体纹理
* 用于反走样的超采样

在实现细节中，我们会详细描述这些特性。

你的代码将使用脚本来对下面列出的测试用例进行测试。请确保你的程序能处理下面例子中完全一样的命令行参数。方便起见, 你可以假定“-jitter”,“-shadows”和“-filter”总是给定。即，你可以忽略这些参数，并在这些特性使能的情形下运行你的光线跟踪。

## 3 初始代码（Starter）

本作业的部分初始代码源自你完成的作业4。我们更新了一些文件，你需要把这些更新的文件融合到你的代码中。在融合前，请确保备份了你的代码，以免在融合时误损坏了你自己的代码。现在把新的初始代码复制到你的代码中并更新所有内容。当然，你总可以随意改变我们给的内容。

有新的场景分析器来处理新的材料类型。如果你在作业4中已经修改了SceneParser.cpp/hpp,记得在我们提供给你的新的场景分析器中作同样的修改。同样，更新了Mesh.cpp/hpp以加快相交测试的速度。为帮助构建过程渲染器，我们提供一些过程噪声程序。最后我们提供了新的Material类来处理新的材质。请用新的更新你自己的Material。

SceneParser::getBackgroundColor(Vector3f dir)现在把光线方向作为参数. 请相应修改你的主循环。

## 4 实现说明

4.1 递归光线

在融合后，你将会给你的光线投射器增加一些全局光照效果。因为你投射了用于阴影、反射、折射的二次光线,现在代码可称为光线跟踪器了。在RayTracer类中将封装上层计算，该类的作用是发出光线并递归计算沿他们的颜色。

为了计算阴影投射，你需要从可见点向每个光源发送光线。如果有交点，可见点在阴影中，忽略该光源的贡献。 注意，阴影光必须发送到所有光源。为了增加反射(及折射)效果,你需要沿镜面反射（折射）方向发送二次光线,这个请回忆课堂上所讲的内容。更多细节可参考<http://www.cs.utah.edu/~shirley/books/fcg2/rt.pdf>文档。 考虑到多次反射（折射），计算需要递归进行。

1. 确保Material类存储了折射系数(index), 这对递归光线跟踪是必要的。
2. 编写新类RayTracer来计算沿光线幅射 (颜色)。更新主函数使用此类来计算通过每个象素的光线。此类封装了沿光线光线的幅射（颜色）计算。它存储了指向SceneParser实例的一个指针以访问物体几何及光源。你的构造函数应该有这些参数(可能还有其他的,具体依赖于你如何处理命令行参数):

RayTracer( SceneParser\* s, int max\_bounces, ... );

其中max\_bounces是光线最大弹回次数(递归深度).此类的主要方法是traceRay,该方法对给定光线计算从光线原点出发沿光线看到的颜色。该计算会对反射（或透明）材质进行递归计算。因此需要max\_bounces来防止无限递归。最大递归深度作为命令行参数传给程序，使用

-bounces max\_bounces.

用它来试试解答。traceRay以当前回弹次数(递归深度)作为其之一参数。

Vector3f traceRay( Ray& ray, float tmin, int bounces, Hit& hit ) const;

3. 现在应该通过测试上一个作业的场景和现在新的场景来确保你没有破坏之前的功能。做法是通过设置max\_bounces=0.检查除了高光被移去外，其他结果是否相同。

4. (选做) 增加新的命令行参数: -shadows, 表示投射阴影光, -bounces, 控制光线跟踪程序的递归深度。

5. 实现阴影投射。向每个光源发送光线检查这些光线是否与物体相交。如果存在交点，丢弃该光源的贡献。注意你需要移动光线原点稍微离开物体表面，或者等价地设置tmin为某个epsilon。

6. 对反射材质实现镜面反射。从当前交点沿镜面反射方向发送光线。对此，建议编写下面函数:

Vector3f mirrorDirection( const Vector3f& normal, const Vector3f& incoming );

使用修改过的递归深度来递归调用traceRay跟踪二次光线.确保traceRay检查合适的终止条件。将反射光线所看到的颜色乘以反射色加到当前光线计算的颜色中。如果光线不与任何东西相交,简单地调用SceneParser::getBackgroundColor(dir)返回背景色.

7. 简单折射. 对透明材质投射折射光(如果Material::refractionIndex>0).空气的折射率为1.可以假定总从空气中开始。初始代码以traceRay的一个附加参数来跟踪当前折射系数.这对进入另一透明物体将不再正确工作. 如果愿意，你选择你自己的实现.建议你编写函数:

bool transmittedDirection( const Vector3f& normal, const Vector3f& incoming, float index n, float index nt,Vector3f & transmitted);

n和nt是当前物体和光线要进入物体的折射率.参考<http://www.cs.utah.edu/~shirley/books/fcg2/rt.pdf>中的完整解释

令d是入射光方向,N是法向.注意如果光线当前在物体内，d·N 将是正数.

