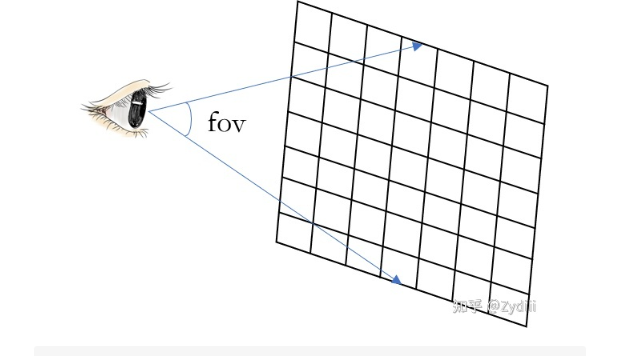
我们从眼睛发出一条射线，去和球体求交，如果相交，那么得到的颜色为球体的颜色，如果不相交，那么得到的颜色为背景的颜色。

我们假定这个代替我们眼睛观察虚拟世界的摄像机正对着屏幕中心，并且位于原点（0, 0, 0）。同时它有一个参数，为视野 fov，它用来表示我们在屏幕上可以看到的空间范围，在这里我们用它表示摄像机的俯仰角。

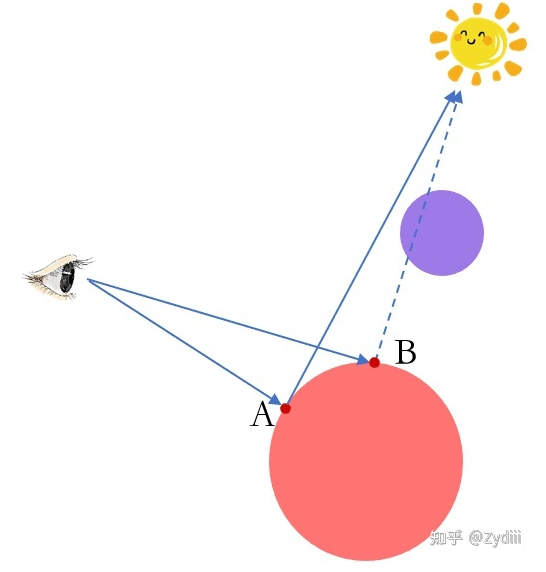


所以具体实现：

基础：！！！！！！coreeeeee

**所以光线追踪就是从我们的眼睛向每个像素发出射线【cast\_ray函数】，判断这些射线是否和球体相交【scene\_intersect(里面单个球用Sphere中的ray\_intersect)】，得到像素的颜色（reader也就是用光照模型可以得出颜色）。**

