Computer Graphics (CS 4731)
Environment Mapping

Joshua Cuneo

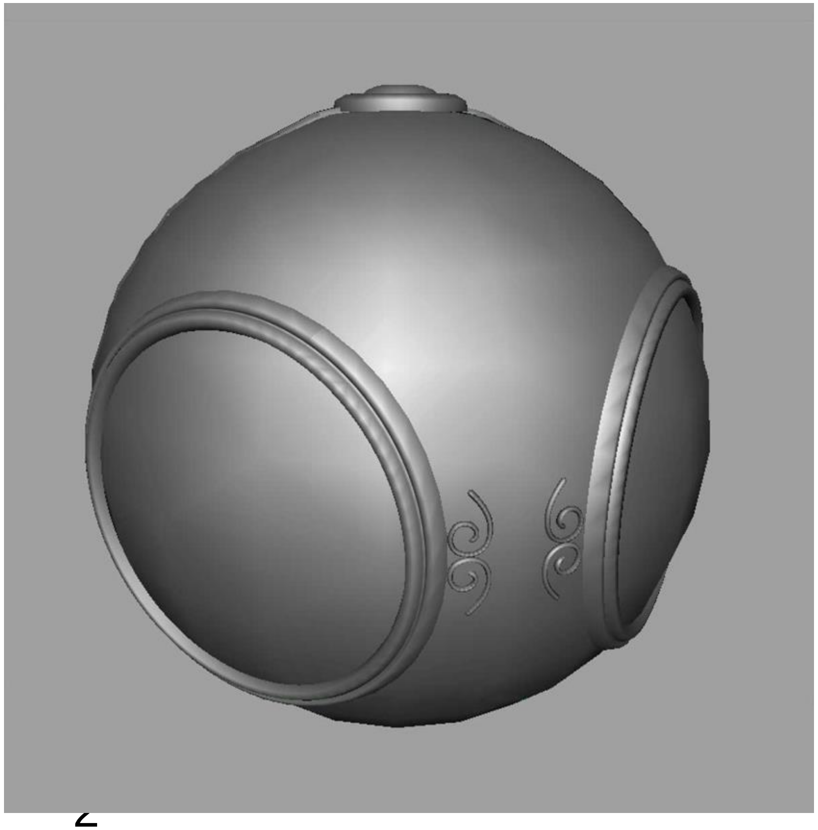
Computer Science Dept.

Worcester Polytechnic Institute (WPI)

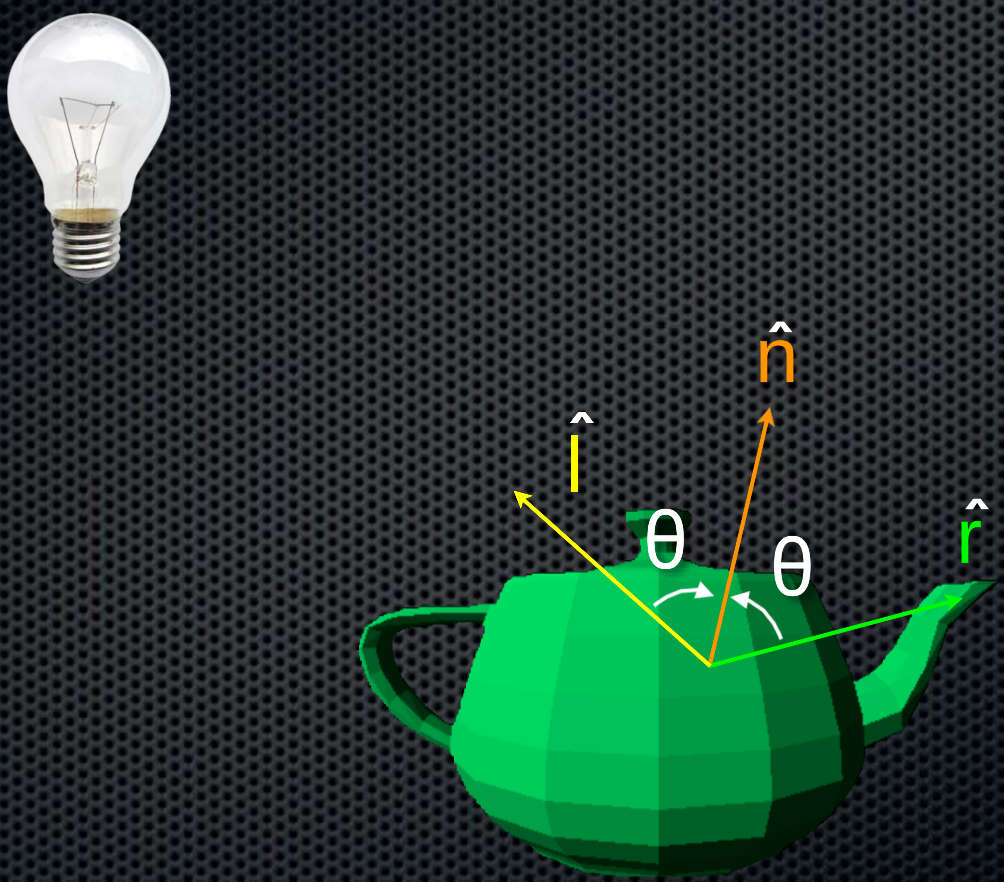
Some Slides Courtesy of Emmanuel Agu (WPI) and Matt Stoker (Utah)

Environment Mapping

- Environmental mapping is way to create the appearance of highly **reflective** and **refractive** surfaces without ray tracing



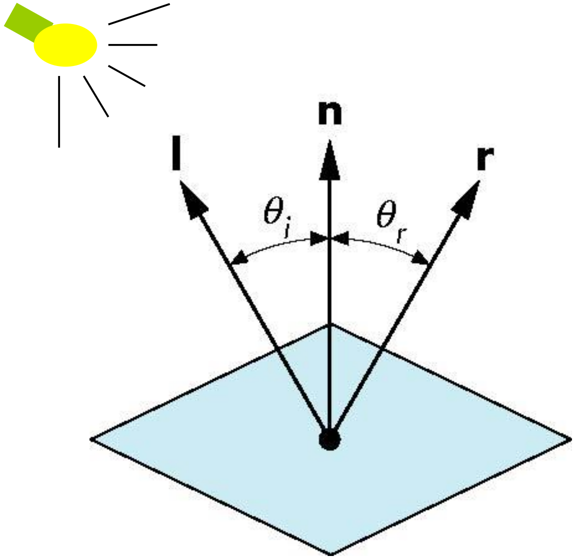
Specular Lighting



Mirror Direction?

