Image Based Lighting

Image Based Lighting

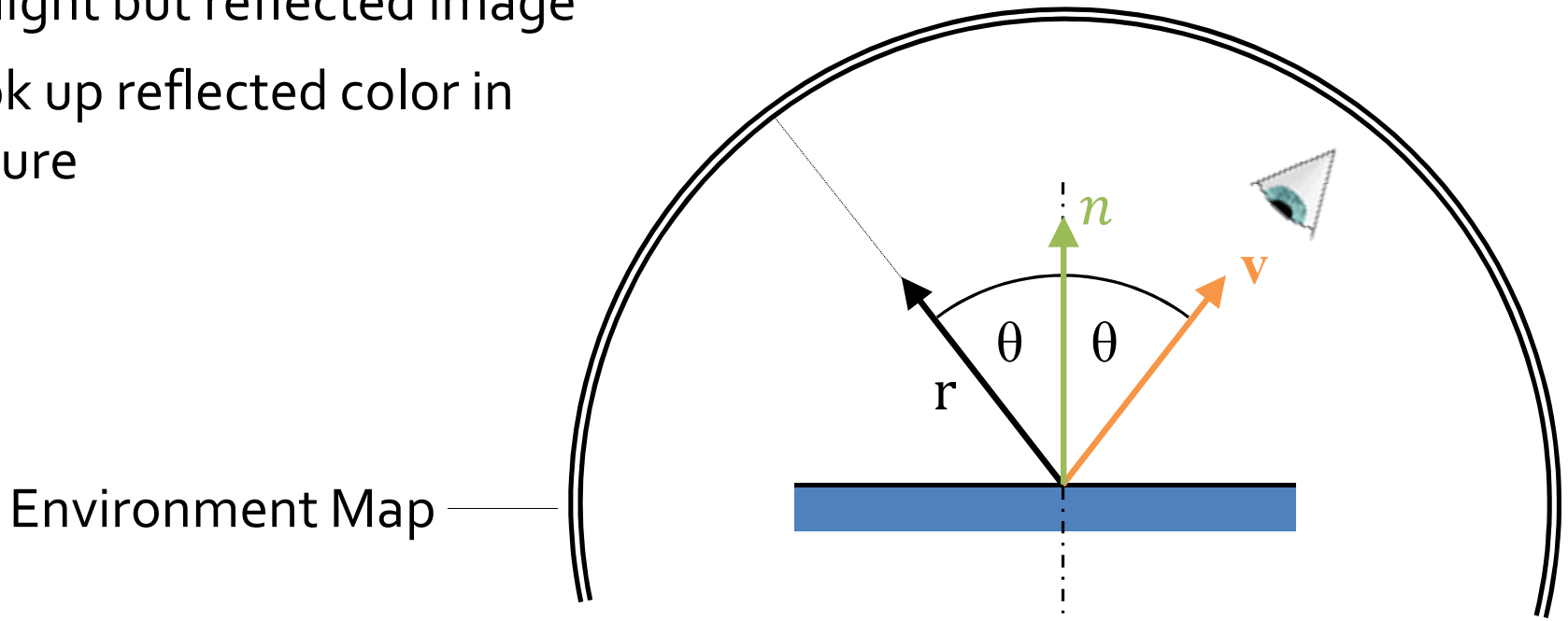
- Fake global lighting effects
- By using textures
- Textures can be
  - Precalculated or
  - Calculated at run-time

