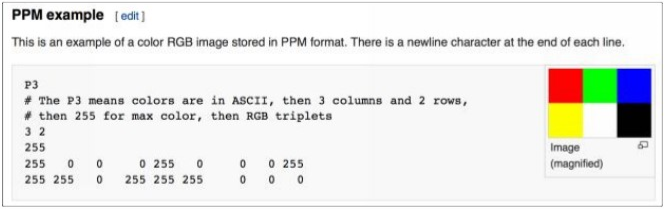
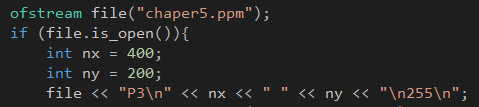
Chapter 1: Output an image

1、我们要通过一个叫ppm格式的文件来显示图案，下载ppmviewer软件才能打开该文件；在IDE中，我们要通过代码建立一个ppm文件，文件要求一定的输入格式：



P3代表文件输入输出的格式，为pixel即像素模式；然后3 2代表列数和行数，可以理解为宽和高，底下的255代表像素的最大值；ppm文件会从左到右、从上到下的顺序绘制像素；像素是以RGB的颜色模式来绘制的，即需要提供三个RGB的值；

\*通过文件的方式建立ppm文件并输入的代码如下：

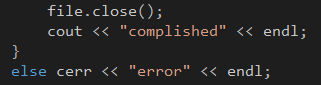


可理解为我们看到的屏幕是400x200的分辨率

设置像素点的RGB颜色：



关闭文件：

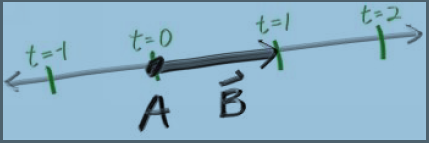


Chapter 2: The vec3 class

1、这一章我们要自己实现一个向量类vec3，用于存储几何向量和颜色；这个类有三个坐标，可以用来表示颜色、位置、方向、偏移等；

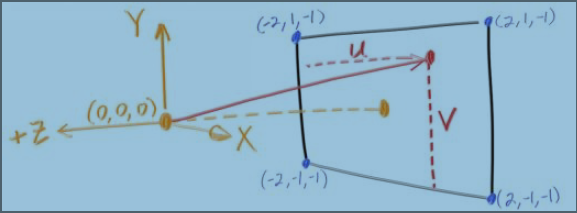
Chapter 3: Rays, a simple camera, and background

1、建立一个光线类，并假设光线方程为*p(t) = A + t\*B*.



A为光线源点，B为光线方向，t为参数，该方程可计算出沿着这条线看到的颜色

2、我们在一个长方形的区域（背景屏幕）内来实现光线追踪器，设向上为y轴正向，向右为x轴正向，采用右手坐标系（大拇指正向是x正向，食指正向是y正向，中指正向是z正向），故朝屏幕里面方向为z轴负向，往外看是z轴正向，如图：



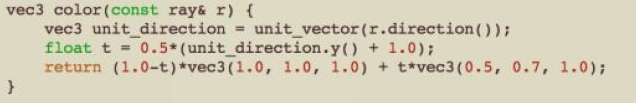
我们假设眼睛（或者说照相机）位于长方形区域的中心位置，且屏幕位于眼前1单位的距离；则如上图所示，左下角是（-2，-2，-1），我们将从左下角开始，通过光线在屏幕上“打点”；我们将假设光源最初射向左下角，然后根据代码中的循环，利用两个偏移量，将光线设置为从左向右、从上至下照射打点；

3、从左下角（-2，-2，-1）开始，由于该区域的x轴为[-2,2]，故偏移量该设置为（0,4,0）；y轴同理，为（0,2,0）；代码的循环其实就是从左到右，从上到下设置光线（用0-1的小数乘以偏移量），找到每一个像素点（得到像素的位置坐标）；

4、于是每次循环迭代我们得到一条光线（一个像素点的位置坐标），我们要根据像素点的y坐标来线性插值设置像素的颜色，即从上到下，插值颜色蓝色到白色（或者说从下到上，插值白色到蓝色），插值公式为：

*blended\_value = (1-t)\*start\_value + t\*end\_value*

也称LERP值，当t=0时，lerp值等于起始值，当t=1时，lerp值等于末尾值；



于是当t=0的时候，我们得到vec3(1,1,1)即白色，当t=1的时候我们得到vec(0.5,0.7,1.0)蓝色；即r光线方向的y分量越大，t越大，获得的结果越蓝，反之越白，这正和结果符合；

**#**此几行代码是理论上能达到的线性插值，因为虽然归一化之后，y的确属于[-1,1]区间，但是因为该平面的z值为-1，所以归一化之后y无论如何不可能恰好为-1与+1，这种情况只有(0,y,0)与(0,-y,0)归一化之后才能相等；具体来说y值的区间会小于[-1,1]，而实际的插值也是从该区间开始，如t属于[0.25,0.75]，那么结果就不会是纯蓝色与纯白色的插值；

