**作业五**

# 1.如何编译代码

使用VS2017编译，把文件中的代码覆盖到完整的原始项目当中。

如果要运行exe，请把tex和其他的TXT场景文件放到exe的文件下，生成文件会在Result\_Picture中。

注：默认为包括抖动采样与反走样，时间会比较慢。Maxbounce默认设为3。

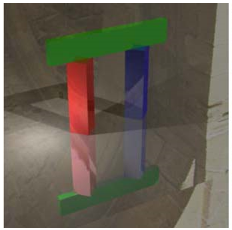
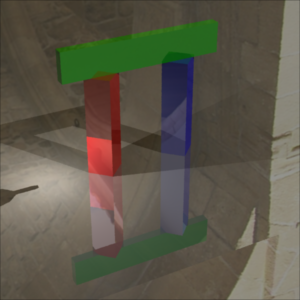
# 2. 合作与给予的帮助

（1）在RayTracer中的光线颜色计算方面，之前作业4计算可能也存在错误。即加上环境光光照是环境光颜色乘上漫反射系数。而对于镜面反射与折射的光颜色计算，都需要在前面乘上一个高光系数。这个目前理解的也比较模糊，一直以为计算的颜色就直接加上去就好。但是乘以一个高光或者漫反射系数原因应该是对特定的光产生一定的区别，如果高光就直接强化这种光。

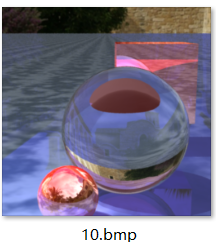
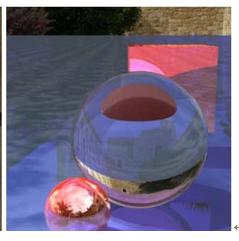
（2）对于Perlin noise处的差值颜色想不清楚，问了同学，这个地方类似折射部分，需要计算一个M（也就是折射的R），作为两个颜色的权重比，这样能够有插值的效果。