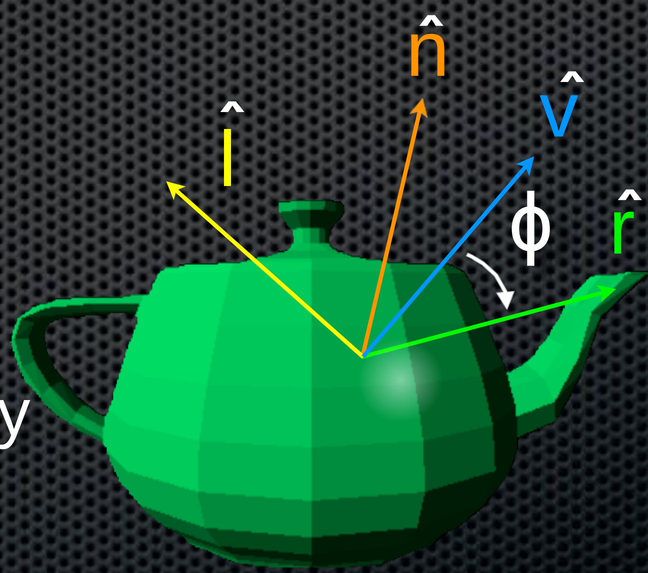
- Angle of reflection = angle of incidence

$$\mathbf{r} = 2\,(\mathbf{l} \cdot \mathbf{n})\,\mathbf{n} - \mathbf{l}$$



Specular Lighting

$$I_s \simeq L_s k_s (\hat{v} \cdot \hat{r})^{\alpha}$$



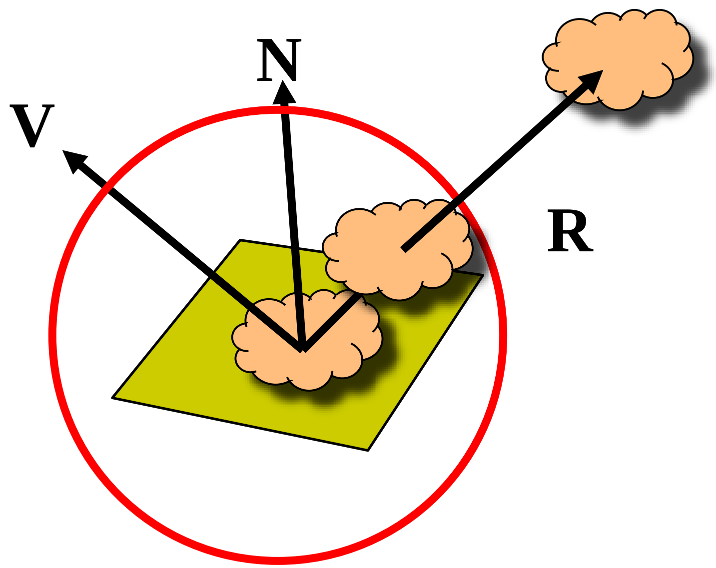
L_s = specular intensity

α = shininess

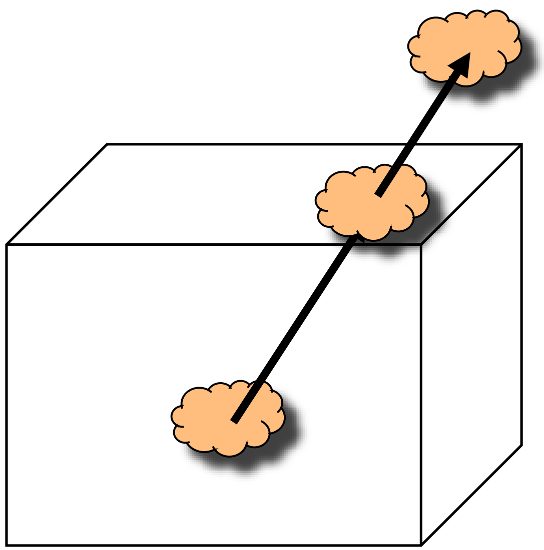
k_s = material specular coefficient

Environment Maps

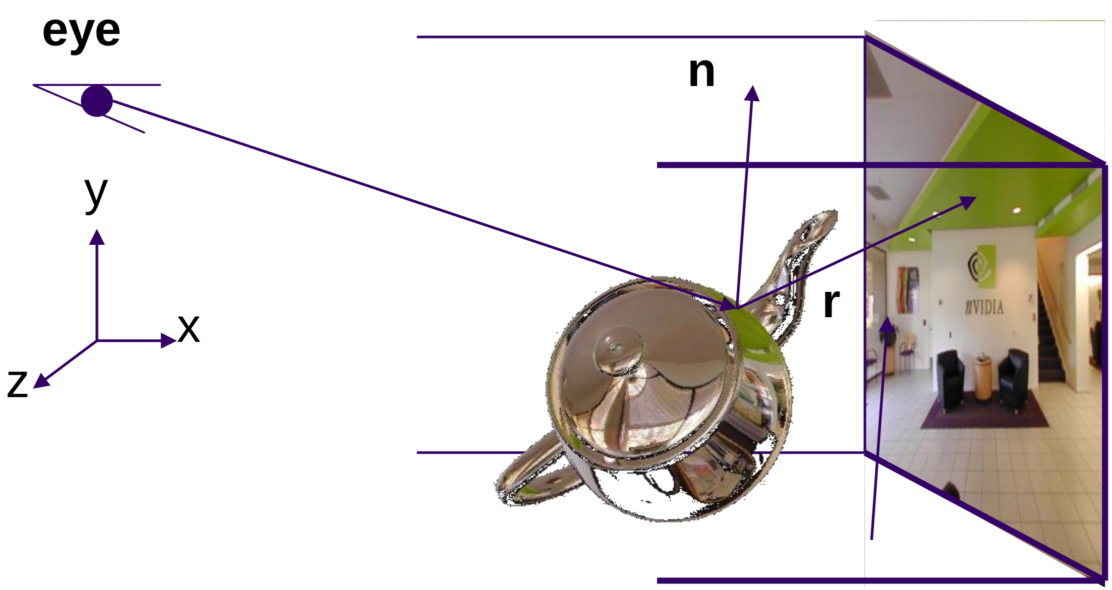
Sphere around object (sphere map)