Chapter 4: Adding a sphere

1、这一章我们要设置一个球体，其圆心位于z轴的-1处，即上一章的长方形区域上，我们要利用光线追踪器判断光线是否触及到球体，触及到的位置用红色像素表示；

2、通过数学知识，推导出的公式是：*t\*t\*dot(B,B) + 2\*t\*dot(B,A-C) + dot(A-C,A-C) - R\*R = 0* （书上的公式写错了）此公式为一元二次方程

（变量t），根据判别式a2-4ac，若大于0则有两个解，该光线穿过球体，若等于零只有一个解，该光线与球体相切，若小于零则无解，光线没有接触球体；

对于该方程：

a=dot(B,B)=dot(r.direction()\*r.direction()); （B为光线的方向向量）

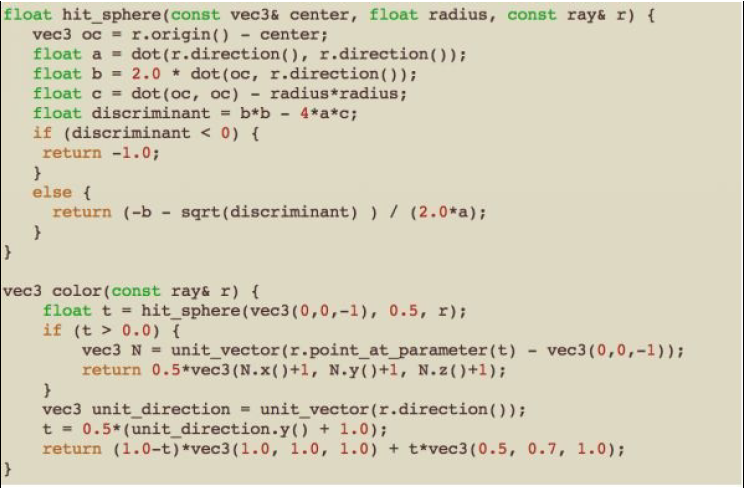
b=2\*dot(B,A-C),其中A=r.direction()，A=r.origin()，C=center

c=dot(A-C,A-C)-R2;

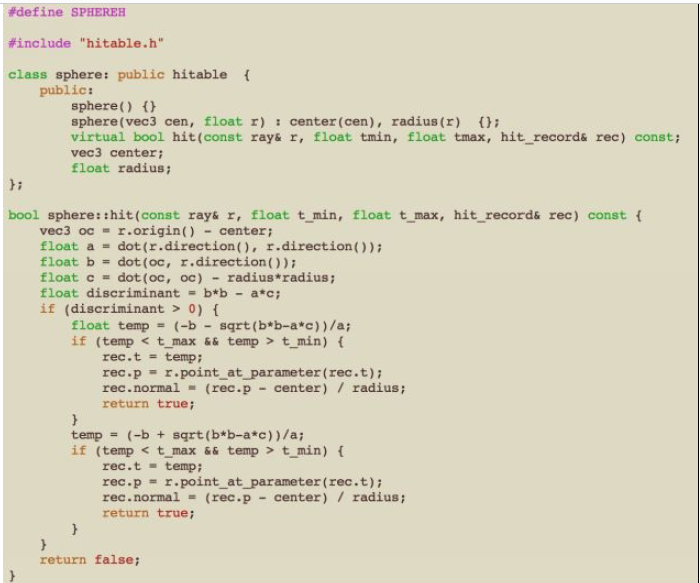
Chapter 5: Surface normals and multiple objects.

#补充：这节课是要使用光线与球面交点的法向量来作为颜色的值，就是将法向量的三个分量作为像素的RGB分量；

1、我们首先要求得该球体前半个球面的法向量（即与光线相交的点的法向量），然后将法向量归一化再将每个分量映射到[0,1]，最后根据每个分量设置元素的RGB值，实现颜色；由于我们光线往球体射去，先与前半球面相交，故在上式一元二次方程的求解中，解为较小值t，根据解出来的t即可通过光线类point\_at\_parameter（t）方法求得光线与前半球面的交点向量，然后用该向量减去球心向量即可得到该点的表面法向量；（\*此节注意hit\_sphere的返回值变为float若不注意会让你debug很久）；

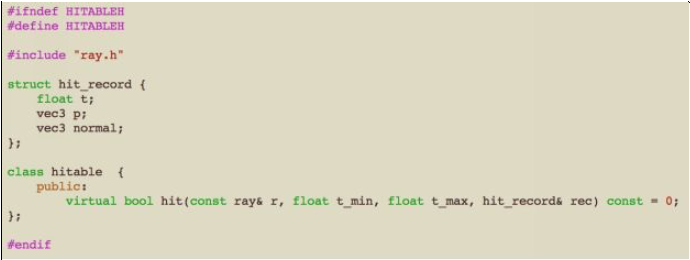


2、接着我们要绘制多个球体或者多个其他物体，我们要构造一个抽象类hitable，该类会拥有一个接受光线的命中函数，即检测到被光线命中会进行记录，需要注意的一个小技巧是，我们对于光线检测会有一个有效的步长限定（tmin,tmax）；这是因为，我们要对每一条光线路径上的每一个物体进行检测判断，并对符合条件的t进行筛选(由于我们只能看到离我们最近的物体，后面的物体会被遮挡，相机后方的物体我们也无法看见，所以我们需要通过这个限定值来更新离光源最近的交点---在递归的过程中，每次命中物体的交点最为tmax，以对后续命中的物体进行筛选，比tmax更小，即更近的物体才会更新交点)