Image Based Lighting

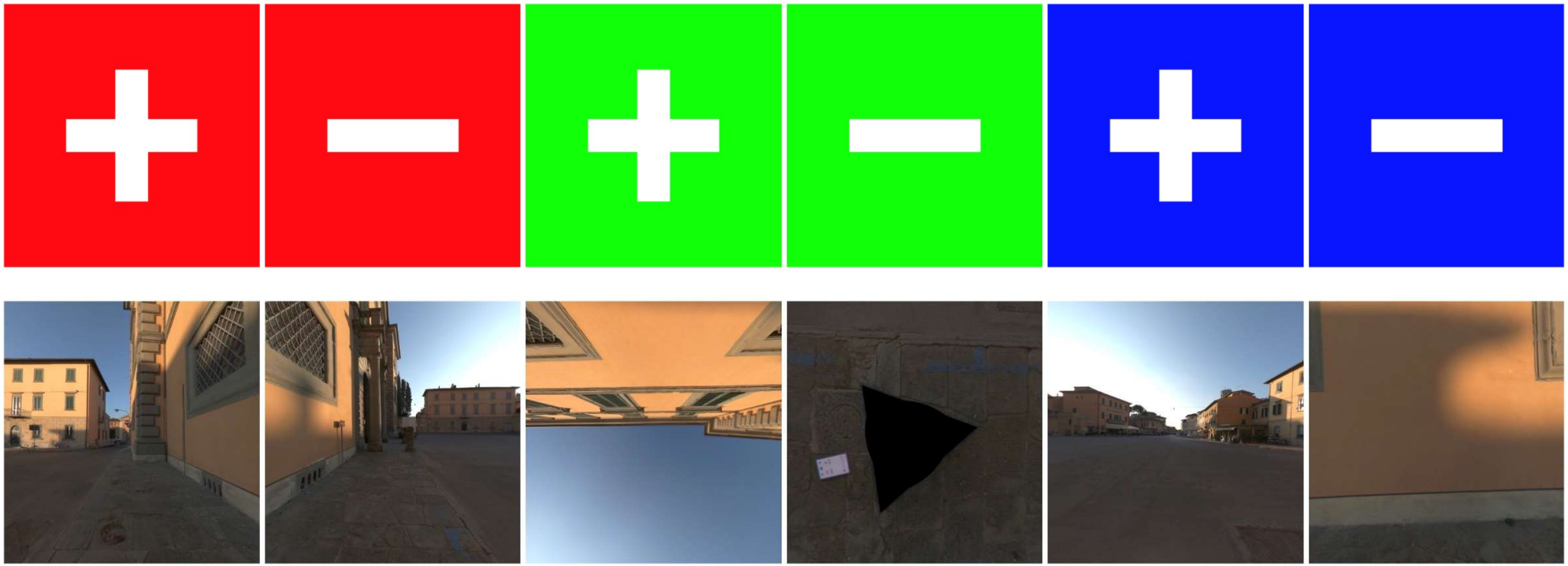
Environment Mapping – Fake Reflections

Environment Mapping

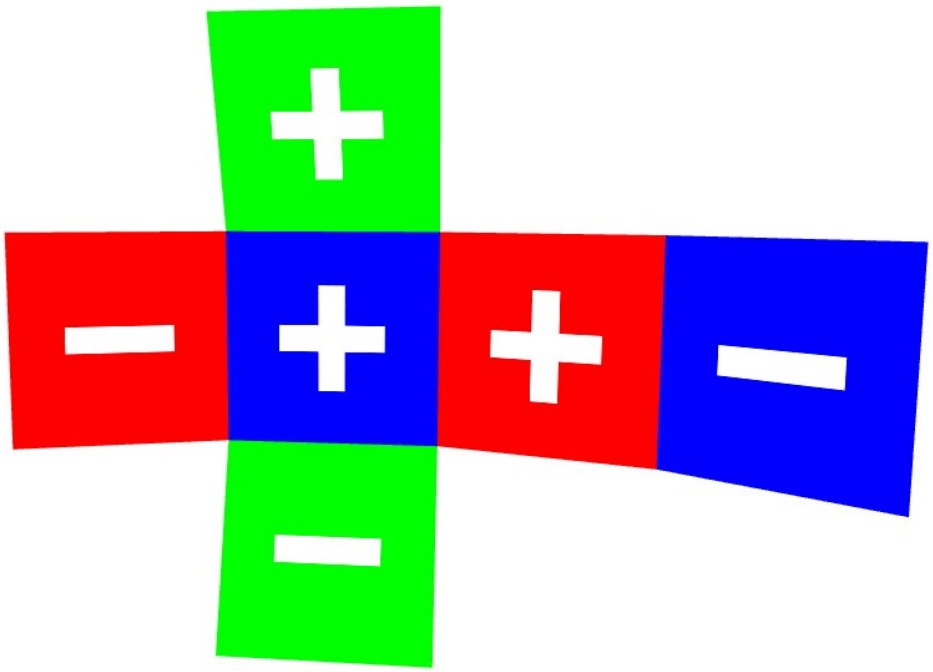
- Better specular reflections
- Not highlight but reflected image
- Idea: Look up reflected color in 360° texture