# 3.已知问题

（1）生成的第11个场景在左下会多出一个阴影。而且蓝色与绿色交界处不一样（感觉是找不同游戏。。），问题应该是出在折射的部分，可能折射光方向计算的不正确。



（2）在第10个场景中的后面会多生成一个黑框。

# 4.附加分

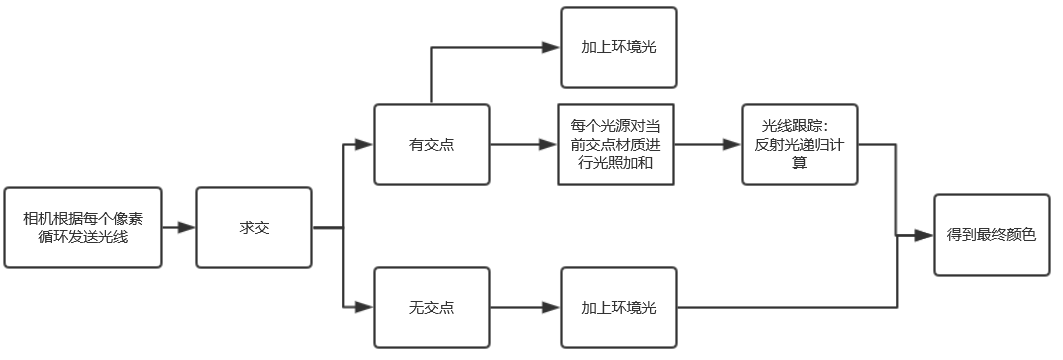
无。

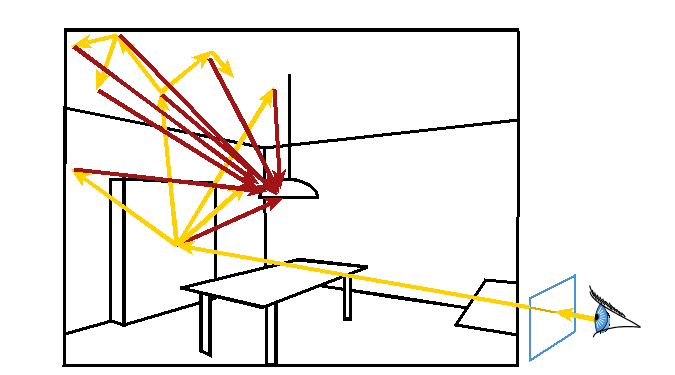
# 5.心得与分享

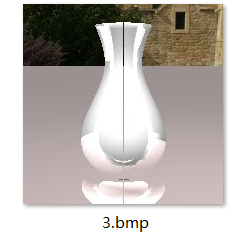
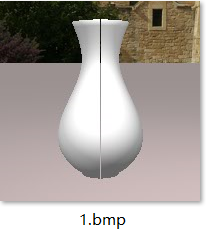
## 5.1光线跟踪

（1）首先应该先理解RayTracer的真正作用。就是当成相机发出的光即可。内部会计算所有的光照颜色，包括漫反射，镜面反射，折射，阴影光等。也就是替换之前在作业4里写的双重for循环像素的内部的求交部分。

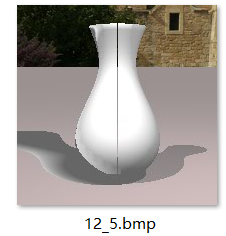
（2）对于光线跟踪，类似于一个不断镜面反射的过程。也就是眼睛看到的光是多个点的光的加和。需要累加在每一个反射点的所有光照。因此一开始就去实现镜面反射函数即可，通过当前交点以及求得的反射光方向，得到新的反射光线，然后再递归下去。







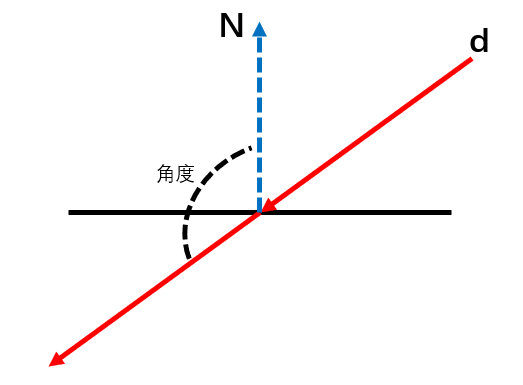
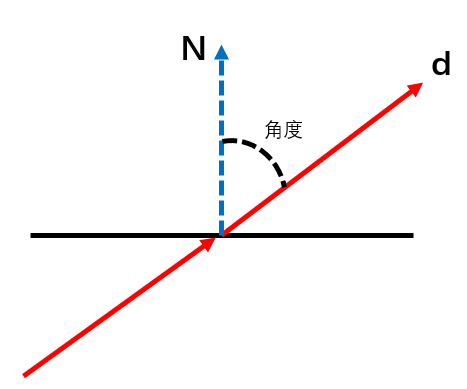
（3）阴影光只是附加一个条件，就是如果有物体遮挡，直接舍弃当前光源的贡献。因此在加光源的光照之前先在指向光源的方向进行一个求交，如果有交点就不做贡献。



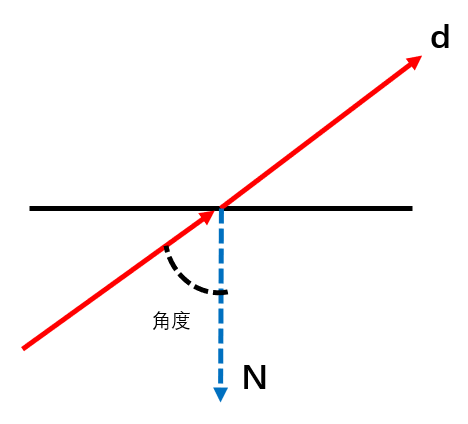
（4）镜面反射。也就是把之前的递归函数做一些调整，在递归颜色的前面乘上一个高光系数，代表是一个镜面反射。

（5）折射。坑点比较多。

首先需要理解d点乘N的含义。为什么点乘正数就在物体内部，负数在外部。例如左图，也就是从空气（物体外）射入到物体内时。由于法向量始终是指向外部的，那么可以看到夹角为钝角，计算时cos为负，那么点乘小于零。而右图，从物体内射到物体外时，夹角为锐角，则点乘大于0。