令n和nt是二种介质的折射率，则折射方向t是



首先需要确定是否有折射，如果没有折射(全反射)返回false.这通过检查根号内是否正值确定。

然后，如果存在折射,则应使用同样的traceRay函数来得到递归光线的颜色。

最后，你需要混合反射和折射光的颜色，使用Schlick对Fresnel方程的近似.我们会计算出对反射颜色的权重R，而对折射颜色使用1−R. R 由下式给定：



4.2 Perlin噪声

接下来使用Perlin噪声来创建材质,这让你能在你的过程纹理中增加可控不规则性。关于其发明[Ken Perlin是这样说的 (http://www.noisemachine.com/ talk1/):](http://www.noisemachine.com/talk1/)

噪声看起来随机，但事实上不是。如果真的随机，那么每次调用将得到不同结果。事实上它是“伪随机”—看上去随机。其外观类似于取一大块随机值并模糊它所得到外观效果 (即:使用高斯核卷积).尽管需要大量计算。

[因此我们将使用更高效地(最近改进的)噪声实现(http://mrl.nyu.edu/ ~perlin/noise/) 转换成C++代码在](http://mrl.nyu.edu/~perlin/noise/)PerlinNoise.{h,cpp}.

PerlinNoise::octaveNoise已经给你实现了,它计算

N(x, y, z)= noise(x, y, z)+ noise(2x, 2y, 2z)/2+ noise(4x, 4y, 4z)/4+ ...,

其中 (x, y, z)是全局坐标

Ken Perlin的原始论文和其在线注释中有很多由噪声函数可以创建的过程纹理很酷的示例。你需要实现一个简单的大理石材质.此材质使用sin函数得到色带来表示大理石脉。用下面噪声函数对色带扰动:

M(x, y, z)= sin(ωx + aN(x, y, z))

补充类Noise,其用合适的参数调用函数PerlinNoise::octaveNoise以得到N(x,y,z)。然后计算M(x,y,z). M(x,y,z)的值是浮点数，应对其进行固定并用于在两个包含的颜色之间进行插值。下面是Noise的构造函数:

Noise( int octaves , const Vector3f & color1, const Vector3f & color2, float frequency, float amplitude);

上式中，频率为w幅值为a

试验不同的参数设置以理解你能得到的各种不同的外观。切记，这并不是大理石的物理模拟，但过程纹理试图模仿我们看到的东西。

4.3 反走样

接下来，需要在光线跟踪器中增加一些简单的反走样。使用超采样和滤波来减少锯齿和云纹模式.

1. 抖动采样

对每个象素，使用随机扰动后的多条光线采样来代替一条光线得到的颜色。你需要将一个象素分成3×3子网格. 这等价于用3倍分辨率来渲染图像.然后对高分辨率图像中位于(i, j)的每个象元，产生范围[−0.5,0.5]内的二个随机数ri,rj得到(i+ri,j+rj)，而非仅使用(i,j)来产生光线.

2. 高斯模糊. <http://en.wikipedia.org/wiki/Gaussian_blur>在解答中,我们使用核K = (0.1201,0.2339,0.2931, 0.2339, 0.1201). 首先水平方向模糊图像，然后竖直方向模糊图像。

水平模糊图像是对每个新图像的象素色I'(i, j)计算原图像多个象素的加权平均。

I'(i, j)= I(i, j − 2)K(0) + I(i, j − 1)K(1) + I(i, j)K(2) + I(i, j + 1)K(3) + I'(i, j + 2)K(4).

注意，你可能希望分配一个新的图像来实现模糊以避免bug.如果(i,j−2)超出图像边界，可使用(i,0)等

1. 下采样. 为了从高分辨率的图像得到原来指定分辨率，对每个象素的3×3邻域平均得到1个象素值.
2. (选做) 处理命令行参数:

• -jittered 使能抖动采样

• -filter 使能高斯平滑及下采样

如果忽略这些参数，你的程序会总是使能此二步。

## 测试用例

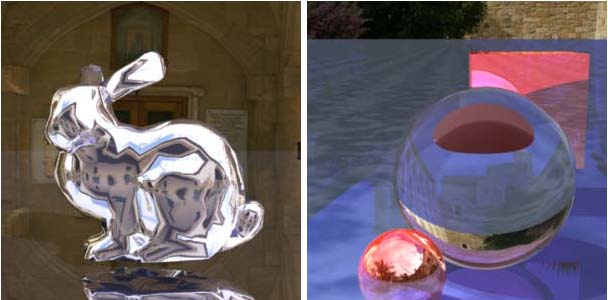
你的作业将用脚本运行来评分，具体示例如下。请确保你的光线跟踪程序产生类似的输出

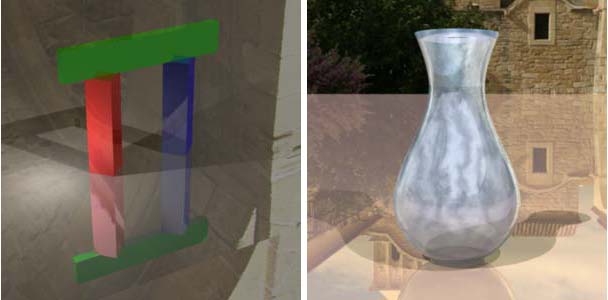
./a5 -input scene06\_bunny\_1k.txt -size 300 300 -output 6.bmp -shadows -bounces 4 -jitter -filter