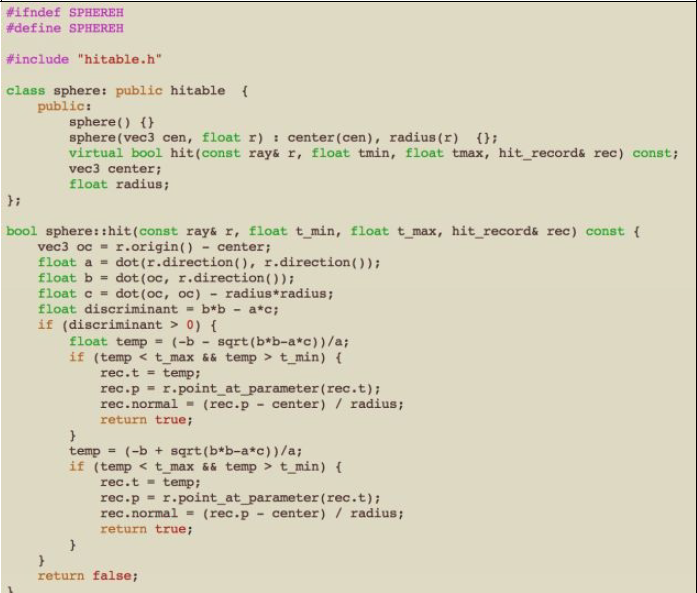


3、对于所有物体的检测，我们需要构造一个hitable\_list类，该列表存放所有物体的指针，由于列表元素是一个指针，所以整个列表我们需要用一个二维指针来表示，就如同若存放的是一个int数组a[n]，则a是一个Int\*指针，若存放的是一个int\*数组int\* a[n]，则a是一个int\*\*指针；故该列表用hitable \*\*l来表示；

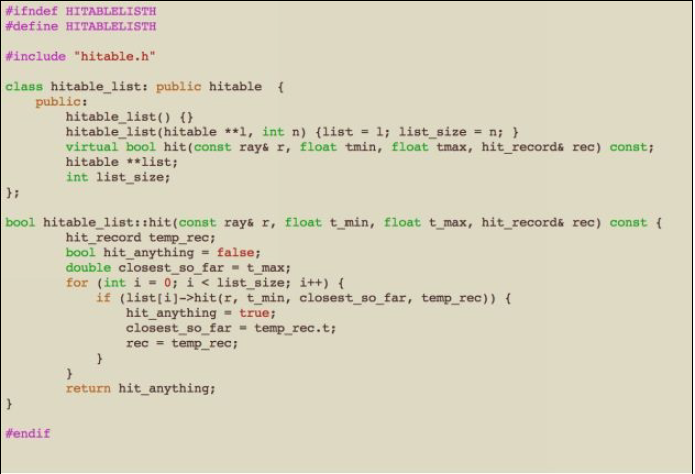
4、代码解析



Hitable头文件包含一个hit\_record记录类，记录利用方程求解出的步长t，以及光线与物体的交点坐标p，还有该点的表面法向量normal；一个hitable抽象基类，只包含一个纯虚方法hit，用来检测光线是否与物体相交；



Sphere头文件包含sphere类，它继承自hitable抽象基类，其继承作用其实只在于那个hit虚方法的具体实现，即根据球体的一元二次方程求出两个解t，先求较小值，再求较大值，只要有解满足条件tmin-tmax就返回true；自身部分包含了球体类的构造（球心、半径）等；



Hitable\_list头文件包含hitable\_list类，它继承自hitable抽象基类，其数据成员包含一个hitable父类的指针的指针，相当于一个一维列表，其列表元素是一hitable\*，那么list就指向该列表的表首；其目的是可以将指向子类对象的指针放入这个列表中，然后在hitable\_list类中的hit方法就是，对于特定的一条光线，遍历这个hitable\*列表，对每一个物体（hitable派生类对象），使用自己特化后的虚方法hit()进行检测，同时更新最近的点，修改hit\_record记录；可以说这个hit虚方法中又调用了特定对象的hit虚方法；



首先建立一个hitable数组list[2]，里面有两个可检测（hitable）的元素，然后我们构建两个球体类实体，并将其放入数组中（因为sphere派生自hitable，所以sphere\*可以放入hitable\*的数组中）；然后我们构造一个hitabale\_list对象，并用list（此刻是一个hitable\*\*）初始化hitabale\_list类中的list，2初始化为列表长度，new的返回值应该是一个hitabale\_list\*，但是由于它仍然派生自hitable，所以可以用一个hitable\*接收；