360° Texture Types – Cube map

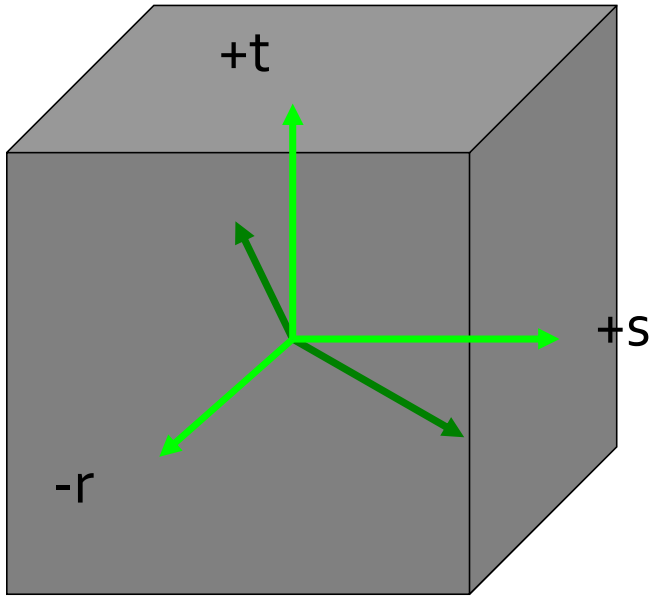


360° Texture Types – Cube map

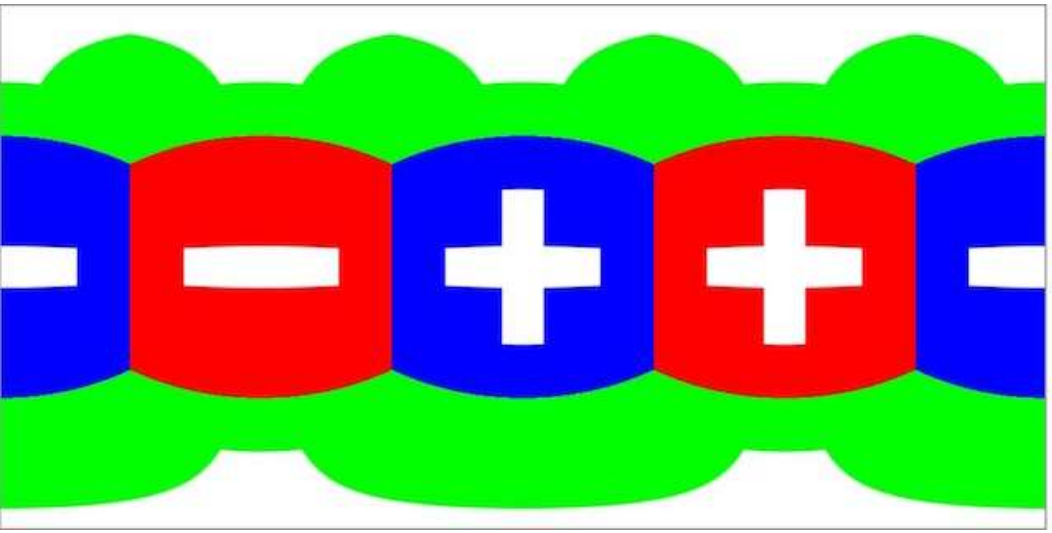


360° Texture Types – Cube map

- Reflected vector can be directly used in **`textureCube`** function
- (x,y,z) maps directly to (s,t,p)