当然前面解释了点乘正负的原因。而且只是判定，还没有开始计算折射光线。计算前，我们需要上一步的点乘作出判定，而且假设只存在两种情况，一是在物体外（空气里），那么当前折射系数为1，二是在物体内，就是hit的Material内的折射系数。不会同时在两个物体内，不然交界处的折射会看成是空气。如果在物体内，我们需要改变法向量的方向，也就是让它反向，这样直接使用原来的折射公式即可。



总的来说，需要理解点乘的概念，将折射系数n和nt两个值设对，正确处理法向量方向。

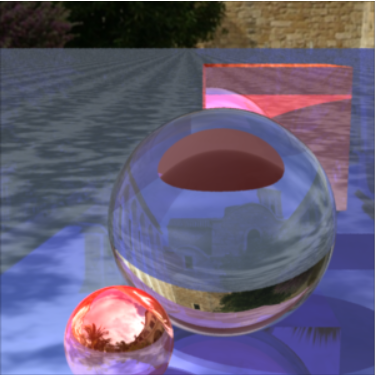
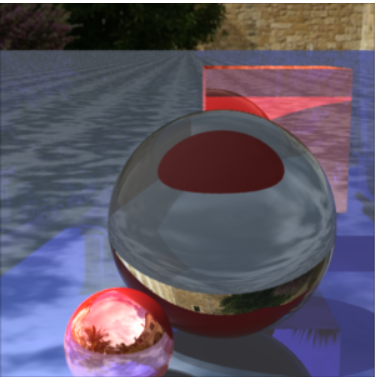
其他的就是需要完成折射函数，只需要根据公式计算即可。而对于完全实现折射。需要类似于反射的做法。也是对当前折射光进行递归处理，并乘上一个高光系数。最后就是简单计算一个R权重，然后根据权重比加和镜面反射与折射光照。

（6）其他细节部分。

需要注意每个试探性求交或者递归的镜面反射或折射求交都要先重定义一个hit，防止改变当前参数中的hit。

需要注意对方向向量正则化（对于反射折射计算之前也需要注意将参数向量正则化）。注意incoming的含义，主要是在光线打入时是从外来到当前交点，而对于其他计算可能是从当前交点向外发射。即方向相反，需要注意。

对于总体RayTracer的结构要考虑清楚。并且如果当前光已达最大光照次数，不是之后不需要进行镜面反射与折射，而是要在最后加上一层。例如左面是判断达到最大就不再进行反射，右图是到最后再进行最后一次反射。



## 5.2perlin noise

这部分比较简单，噪声的求解已经给出，只是使用一个perlin公式即可。然后再进行颜色插值。

插值的计算相当于两个颜色（注意到在构造函数时给出）的比例，并把两个颜色进行加和。



## 5.3反走样

根据公式，抖动采样就是每个光线的方向变为随机，只是在当前方向产生较小的误差抖动，其中我们需要先把图片变为高分辨率再取平均值（下采样）这样做更容易具有普遍性。并且作出抖动时我们可以看到之前图片的一条黑线消失了。黑线的成因，猜测是三角形的交界处正好在正中间，并且光线正好直接穿过交界处（包括后面的三角形交接），那么无法命中任何物体（三角形），而采用抖动采样会将光线偏离交界的一小部分，那么就会命中三角形了。



高斯模糊是类似于CNN中的卷积层，只是保存物体的大致形状而对其他进行模糊虚化，这样看起来更加光滑。做法是把高分辨率的图片先进行横向虚化再进行竖向虚化，需要注意临界值的判断。

下采样类似于CNN的Pooling层。是把高分辨率的图片化为低分辨率图片，做法是成比例对图片像素颜色进行平均。

## 5.4总结

感觉本题目主要难度在于理解RayTracer的真正含义与内部结构设计。它主要是为了计算光线的颜色，并把所有颜色进行加和。而对于光线跟踪只是一个比较抽象的概念，具体是把光线拆分成镜面反射与折射两部分，分别进行光线递归。之后就是注意一下各种细节了，当然也需要深刻理解文档中所提出的注意事项。虽然最终代码量可能不是很多，但是需要对题目有一定的理解才能更好的去实现。