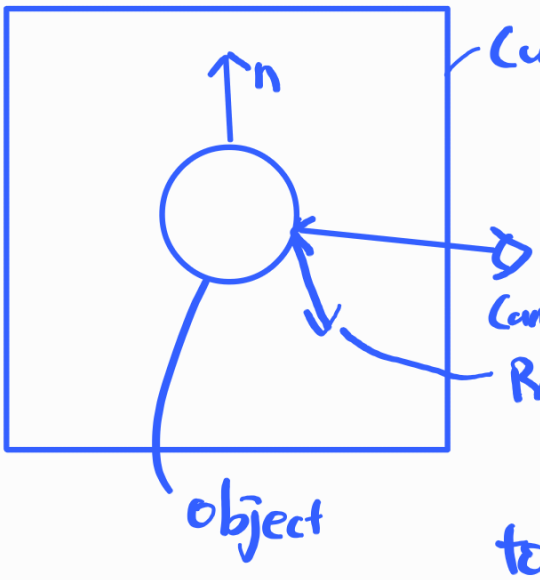
Environment Mapping (환경매핑)

Reflection(반사선 반사)을 이용해 Pixel Shader에서
물체에 반사된 빛을 가져온다.



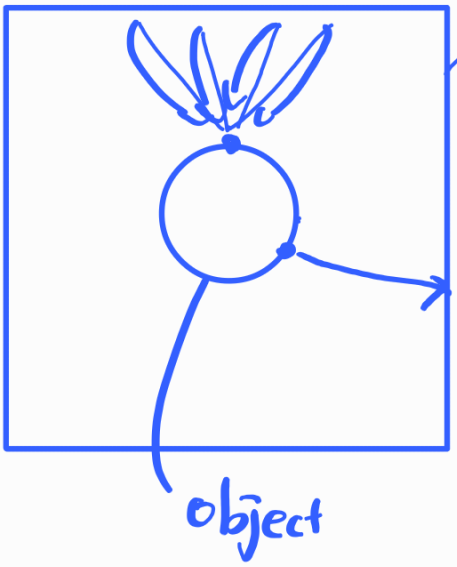
Normal Vector와 광선이 들어오는 방향(시점이 있는 방향)으로 Reflection Vector를 구한다.

이 반사벡터를 각 Pixel이 주변 환경에 어떤 색을 가져와야 하는지 알 수 있다.

```
toEye = normalize(eyeWorld - input.posWorld);
EnvironmentColor = g_textureCube.Sample(g_sampler,
    reflect(-toEye, input.normalWorld));
```

Image based lighting (이미지 기반 라이팅)

directional light, Point light, Spot light 3가지 조명을 아무리 잘 조합해도 현실의 복잡한 조명상황은 만들기 매우 어렵다.
그래서 Texture를 아주 잘 사용하는 방법으로 발전하고 있다
IBL은 환경 매핑을 조명처럼 사용하는 방법이다



Shading하고자하는 지점이 빛을 얼마나 받을 수 있는지 가져올 수 있는 모든 곳에서 색을 가져와 평균을 내보기로 함.
(계산량이 많아 느리다)

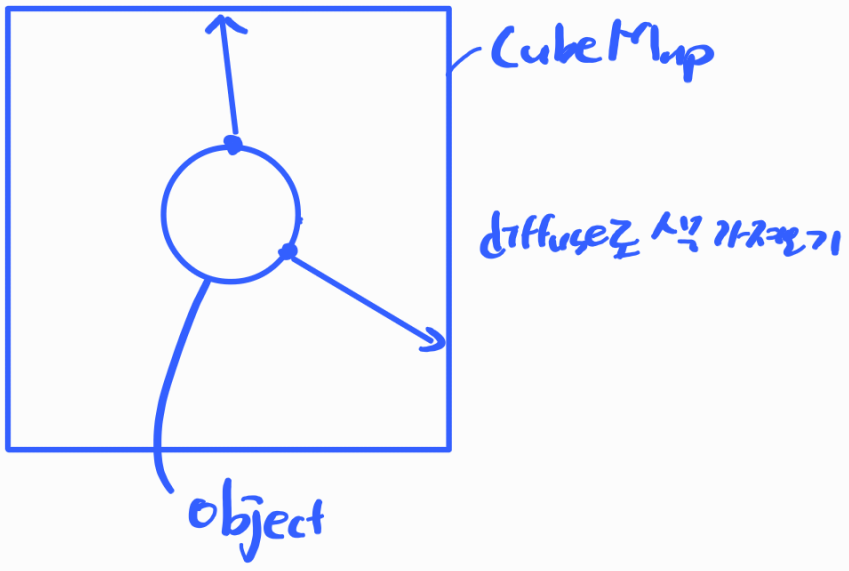
그래서 IBL에선 색을 한번만 가져와도 조명 효과를 제대로 적용가능하도록 Texture를 미리 처리한다

Diffuse Texture Image = 난반사 모델링한 Texture
(이미지를 매끄럽게 전처리)

Specular Texture Image = 선명한 이미지사용(금속, 거울 등)

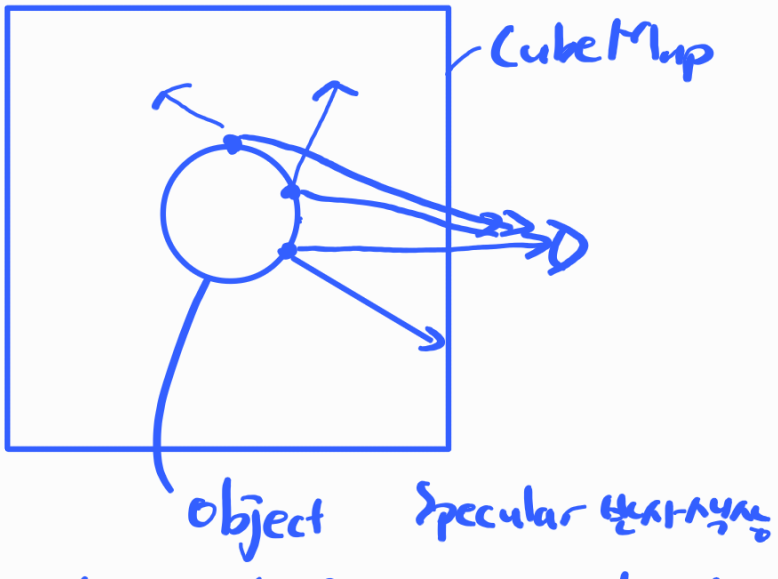
즉, 여러 효과로 전처리된 Texture Image들을 여러개 사용하는 것이다.

float4 diffuse = g_diffuseTexture(g_sampler, input.normalWorld);
diffuse x= material.diffuse;



float4 Specular = g_SpecularTexture(g_sampler,
reflect(-toEye, input.normalWorld);
Specular *= pow((specular.x + specular.y + specular.z) / 3.0,
material.shininess);

Specular x= material.specular;



color = diffuse + specular;