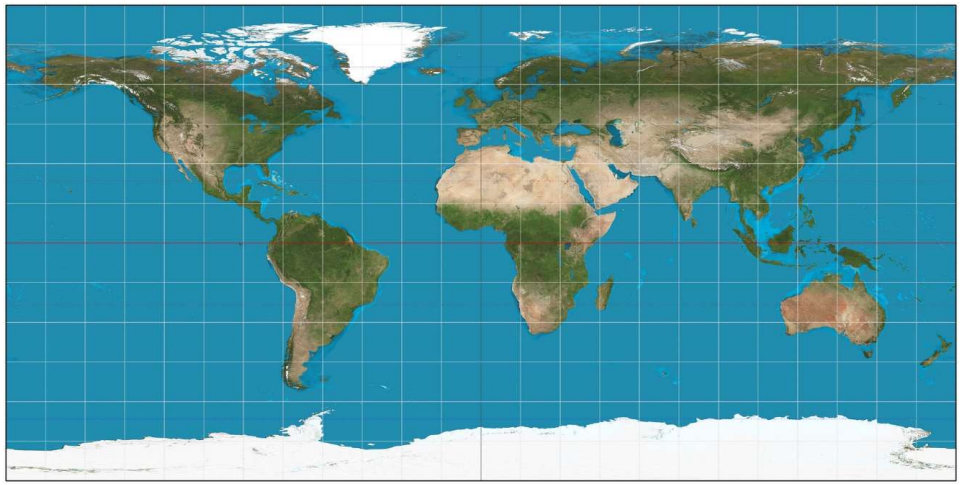


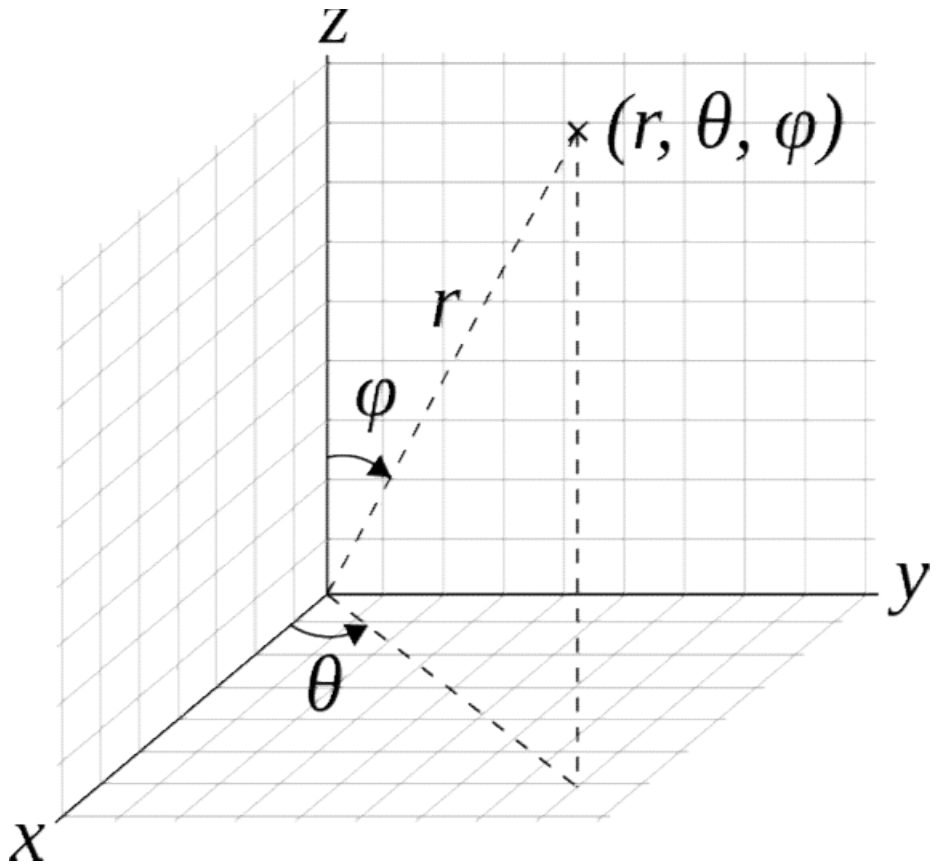
360° Texture Types – LongLat map



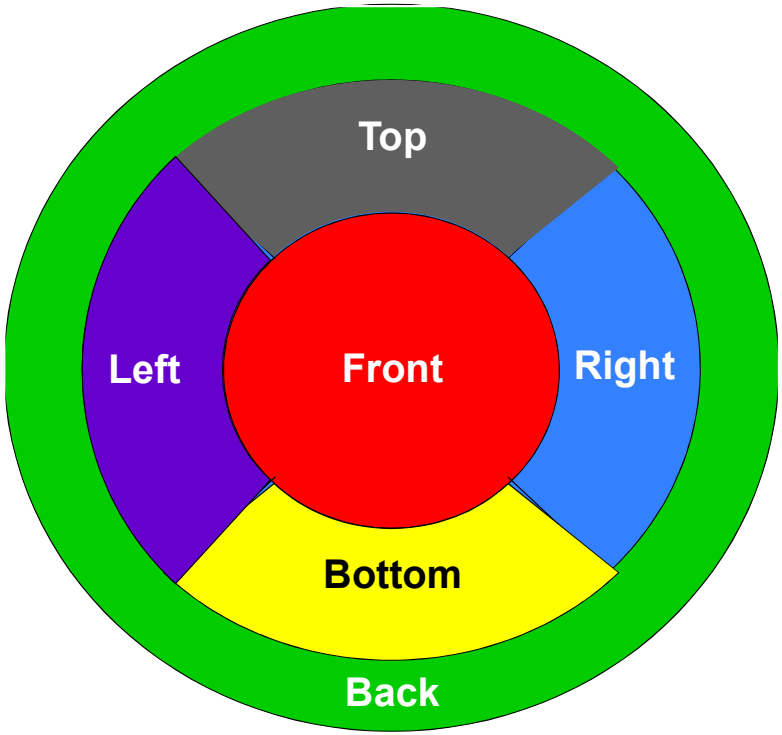
360° Texture Types – LongLat map

- Maps azimuth θ and elevation φ to s and t texture coordinates
- $\theta = \tan 2^{-1}(-x, z) + \pi$
- $\varphi = \cos^{-1} y$