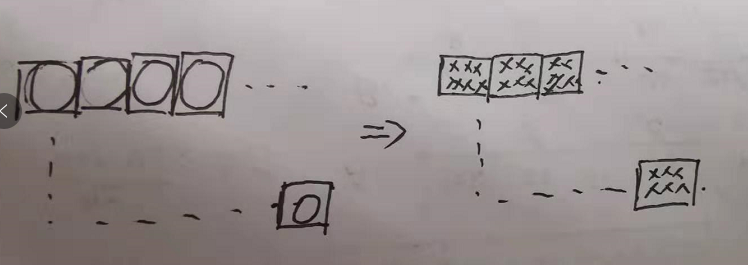
#补充，hitable \*word=new hitable\_list(…)，是用父类指针指向一个派生类对象，这样才能通过父类指针调用子类对象中的虚方法；

接着我们开始总结循环中代码的含义：对于每一条光线r，我们检测它可能穿过的任何一个物体（存放在world列表里）,并且我们找出离光源最近的交点，并根据交点表面的法向量进行着色（法向量分量皆被映射为[0-1]），然后返回一个像素点的颜色col，并绘制在屏幕上，若该光线的方向没有相交的物体，则绘制背景色，即蓝白线性插值混色；

\*上述的指针知识点即用到了类继承当中的动态绑定：可以将基类的指针或引用绑定到派生类的对象上，用指针或引用调用虚方法时，会根据动态绑定的类型决定调用哪一个虚方法（即该派生类的虚方法）；

Chapter 6: Antialiasing

1、这一节我们要实现抗锯齿处理（即边缘锐化），实现在图形的分层区域自然过渡，其原理是，我们绘制的时候是通过光线找到屏幕上的每一个像素点，并根据光线与物体交点的法线向量来确定该像素的颜色，那么当放大的时候会发现，在不同物体的分层处，相邻网格之间的颜色差距比较大，就形成了视觉上的锯齿，那么我们的做法是，对于取到的每一个像素点，再进行划分，在该点和相邻点之间的网格中，取一些样点（比如100个点），我们仍然可以通过函数计算出每一个更细分样点的法向量（也即获得这个点的颜色），然后我们计算这些样点颜色值的平均值，来作为该像素点的颜色，那么就实现了它跟相邻网格颜色的过渡（在数学上类似于将离散值连续化）；如图：



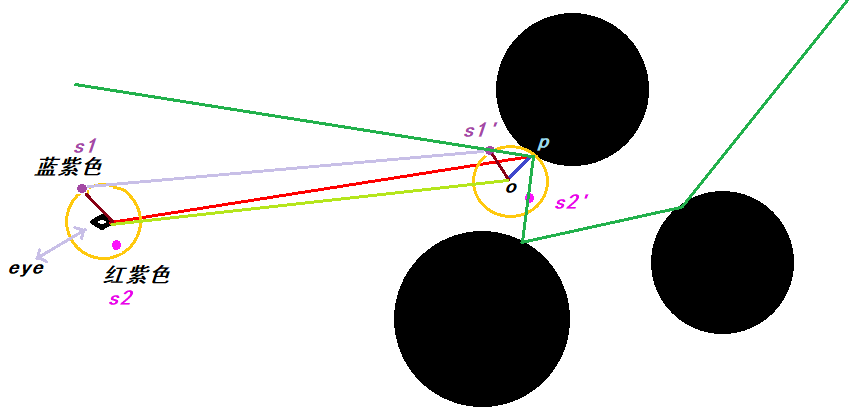
可想象为，一开始从左到右，从上到下，每个像素点由一束光照决定该点的颜色，进行抗锯齿处理后，我们在每一个像素方格中，随机抽取一定数量的样本，并计算出每个点的颜色，取和求平均值，就得到均匀的过渡颜色了；

2、根据比较规范的说法，此处理方法称为使用采样模式对锯齿进行修缮，可借鉴<https://www.cnblogs.com/lv-anchoret/p/10190092.html>上的笔记进行理解；

Chapter 7: Diffuse Materials

1、这一章，我们开始利用光线追踪器制作一些逼真的材质，我们先从漫反射（磨砂）材料开始，光线照射到漫反射材质表面，会具有随机方向的反射光线，并且也可能被吸收掉，这一节的目的就是实现漫反射表面光线的随机反射，其中涉及到一些数学推导，借助图会比较好理解，见：