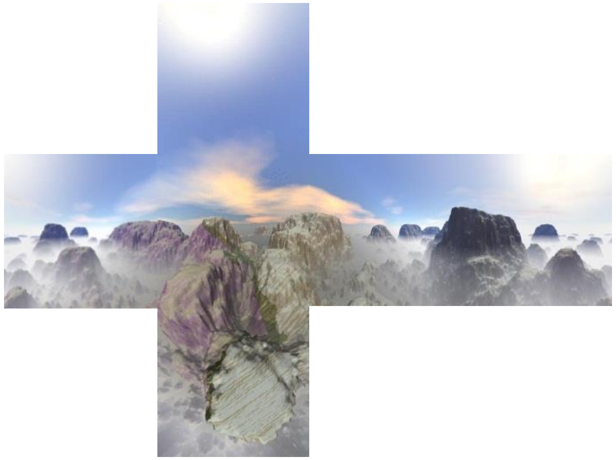


Cube around object (cube map)



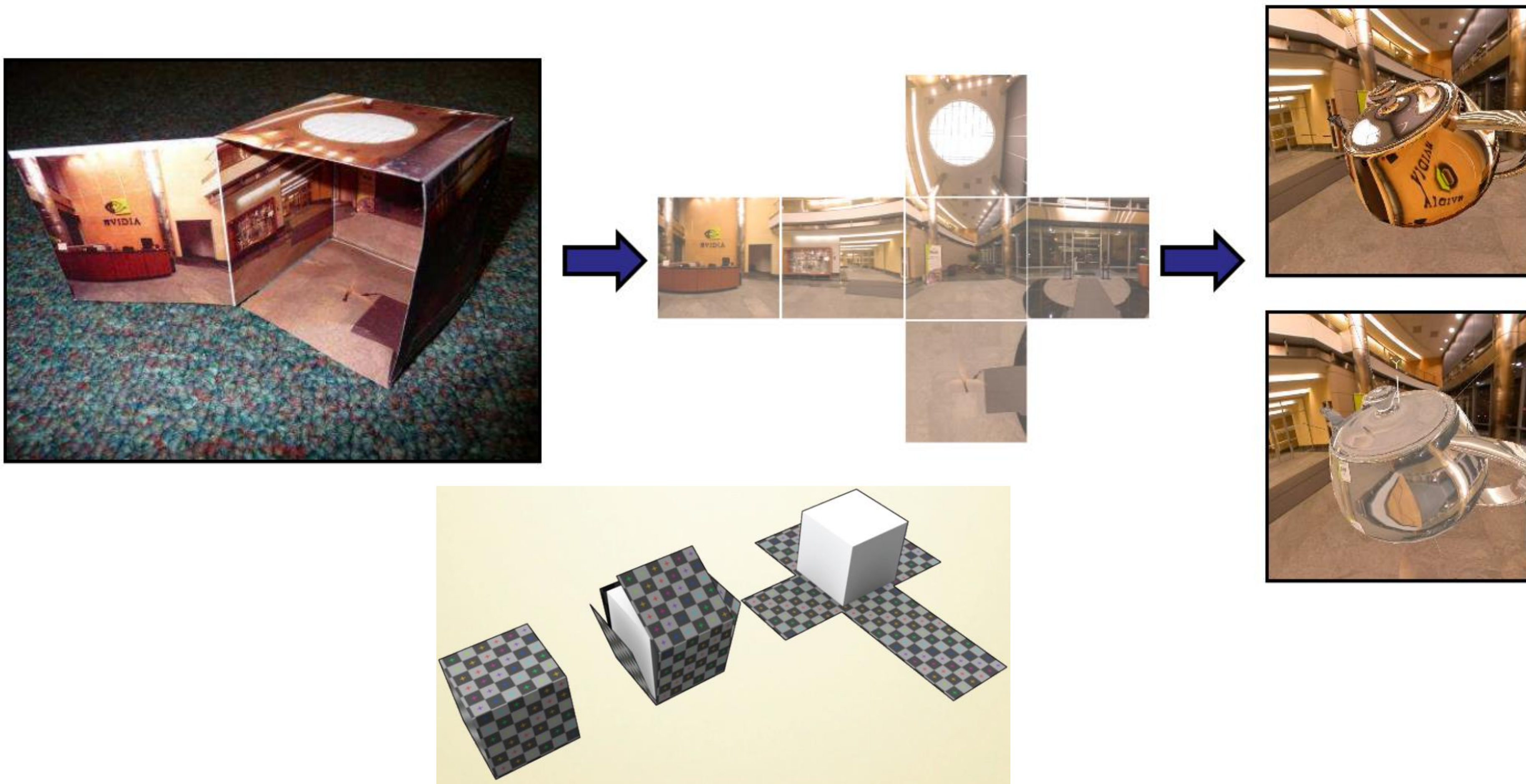
Cube mapping





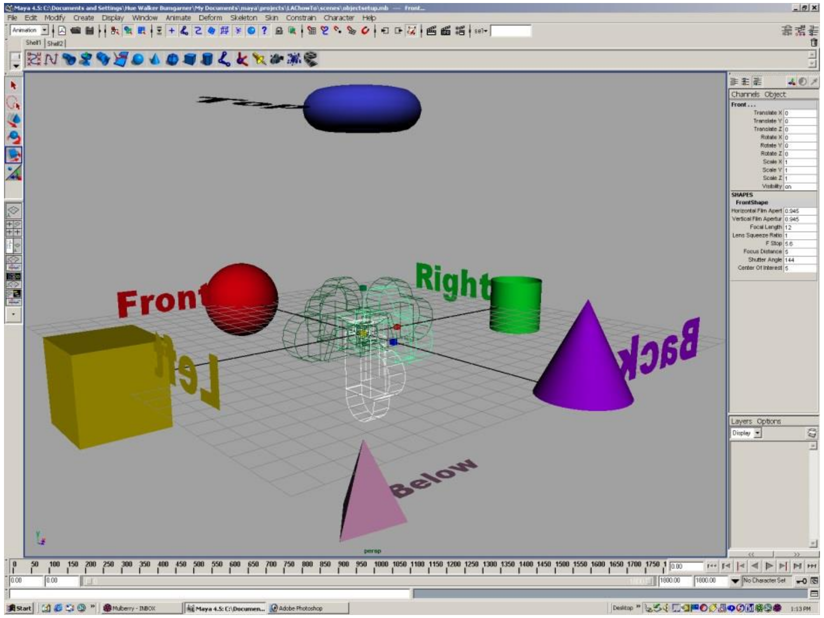
- Need to compute reflection vector, $\mathbf{r}$
- Use $\mathbf{r}$ by for environment map lookup

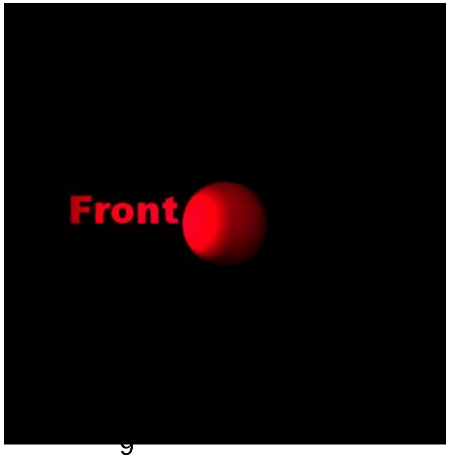
Cube mapping



Creating Cube Map

- Use 6 cameras directions from scene center
  - each with a 90 degree angle of view





Refraction using Cube Map

- Can also use cube map for refraction (transparent)

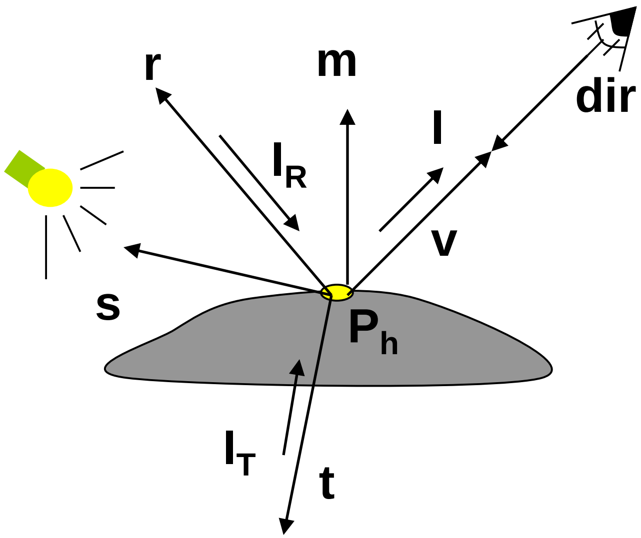
Reflection

Refraction

Reflection and Refraction