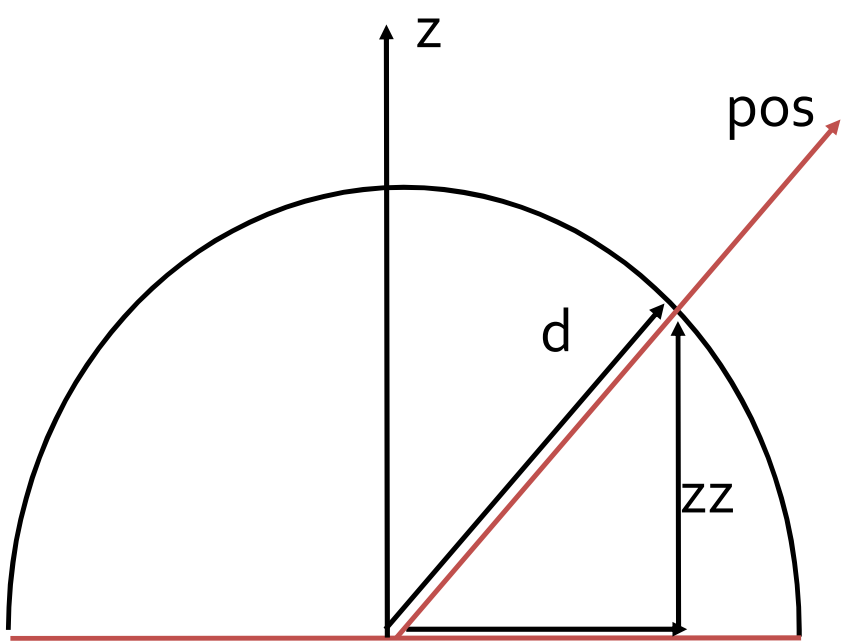


360° Texture Types – Sphere map

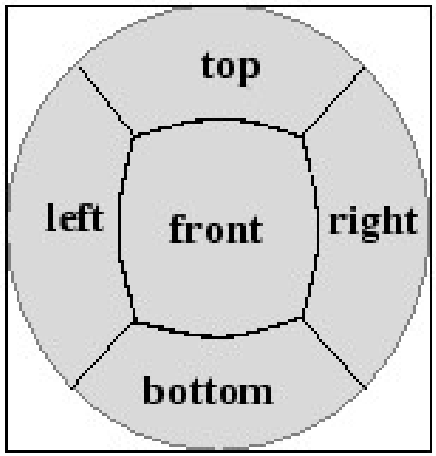


360° Texture Types – Sphere map

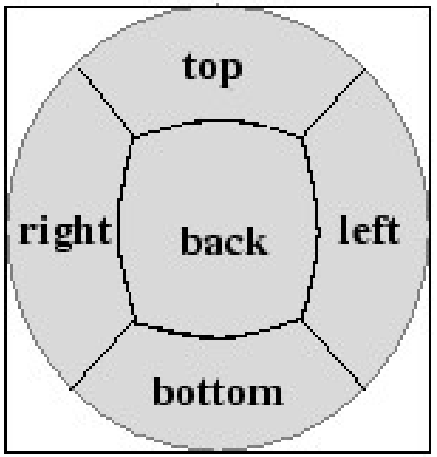
- Projection onto unit sphere
  ```
  normalize(vec3 pos).xy
  ```
- Back from sphere:
  ```
  float z = sqrt(
     1 - dot(d,d))
  return vec3(d.x,d.y,z)
  ```



