光线追踪报告

计科20 2012012440 宋正阳

2015 年1 月23 日

1 算法综述

光线追踪为一经典算法，故在此不再赘述。大致来说就是首先建立起场景后设定视平面，然后设定摄像机位置及观察方向。之后对于视平面上的每一个点，由摄像机向其投射一条光线，若其未碰到任何物体则返回背景颜色，否则根据其所碰撞材质的反射折射散射特性衍生出其他光线，直至达到最大追踪深度。据此计算出视平面上每个点的颜色值，并将其描绘出来。

2 代码综述

本次作业根据Kevin Suffern 所著Ray Tracing from the Ground up 中代码框架。具体如下：

2.1 World

World 类包含了与场景渲染相关的所有内容，如视平面，摄像机，光线追踪器，场景所含几何物体，场景所含光源，背景颜色以及环境光。整个程序的运行从World 类的build 函数开始，在其中可以设计各种不同的场景，以及设定以上各项。然后直接调用Camera 的render_scene 函数便可以对场景进行渲染。值得注意的是对于不同的需求我们可以设定不同的光纤追踪器以及不同的摄像机来渲染出想要的不同效果。

ViewPlane 类则是包含了视平面的大小，像素点的大小，以及每个像素点采样个数，光线追踪的最大深度，采样器种类等等。抗锯齿部分我们便是采用抖动采样器来进行处理的。

2.2 Cameras

上回说到World 会调用Camera 的render_scene 对场景进行渲染，不同的摄像机会采取不同的投射光线的方式。我们可以通过继承扩展Camera 基类达到不同的效果。如我们这次实现的PinHole(针孔相机) 是由摄像机位置向视平面各点投射直线，其他摄像机(如鱼眼相机)可能会在此基础上有一次偏折。所投射每一条光线的原点及方向确定之后，便直接调用World 中光线追踪器对这条光线所应返回的颜色进行计算。

2.3 Tracers

我们可以通过继承扩展Tracer 类来得到不同的光线追踪器，不同的光线追踪器的区别在于调用材质的函数不同，例如我们直线的两个Whitted和AreaLighting, 在检查追踪深度是否已超过最大深度之后，有则返回黑色。否则检查此条光线是否与场景中的物体有碰撞，没有则返回背景颜色。否则Whitted 对于碰撞物体的材质调用shade 函数，而AreaLighting 对于材质调用area_light_shade 函数。这便要求我们对于不同的光线追踪要求，除了编写对应的光线追踪器外，还要对所有材质编写相对应的返回函数。追踪过程中所有的颜色信息都存放在ShadeRec 中。

2.4 Materials

不同的材质的shade 函数不同，即对于确定的投射光线会表现出不同的特征。举个例子来说，Matte 材质为无反射，故其直接根据表面的diffuse 遍历所有光源根据光线返回颜色值。而Reflective 为反射材质，其除了Phong 材质的shade 返回值外，还会计算出反射光线，并继续追踪，并最终返回两者的加和。

2.5 ShadeRec

ShadeRec 类在光线追踪过程中起到了信使的作用，其包含了丰富的信息。例如光线是否碰撞到了物体，碰撞的材质如何，碰撞点坐标，碰撞点的法向量，光线追踪深度，光线的参数信息，还有便于后续过程引用的World 类指针。

2.6 Samplers

Samplers 类包含了多种不同的采样方式，其包含一个2DPoint 的数组，调用不同的生成采样的函数会将此数组中生成不同的采样点。在可选的锯齿处理部分我们实现了抖动采样器MultiJittered. 其首先将采样区间根据采样数划为一个个小方格，然后在每个小方格中随机取点。不同的采样器的区别在于生成samples 数组的generate_samples 函数不同。

2.7 GeometricObjects

GeometricObjects 包含了各种不同的几何物体，几何物体的形状是通过hit 函数来体现的。对于不同的物体只需编写不同的hit 函数便可容易地在渲染结果中加以体现。我们目前只实现了Sphere 和Plane 及Rectangle, 值得指出的是其他几何物体的实现亦将十分简单。

2.8 Lights

Light 类包含各种不同的光源，如环境光、方向光、点光源，以及可选部分的面光源(软阴影)。不同的光源在返回颜色值以及判断是否有阴影时都会有不同。

2.9 BRDFs, BRTFS

BRDF 与BRTF 是物理相关的一些函数，指导了不同的入射方向应该返回何种颜色值。

2.10 Utilities

Utilities 中包含了其余的各种辅助类，如常数，数学运算，矩阵，向量，点，光线，颜色，法向量等。

2.11 Textures

Texture 中Cherker3D 的实现最为简单，我们只是简单地将三个坐标值相加，然后根据返回的模数设定不同的颜色值。

而图像纹理则较为复杂，首先Image 类会读入ppm 格式的文件，计算出水平竖直方向的像素点个数(hres, vres),并将所有的像素点颜色值压入vector 数组pixels.

而Mapping 类会根据传入的交点坐标计算出应该返回Image 中像素点的坐标，在这里我们只实现了SphericalMap, 其他的Map 只要重写get_texel_coordiantes 即可。

MapTexture 则是包含两个指针，分别为Image 和Mapping, 这样就可以由Mapping 计算坐标然后在Image 中取出应该返回的像素点颜色值了。

3 用法

只需根据不同的需要在World 的build 函数中设定视平面长宽距离，摄像机种类，光线追踪器种类后，设定光源信息add_light 添加光源，设定物体信息add_object 添加物体。之后调用摄像机的render_scene 函数即可。

4 编程体会

本次作业过程中的经验总结如下：

1. 定义类名时要注意不要与C++ 标准库中已有名字相同。
2. 在传递不需要改变的参数时const & 效率最高。
3. 纯虚类(virtual 后跟= 0) 必须在继承子类中实现。
4. 不改变原有对象的函数可在声明后加const.
5. 对于类中声明的所有virtual 函数均要给予相应的实现。
6. 多态中要注意所有参数类型及返回值类型均要一致。
7. 注意理解各参数的含义以及设定合适的参数。
8. 多与同学讨论有助于高效地解决问题。