### 10.1.3反射

Key Note:

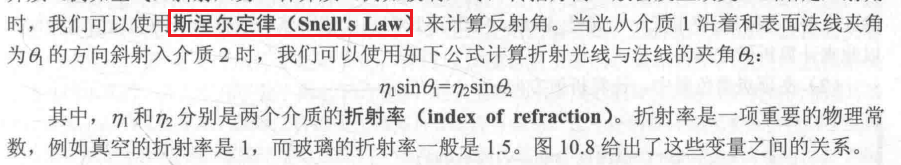
1.reflect(-o.worldViewDir, o.worldNormal); （vertex shader）

2.texCUBE采样cubeMap，而且传方向即可，不需要归一化

3.lerp混合。

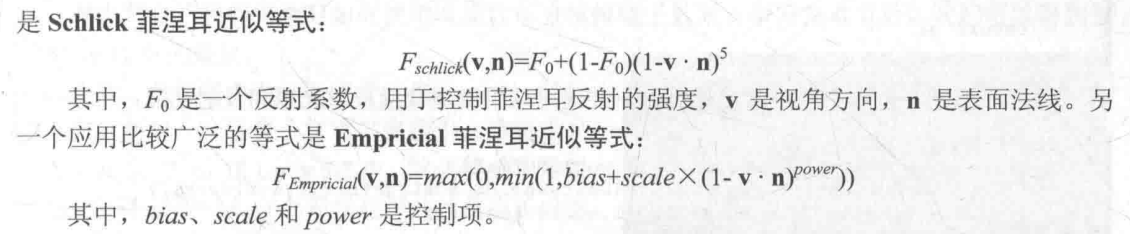
### 10.1.4 折射

折射公式



1. refract(-normalize(o.worldViewDir), normalize(o.worldNormal), \_RefractRatio);
2. texCUBE(\_Cubemap, i.worldRefr).rgb \* \_RefractColor.rgb;

### 10.1.5菲涅儿



lerp(diffuse, reflection, saturate(fresnel)) \* atten

### 10.2.2 玻璃效果

通常使用GrabPass来实现诸如玻璃等透明材质的模拟，和简单的透明混合不同，GrabPass可以对物体后面的图像进行更复杂的处理，如法线模拟折射。

GrabPass通常于渲染透明物体。用GrabPass要小心物体的渲染队列设置。尽管代码不包含混合指令，但是选需要把渲染队列设置为渲染队列（“Queue”=”Transparent”）。这样才能确保渲染该物体时，所有的不透明物体已经绘制在屏幕。

GrabPass支持两种形式。

GrabPass{}，然后在后续的Pass中直接使用\_GrabTexture 每个物体的抓取屏幕都会导致重新执行抓取操作，消耗大。

GrabPass{“TextureName”}，每个物体的抓取操作都是抓取同一个Texture，消耗较低。

渲染纹理 vs GrabPass

GrabPass：

好实现、图像分辨率和屏幕一样(可能有带宽问题)、移动设备上不会重新渲染场景，但是往往需要cpu直接读取后备缓冲(back buffer)中的数据，破坏cpu和gpu之间的并行性。

渲染纹理:

需要把部分场景重新渲染以便，但可以通过相机剔除层。

Command Buffers