./a5 -input scene10\_sphere.txt -size 300 300 -output 10.bmp -shadows -bounces 4 -jitter -filter

./a5 -input scene11\_cube.txt -size 300 300 -output 11.bmp -shadows -bounces 4 -jitter -filter

./a5 -input scene12\_vase.txt -size 300 300 -output 12.bmp -shadows -bounces 4 -jitter -filter





## 6 提示

* 不必把类中所有方法都声明为virtual, 只需对子类将重载的这样做。
* 打印足够多的信息来调试。当得到奇怪的结果时，可看一下简单的情形，并手算来检验你的结果。可以考虑仅投射一条光线（及其后的递归光线），而不是构造图像用的所有光线。
* 修改测试场景减少调试的复杂性: 移去物体、光源, 改变材质参数，以便能看到不同组分的贡献等。
* 为避免segmentation fault,请确保不要访问超过图像宽度和高度的区域。边界上的象素有一个裁剪支持区域。

## 7 附加分

大多数扩展需要修改分析器来处理附加特性。请确保产生(并提交)合适的输入场景来展示你的扩展。

7.1 易

* 使用Perlin噪声创建木材材质或其他过程材质。
* 场景中增加更有趣的光, 如随角度衰减的聚光灯。
* 景深渲染(如果在上个作业中没有做此).

7.2 中等

* 凹凸映射(如果上个作业没有做): 在高度场图像或者法向映射中为你的表面查找法向量.这需要求导得到切向框架。你可能需要在网上找到这样的免费图像和模型。
* 双向纹理函数(BTFs): 让你的纹理查找依赖于视角.网上有这样的数据集（[链接）.](http://cg.cs.uni-bonn.de/en/projects/btfdbb/download/)
* 装入和创建更多有趣的复杂场景.你可以从网上下载更多模型和场景。
* 增加面光源和软阴影的Monte-Carlo积分
* 渲染不光滑（glossy）表面
* 渲染有趣的BRDF,例如铣削表面
* 非直接光照的分布式光线跟踪(非常慢).投射大量随机二次光线采样可见点的半球区域。建议在一次回弹后停止。一致采样或者根据cosine项采样(注意，对半球一致采样并不很简单).
* 一致格网.构建3D格网并“光栅化”你的物体到其中。然后，移动光线通过格网，仅在当前体元有交点时停止。调试会比较困难。
* 仿真色散 (及彩虹). 仿真彩虹是困难的,因为要实现Newton棱锥演示。
* 生成一个小动画(10秒24fps就够了).例如，若实现了景深,展示改变焦距的动态效果。移动光和物体。
* 在运动物体的动画中增加运动模糊

7.3 难

* 光子映射（Photon）及kd-树加速来渲染焦散。
* 辐照度缓存（Irradiance caching）
* 次表面散射.渲染牛奶, 翡翠等.
* 带有重要性采样的路径跟踪, 路径终止于俄罗斯轮盘赌等
* 透体光线跟踪.给定规则格网编码某个介质（如雾）的密度，仿真光线进入格网由于雾而产生的衰减。向光源发送光线，并考虑到由于其他物体而产生阴影及中间介质所产生的衰减.你将看到忽明忽暗的效果。
* 做出往容器中倒水或者烟雾升起的动画。

## 8 提交作业方式

请写README.txt (或者PDF)来回答下列问题:

* 如何编译代码? 不必告诉如何运行，因为代码必须按前面提供命令行参数来运行。
* 是否有与其他同学合作? 如果有,请说明和谁交谈，并说明得到或者给予了什么样的帮助。
* 有什么参考资料 (书, 论文, 网站等)对你完成作业特别有帮助? 请提供一个列表
* 你的代码有已知的问题吗? 如果是，请提供一个列表，如果可能，请描述你认为造成的原因是什么,如果给你更多时间，你如何解决他们。这很重要，如果有助于让我们理解问题所在，我们会为此给你附加分.
* 如果你做了附加分，请告知如何使用附加特性。如果涉及大量工作，请描述之，并说明如何做的。
* 关于本作业你有些什么心得可以共享?

请提交单个压缩文件(.zip 或 .tar.gz)包含:

* 源码
* 名为a4的编译好的执行文件.
* 任何必要的附加文件
* README文件