- At each vertex

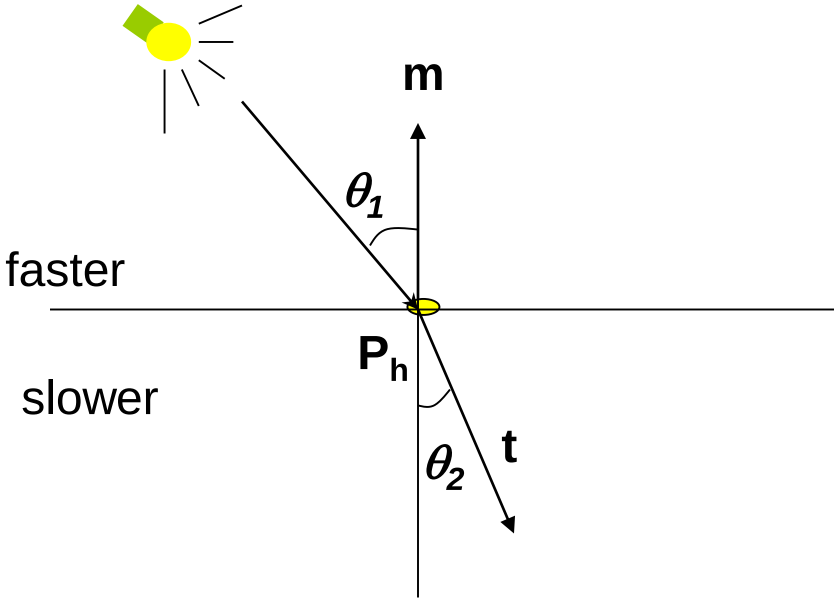
$$I = I_{amb} + I_{diff} + I_{spec} + I_{refl} + I_{tran}$$



- Refracted component I_T is along transmitted direction **t**

Finding Transmitted (Refracted) Direction

- Transmitted direction obeys **Snell's law**
- Snell's law: relationship holds in diagram below



$$\frac{\sin(\theta_2)}{c_2} = \frac{\sin(\theta_1)}{c_1}$$

c_1, c_2 are speeds of light in medium 1 and 2

Finding Transmitted Direction

- Some measured relative c1/c2 are
  - Air: 99.97%
  - Glass: 52.2% to 59%
  - Water: 75.19%
  - Sapphire: 56.50%
  - Diamond: 41.33%