360° Texture Types – Dual-paraboloid Map



front texture



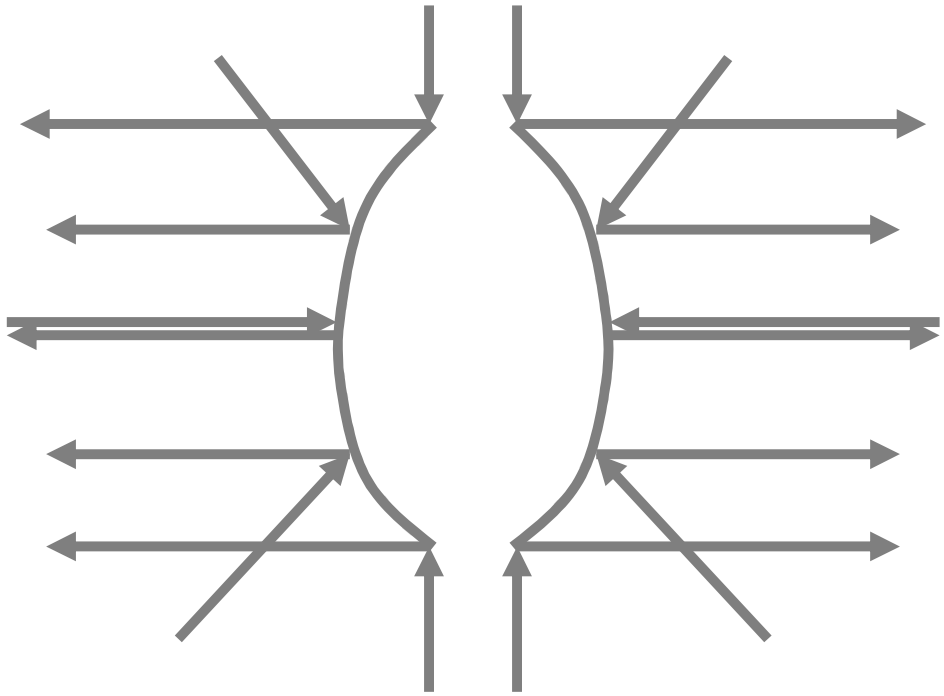
back texture

360° Texture Types – Dual-paraboloid Map

- Projection onto parabola
  `pos.xy/(pos.z - 1)`
- Back from parabola:

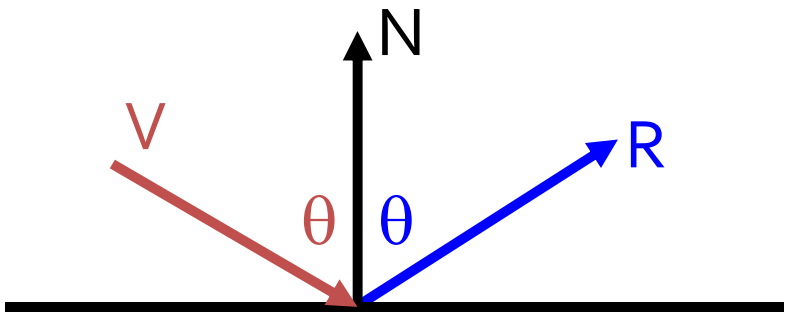
```
float z = 0.5
   - 0.5 * dot(d,d);
return vec3(d.x,d.y,z);
```