加上阴影：

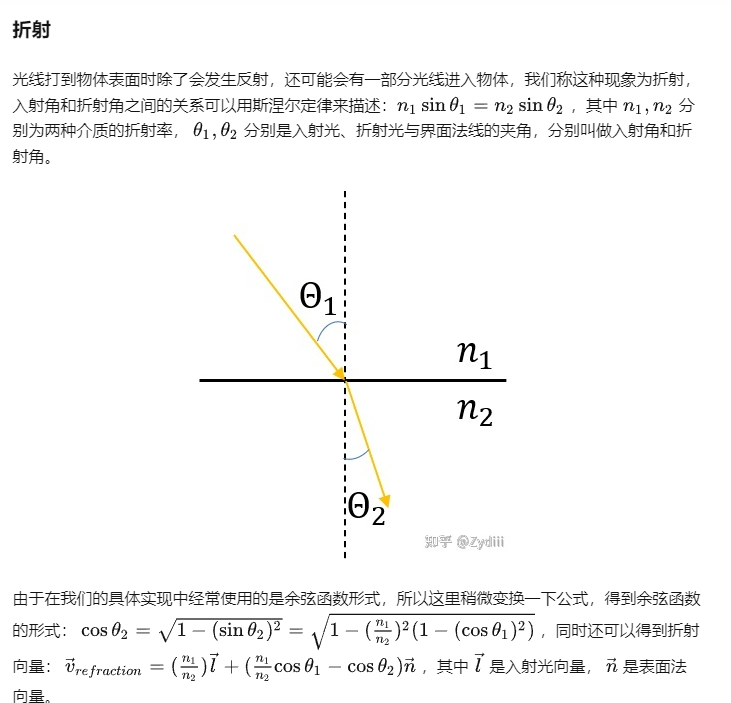
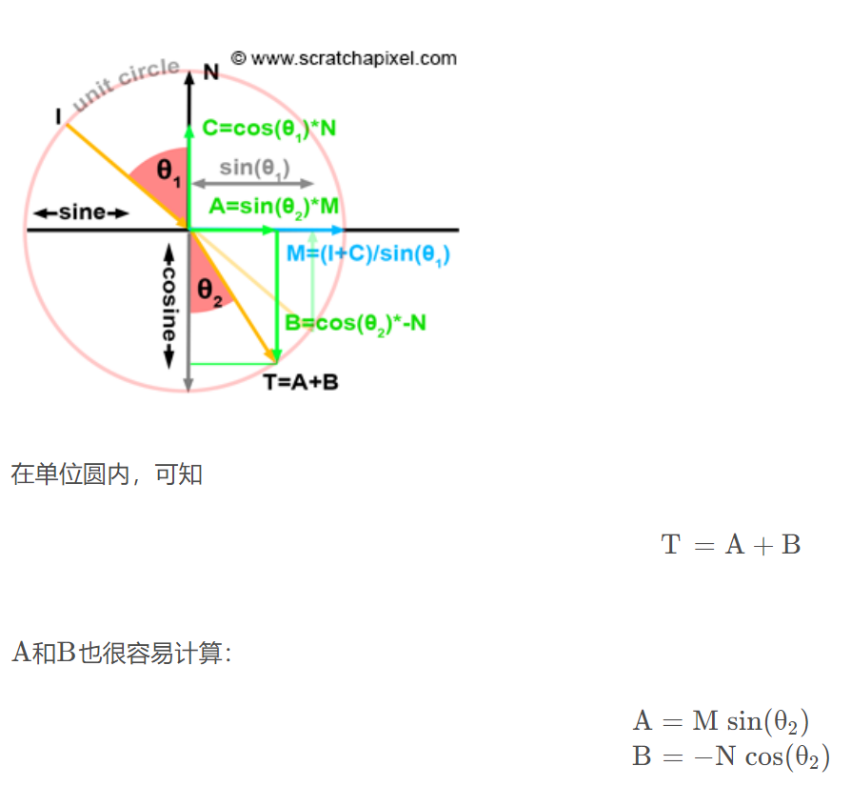


只需要在计算正常的光照之前，**先检查交点与光源之间的向量是否与场景中的物体相交，并且检查其与光源之间的距离是否大于其与其他相交物体之间的距离。**在这里有一个细节需要考虑，如果我们直接使用球体上的交点，那么我们始终会算出来交点与光源之间的向量与自己相交，这样是不对的，所以我们通过将交点沿着法线方向移动一段很小的距离避免这种情况。

# 反射：

光线追踪的另一个核心**：假定光线会在场景中的物体表面进行多次弹射，在计算某一个点的颜色时，我们只需要累加光线每次弹射产生的光照贡献即可（为了控制光线的弹射次数，我们添加一个深度变量 depth，当深度超过预设值以后光线便不再弹射。）**

在这里我们假设光线弹射的方向为镜面反射的方向，**当计算下一步的弹射光照时，我们只需要递归调用投射光线的函数即可，得到反射的颜色。**同样的，为了避免某物体表面弹射光线后与自己相交，我们也需要把交点沿着法线方向移动一小段距离。

折射：斯涅尔定律公式