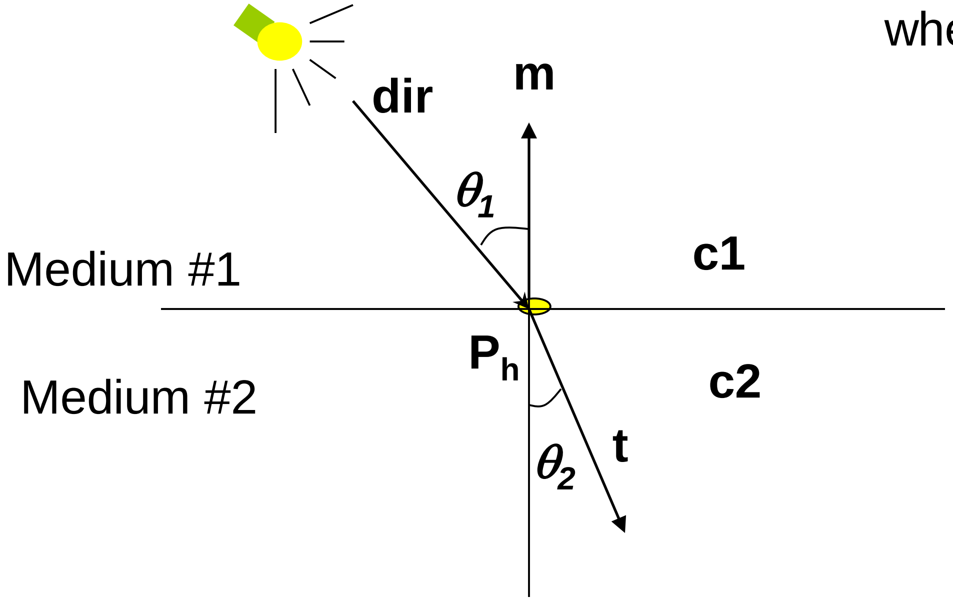
Transmission Angle

- Vector for transmission angle can be found as

$$\mathbf{t} = \frac{c_2}{c_1}\mathbf{dir} + \left(\frac{c_2}{c_1}(\mathbf{m} \bullet \mathbf{dir}) - \cos(\theta_2)\right)\mathbf{m}$$

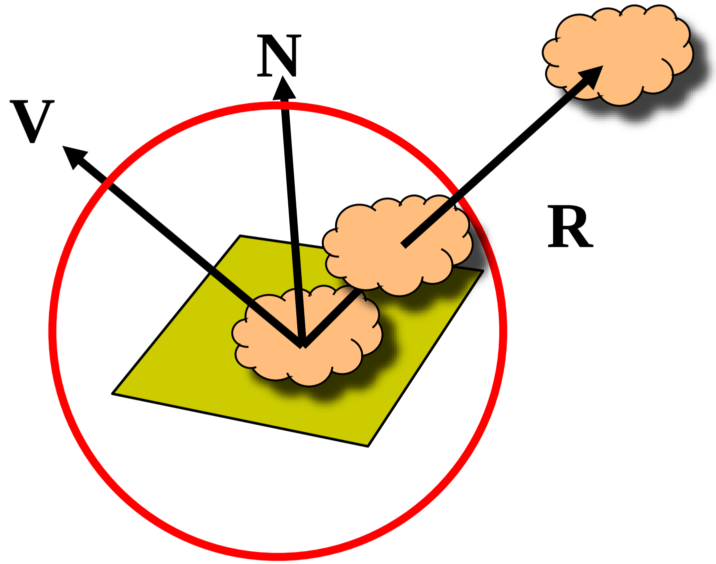
where

$$\cos(\theta_2) = \sqrt{1 - \left(\frac{c_2}{c_1}\right)\left(1 - (\mathbf{m} \bullet \mathbf{dir})^2\right)}$$

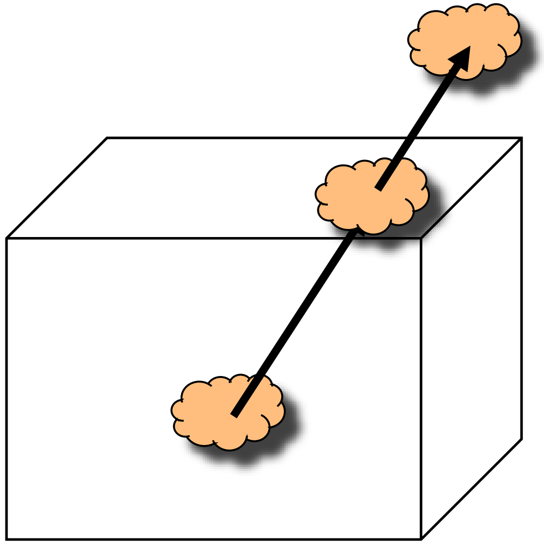


Environment Maps

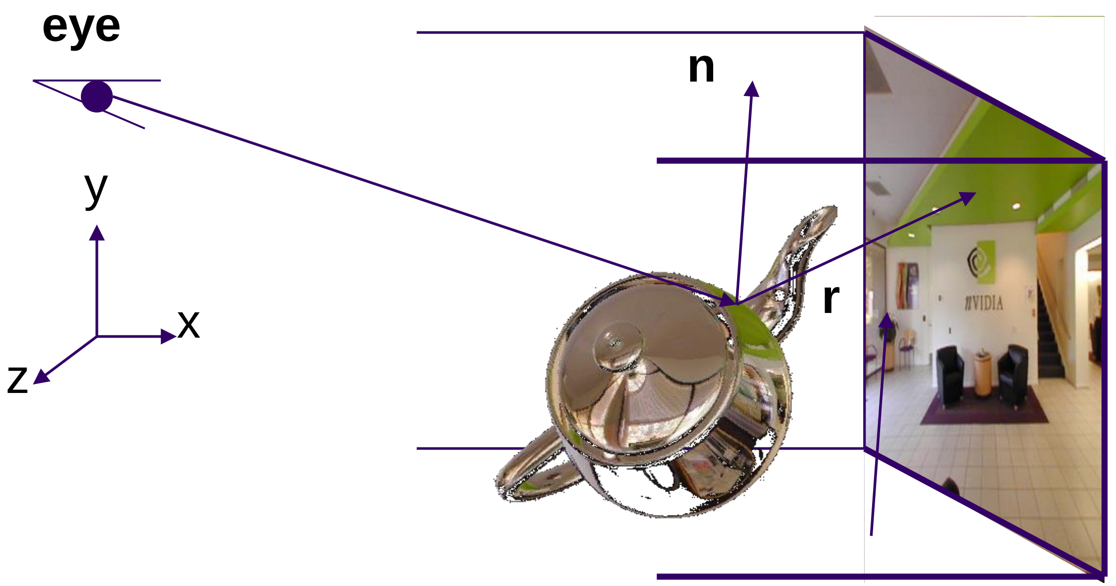
Sphere around object (sphere map)