<https://www.cnblogs.com/lv-anchoret/p/10198423.html>



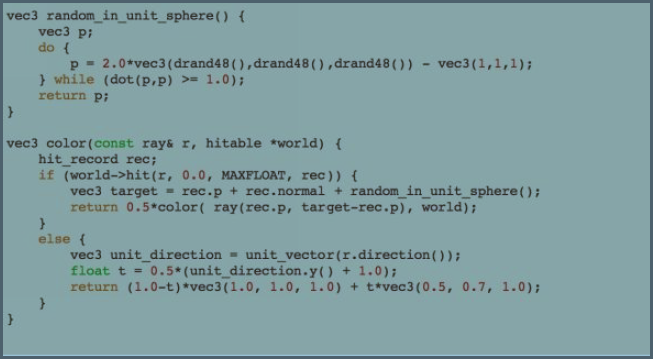
光照射到物体的点p上，我们借助一个与点p相切的单位球体来表达随机的反射光线，根据[Lambertian](https://www.cnblogs.com/jerrycg/p/4941359.html)（朗博体）漫反射模型，在点p处的反射光线的范围只会经过这个单位球体内，也就是说，如果能在这个单位球体中随机找到一点s’（图上是s1’），然后连接p->s’，该向量就是代表了随机的一个反射光线；由于在该球体上直接寻找不方便，我们平移到光线源点，得到同样的一个单位球体，并根据算法得到一个随机点s，那么源点->s向量就等于o->s’向量；又根据推导可得到o点的位置为源点->p+法向量n，那么s’的坐标就可以表达为*p+n+random\_s（源点->s）*；

那么p->s’光线就可以表达为：源点：p，方向：s’- p=n+random\_s；其实反射光线的方向就是random\_s+n；

2、像素成色原理理解：

我们回到一开始绘制的原理，根据光线从左到右、从上到下打点，如果光线没有与物体相交，则根据照射到屏幕上的光线方向的Y值线性插值混色，所以背景图是蓝白色（从上到下），根据光线检测的算法，若光线与物体相交，取离光源最近的点的法向量，对法向量的三个分量映射到0-1后分配给像素点RGB；那么我们要模拟真实世界物体的成像，真实的物体并不是由它某一点的法向量就决定了这个点的颜色，此时引入了漫反射物体，漫反射物体会反射一部分光线，也会（同时）吸收一部分光线，所以每次光线照射到物体的某一个点，光线会衰弱50%（每反射一次衰减50%）；我们看到该点的颜色，正是经过了光线多次的反射、衰减（吸收）看到的，而且反射越多次，光线衰减越多，颜色越暗淡，又由于漫反射光线的方向是随机的，所以对于每条光线，照射到物体上反射的次数也是随机的，所以造成了每个点的颜色深浅不同，造成了哑光的效果；

3、代码解析



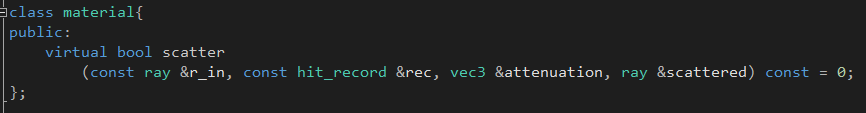
(1)随机生成的向量映射为三个分量都是(-1,1)，再限定向量的长度小于1，那么就实现了随机生成一个单位长度球体内的点；(2)color函数的上半部分其实是递归，每次反射衰减50%的强度，并把交点作为新的源点，继续随机进行反射，直到反射光线没有与其他物体相交，返回此时的光线，并由该光线进行混色；

\*疑惑的部分，虽然不断的反射衰减，但是最后的光线方向又进行了归一化，所以其实最后所有的光线三个分量还是（0,1）之间，感觉没有达到衰减的效果啊…

答：理解错误了，归一化只是递归边界的最后一次，进行了光线方向的归一化，再递归迭代返回的过程中，其实返回的是color，所以颜色是不断递归衰减的；

Chapter 8: Metal

1、这一节我们要实现镜面反射的效果，如此我们有两种材质，漫反射材质（[Lambertian](https://www.cnblogs.com/jerrycg/p/4941359.html)）和镜面反射材质（Metal），抽象出一个抽象基类material并令此二者继承自它，由于不同的材质会进行不同的反射行为，故定义了一个纯虚的反射函数scatter（根据后续的折射内容 其实应该是散射更准确）：

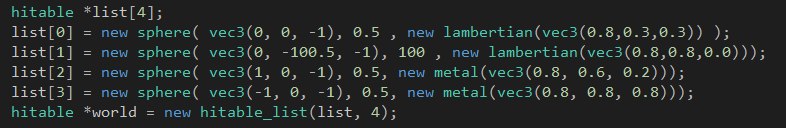


此函数的作用是：根据入射光线及光线命中检测的记录，获得交点的反射光线以及材质的反射率（衰减率），获得的此反射率用于返回col之前的系数；

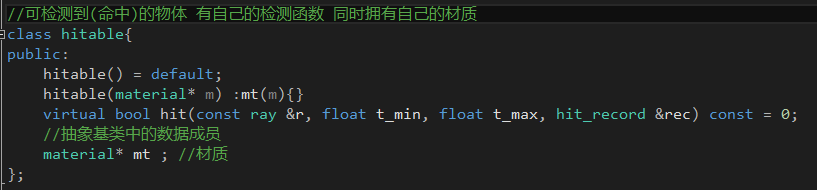
2、光线反射率（衰减率）attenuation，上一节我们的衰减率定义为一个常数50%，这一节这个衰减率的三个色相（RGB）都可以自定义，比如attenuation（0.3,0.5,0.8）意味着光线在该点的衰减度为：红色衰减为0.3倍，绿色衰减为0.5，蓝色衰减0.8；