- Use orthographic reflection of two parabolic mirrors



Reflective Environment Mapping

- Angle of incidence = angle of reflection



R = V - 2 (N dot V) N
= `reflect(V,N)`

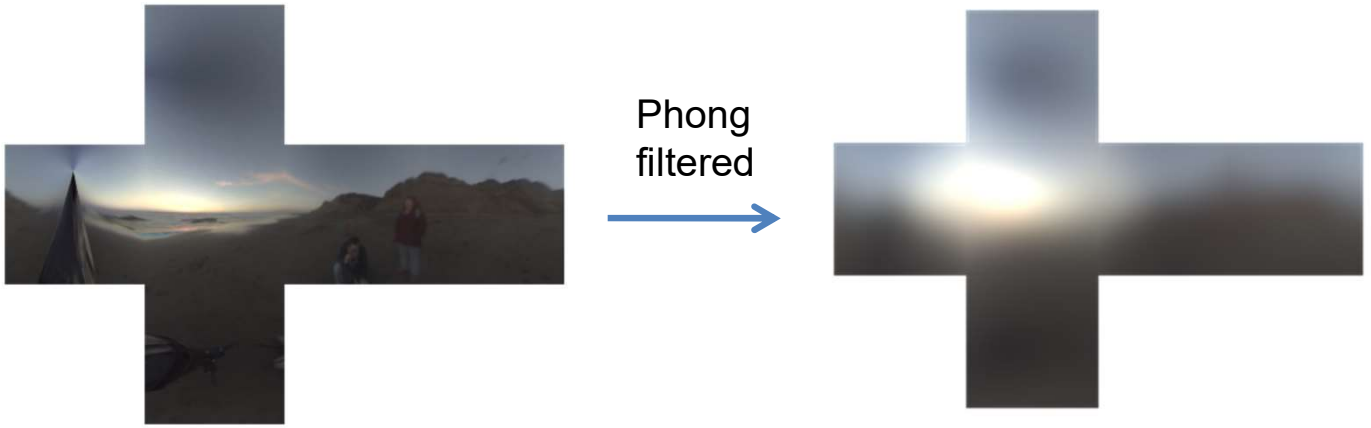
V and N normalized!

V is incident vector!

- Cube map needs reflection vector in coordinates (where map was created)

Specular Environment Mapping

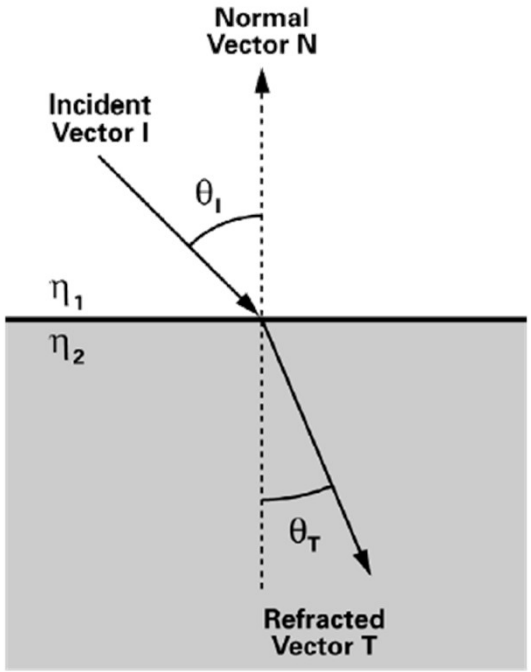
- We can pre-filter the environment map
  - Equals specular integration over the hemisphere
  - Phong lobe (cos^n) as filter kernel
  - **`textureLod`**;**`level`** according to glossiness
  - R as lookup