Cube around object (cube map)



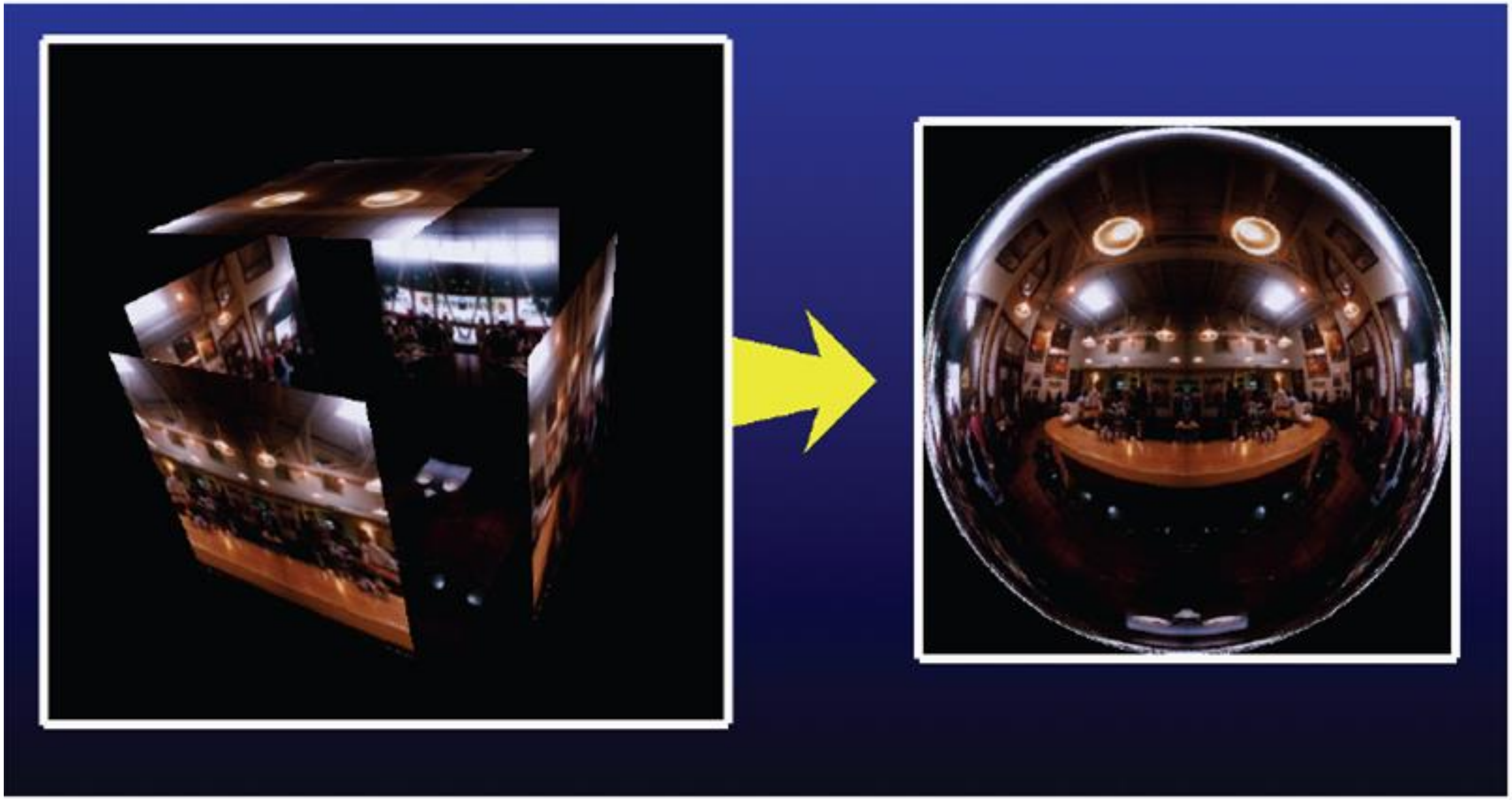
Cube mapping



- Need to compute reflection vector, $\mathbf{r}$
- Use $\mathbf{r}$ by for environment map lookup

Sphere Environment Map

- Cube can be replaced by a sphere (sphere map)



Sphere Mapping

- Original environmental mapping technique
- Proposed by Blinn and Newell
- Uses lines of longitude and latitude to map parametric variables to texture coordinates
- OpenGL supports sphere mapping
- Requires a circular texture map equivalent to an image taken with a fisheye lens

Sphere Map

- A sphere maps is basically a photograph of a reflective sphere in an environment