3、scatter的返回结果是一个bool值，意味着反射是否成功，对于漫反射物体，因为反射是随机的，所以恒为true，~~对于镜面反射物体，有可能被吸收（折射），此时返回false~~（？），否则都返回true；attenutaion这个参数是外界传递过来的，我们要在方法中将材质的衰减率赋值给它，相当于material类的一个对象调用了一个成员函数，并且在这个函数中将自己的衰减率赋值给传递过来的attenuation参数（因为它是一个引用，也可以说是修改该点的反射率）；#也就是调用scatter方法，相当于在反射的过程中取得交点材质的折射率，赋值给参数attenuation；

4、代码解析以及重点注意：

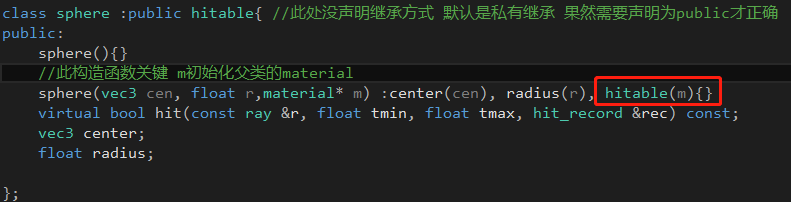


\*在主函数中，我们定义了可命中物体列表中有四个元素，并构造了四个球体，注意在这里sphere的构造函数中多了一个材质的指针，即new lambertian(vec3(xxx))；这正是书上遗漏的部分，我们根据自己的理解补全代码内容；根据如上所说，我们在hitable抽象基类中定义一个材质类指针material\* mt:



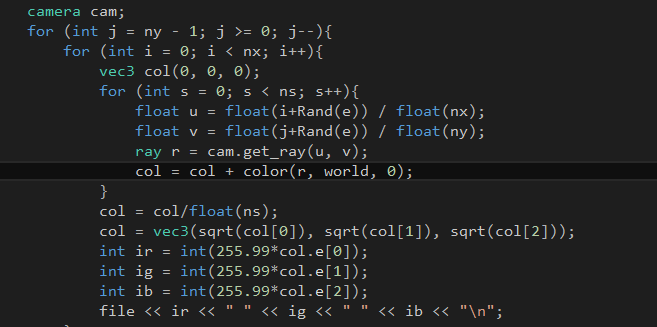
使得每个被命中的物体都有自己的材质；并且要书写默认构造函数，以及构造函数；

于是在hitable的派生类sphere当中，我们要把父类的material指针列表初始化：

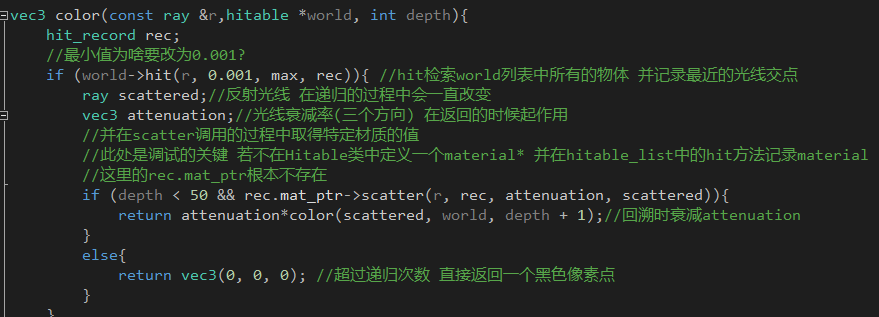


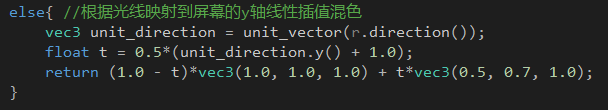
这里正说明了父类定义默认构造函数和构造函数的意义；于是如主函数中所示，每个球体构造的时候，拥有了自己的球心位置、半径以及特有的材质，并且最后一个material\*指针参数，我们是用具体的派生类指针赋值给它的，如lambertian或者metal，此两类的构造函数都是通过一个vec3赋值给albedo反射率构造；

5、最后来理解主函数的逻辑：



循环部分仍不变，对于某一条特定的光线，调用color（r,world,0）函数获得颜色，并进行抗锯齿处理；color函数如下：





如上面的注释，调用world->hit方法，在物体列表中计算是否与该条光线相交，若不相交直接返回背景色（y轴蓝白混合）；若相交，找到最近的交点，并从该点开始反射，定义了scatterd和attenuation，通过调用交点材质的->scatter（）方法获得（修改）,并进行一次光线衰减，然后再从这个点开始作为光线的起点，用获得的反射光线scattered作为新的入射光线，再进行下一次的反射迭代，直到没有物体可以反射光线，或者迭代反射超过递归深度50次为止，如果超过50次（~~或者光线被折射~~），直接返回vec3(0,0,0)（黑色）;