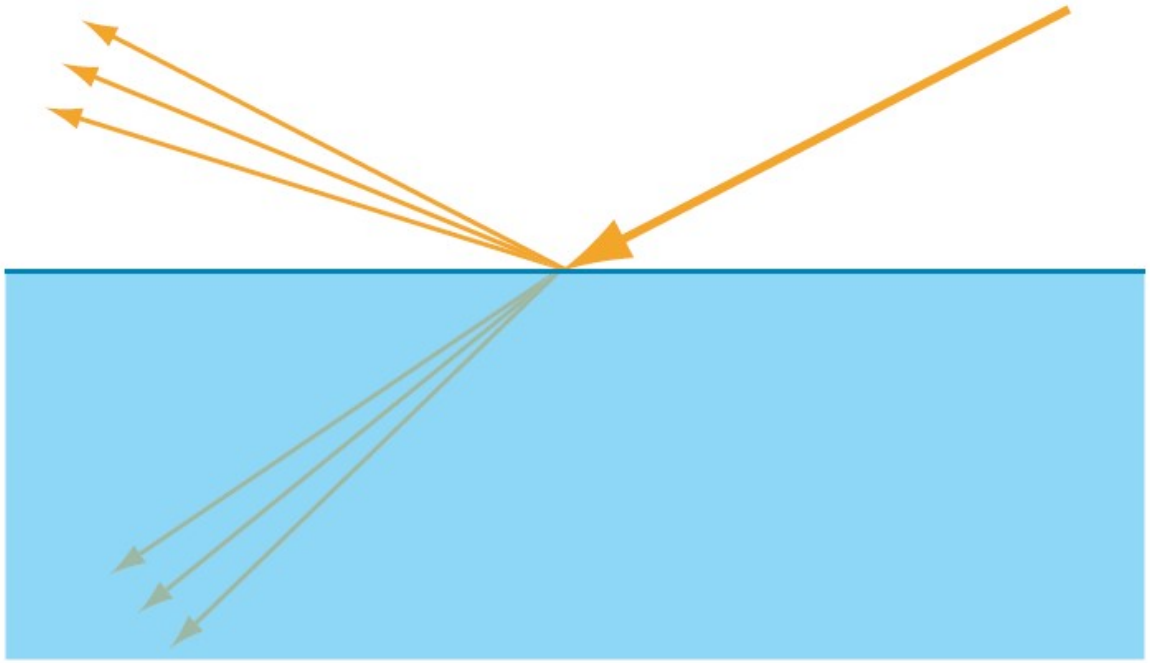


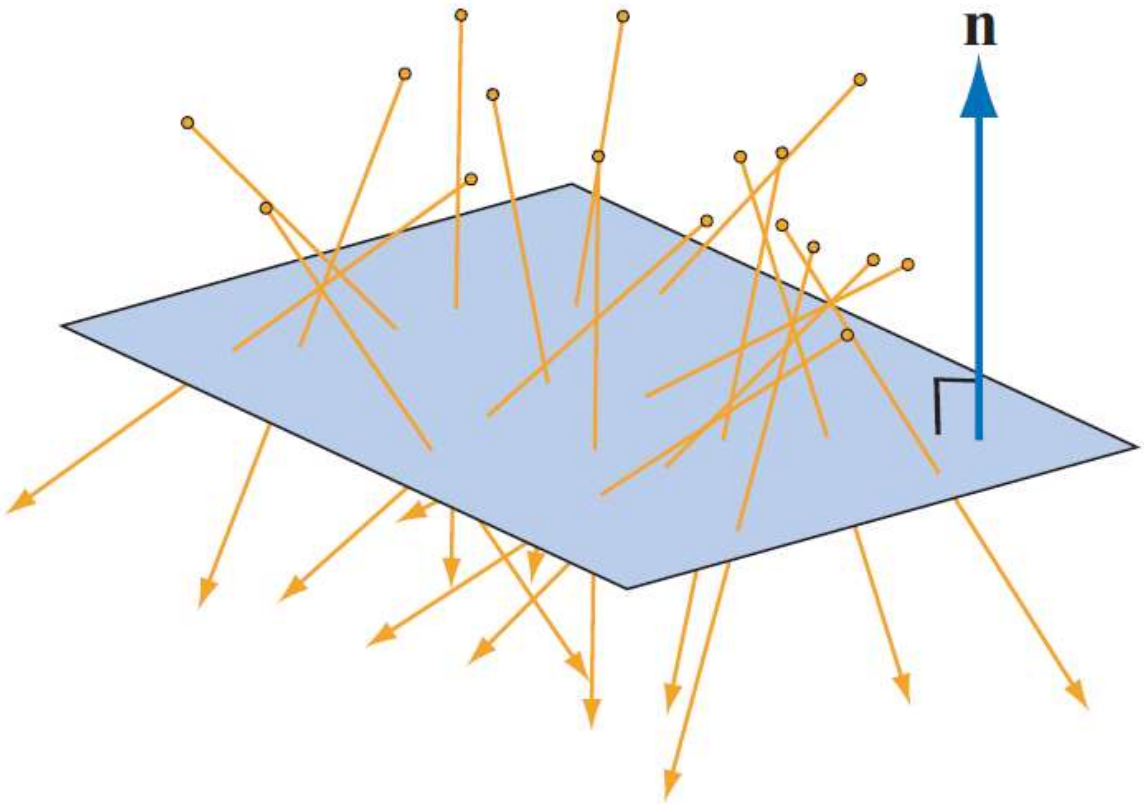
Refractive Environment Mapping

- Use refracted vector for lookup:
  - Snells law:

$$\eta_1 \sin\theta_I = \eta_2 \sin\theta_T$$






n