Paul DeBevec, www.debevec.org

Sphere map

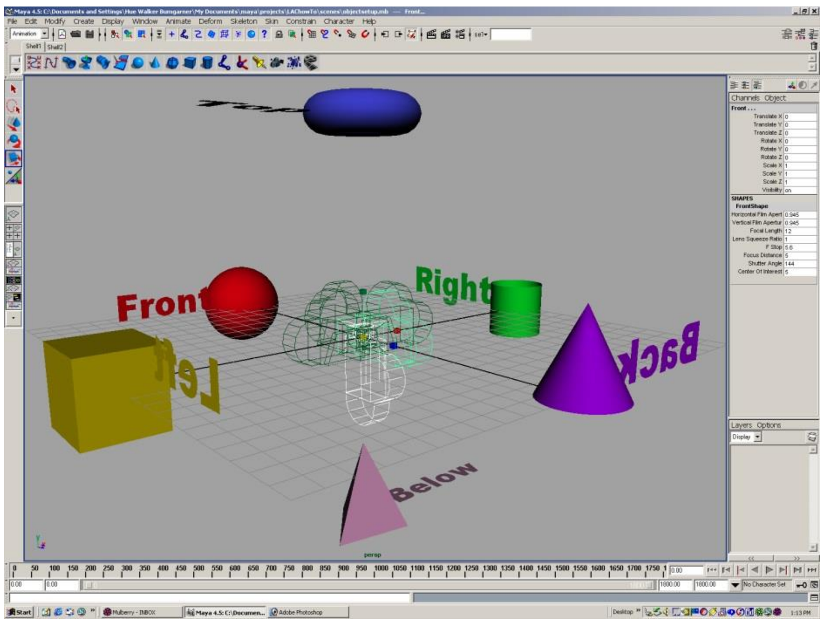
- example

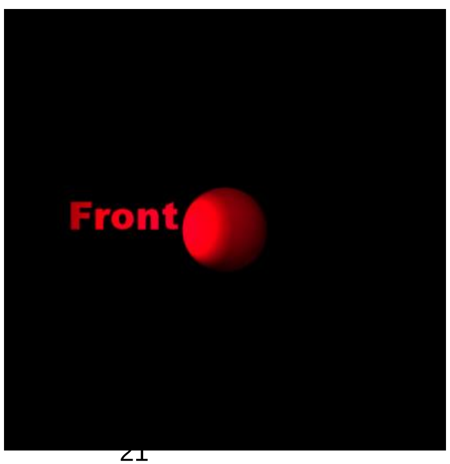
Sphere map
(texture)

Sphere map
applied on torus

Creating Cube Map

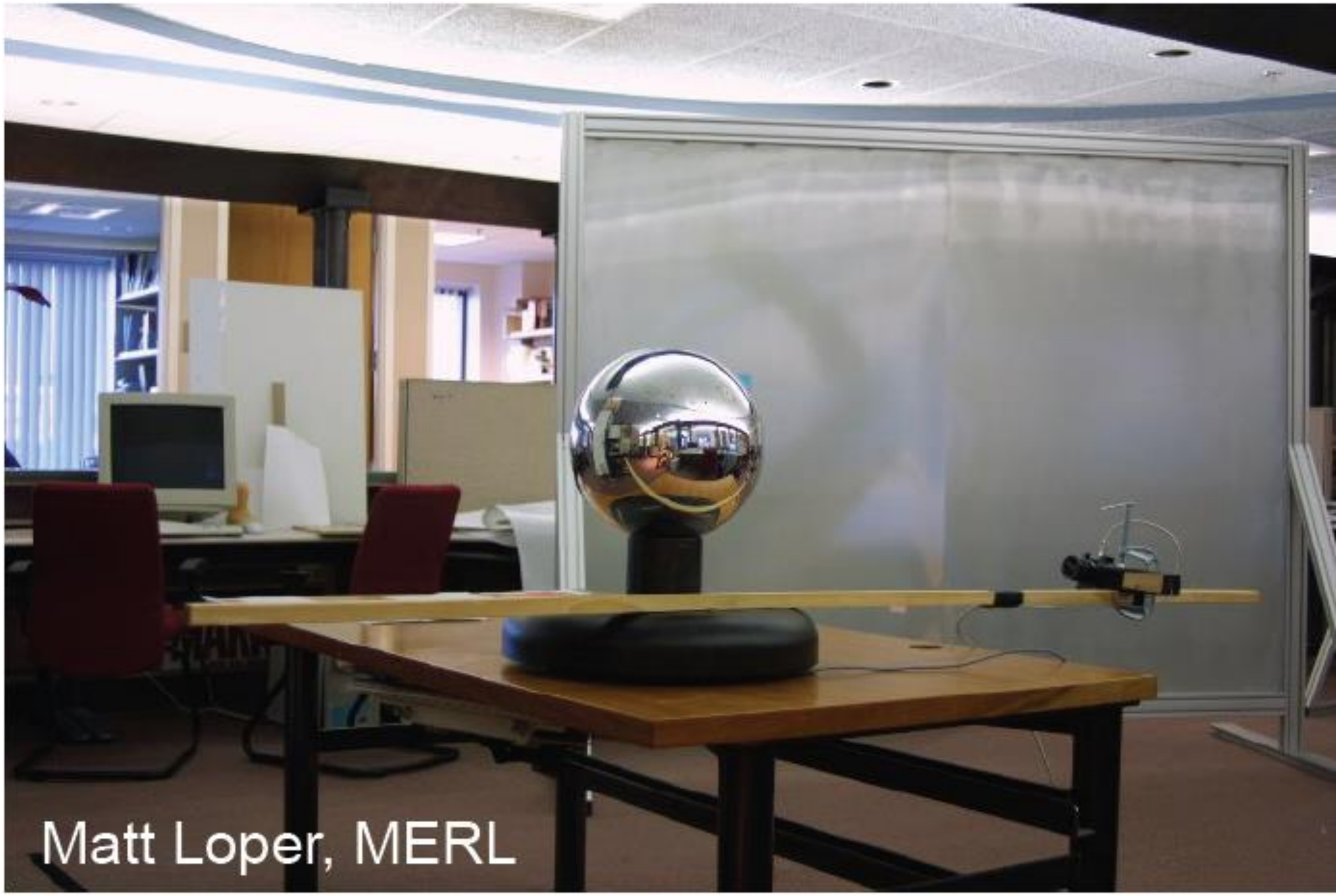
- Use 6 cameras directions from scene center
  - each with a 90 degree angle of view