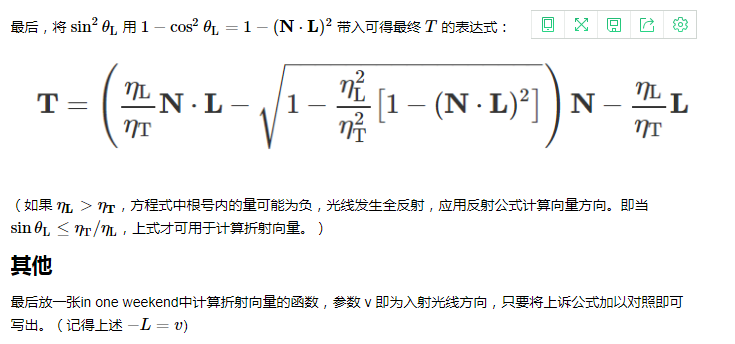
感观理解：如果是光滑的球面镜子，它会不断的反射多次，直到某一次反射的光线没有触到其他物体，那么此时该条光线计算出的颜色值相当于背景的蓝白混色（下面的else部分，递归边界），然后开始回溯，衰减多次直到到最初的color像素点，就形成了透明的样子，跟背景色相似；

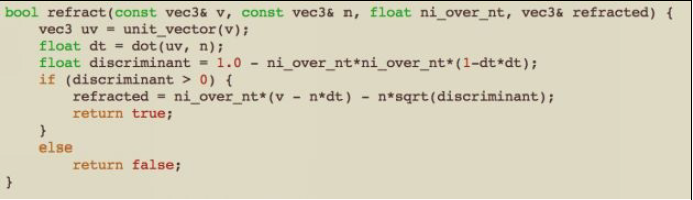
6、在matal材质中引入了一个模糊系数，即镜面反射可能会因此产生一定的模糊效果，反射光线进行一个随机的偏移；

Chapter 9: Dielectrics

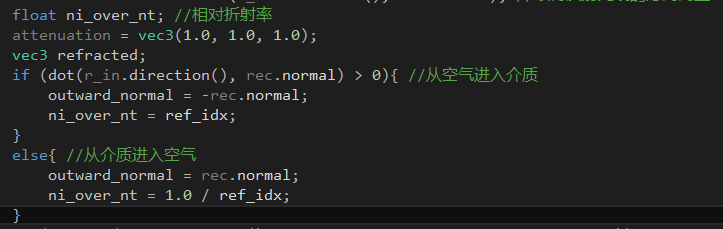
1、这节对电解质的材质实现折射效果，snell折射率公式见推导：

<https://www.cnblogs.com/theWhisper/p/10269574.html>

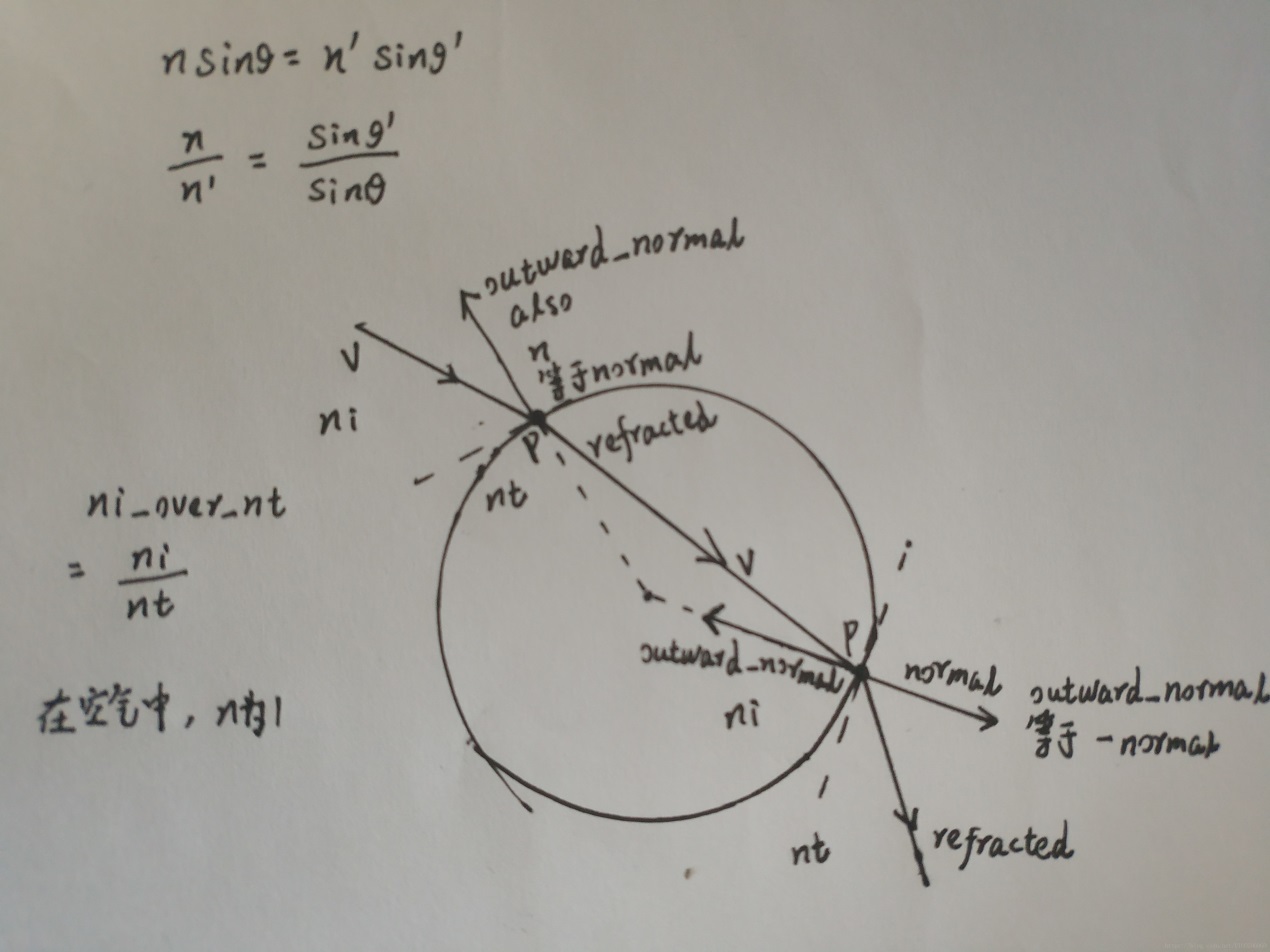




2、对于代码部分的理解：



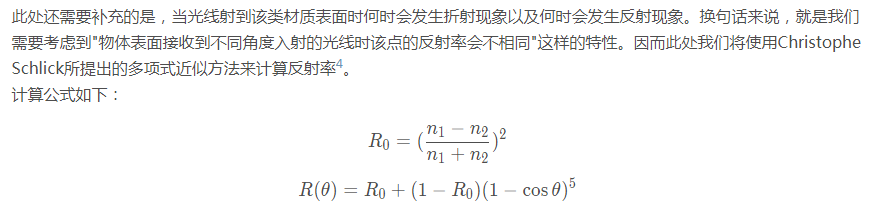
注：此处写反了，if里是介质进入空气，else是空气进入介质；



（自己也画了一个图）由此可见，当光线从介质中进入空气时，交点的法向量是向外的，即跟（新的）入射光线的方向一致，故他们的点积>0，所以此时的平面法向量应该取法向量的负向，并且相对折射率去介质折射率的倒数；

3、反射系数和反射率的概念，反射系数（reflection coefficient）是指光（入射光）投向物体时，其表面反射光的强度与入射光的强度之比值。受入射光的投射角度、强度、波长、物体表面材料的性质以及反射光的测量角度等因素影响。反射系数的平方称为反射率。例如我们面对一面玻璃窗时，以垂直角度去看就会像一面镜子；

反射率公式采用一个近似公式：



n1和n2是两种介质的折射率,θ是入射角；一些不理解之处可借助：

<https://blog.csdn.net/libing_zeng/article/details/54428732>

Chapter 10: Positionable camera

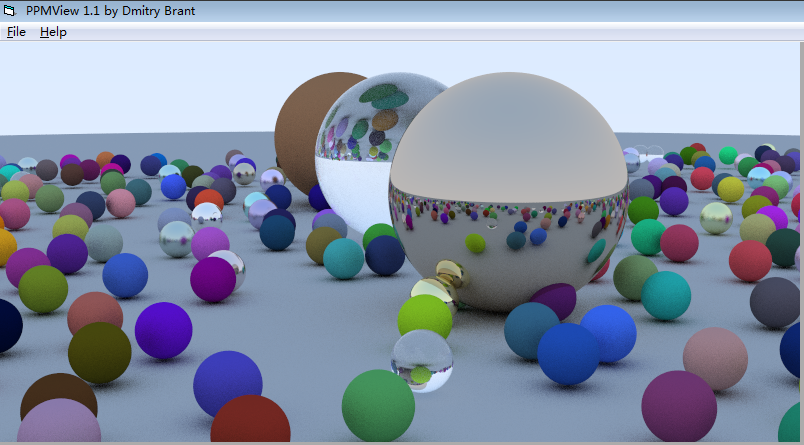
这一节设置相机的位置和视线，涉及一些数学推导，直接见笔记：

<https://www.cnblogs.com/lv-anchoret/p/10221058.html>

Chapter 11: Defocus Blur

散焦模糊/景深；

完结撒花~~！



额外补充：