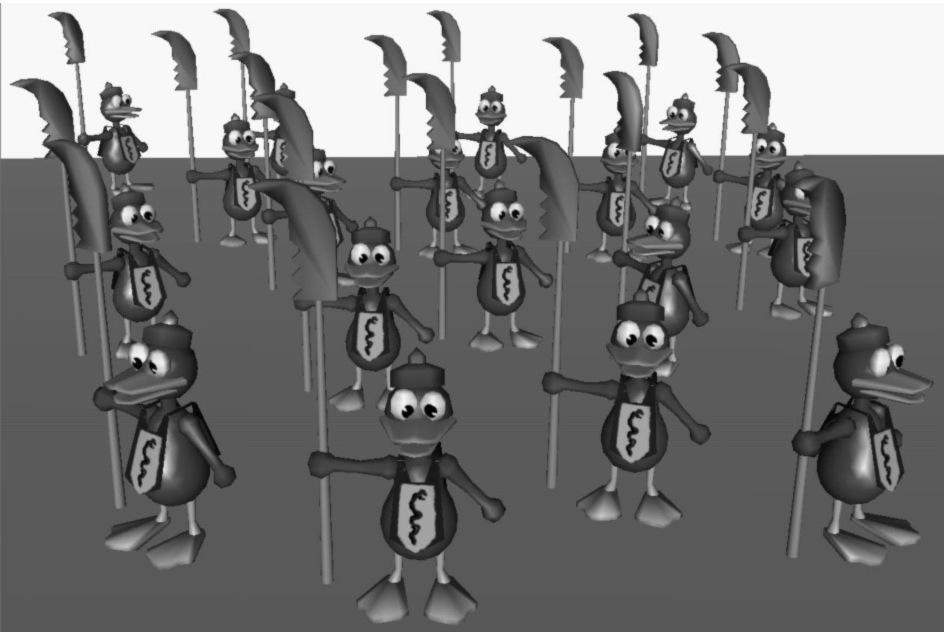


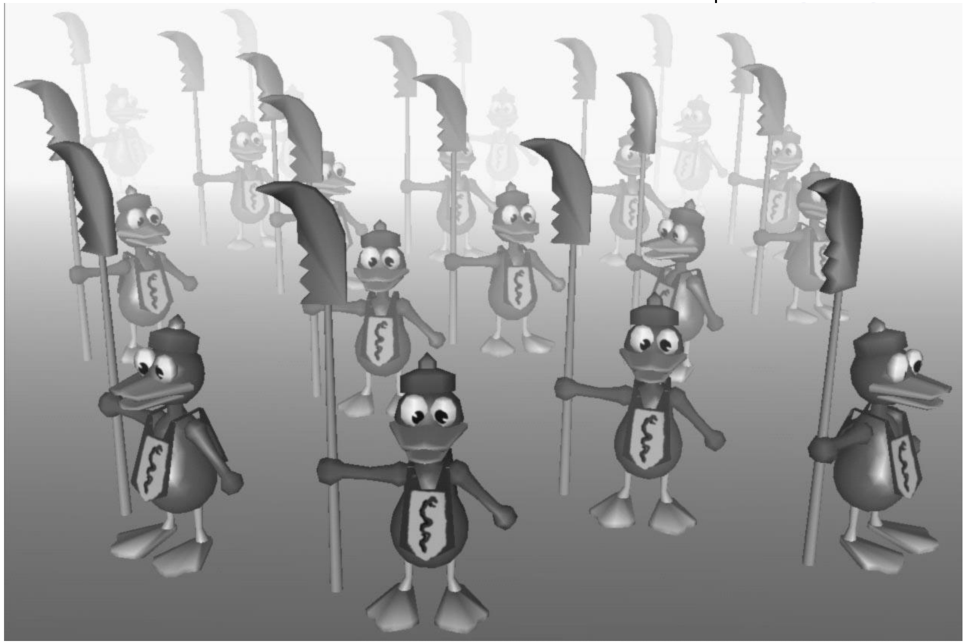
How do we capture a sphere map?

Capturing a Sphere Map



Fog example

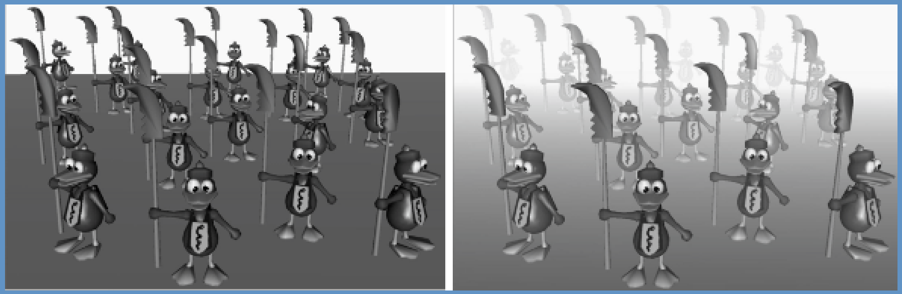




- Fog is atmospheric effect
  - Better realism, helps determine distances

Fog

- **Shaders implementation:** fog applied in fragment shader just before display



How might we generate fog like the one shown in this image?

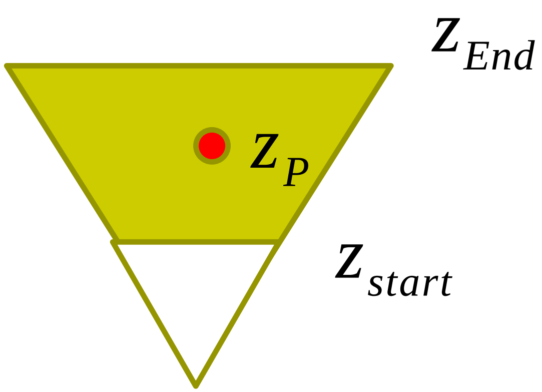
Rendering Fog

- Mix some color of fog: $\mathbf{c}_f$ + color of surface: $\mathbf{c}_s$

$$\mathbf{c}_p = f\mathbf{c}_f + (1-f)\mathbf{c}_s \qquad f \in [0,1]$$

- If *f = 0.25*, output color = 25% fog + 75% surface color
  - *f* computed as function of distance *z*
  - 3 ways: linear, exponential, exponential-squared
  - Linear:

$$f = \frac{z_{end} - z_p}{z_{end} - z_{start}}$$



Fog Shader Fragment Shader Example

$$f = \frac{z_{end} - z_p}{z_{end} - z_{start}}$$

```
float dist = abs(Position.z);
Float fogFactor = (Fog.maxDist - dist)/
                                   Fog.maxDist - Fog.minDist);
fogFactor = clamp(fogFactor, 0.0, 1.0);

vec3 shadeColor = ambient + diffuse + specular
vec3 color = mix(Fog.color, shadeColor,fogFactor);
FragColor = vec4(color, 1.0);
```

$$\mathbf{c}_p = f\mathbf{c}_f + (1 - f)\mathbf{c}_s$$