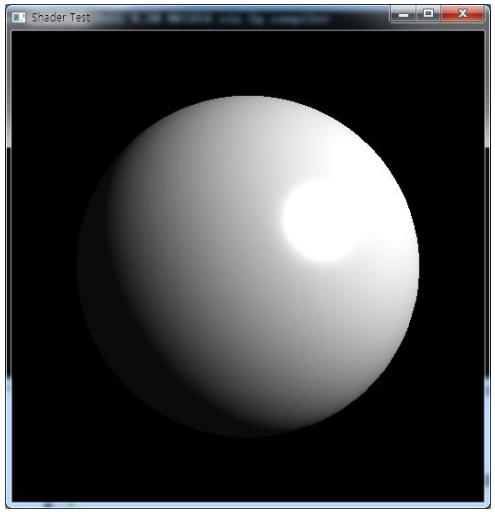
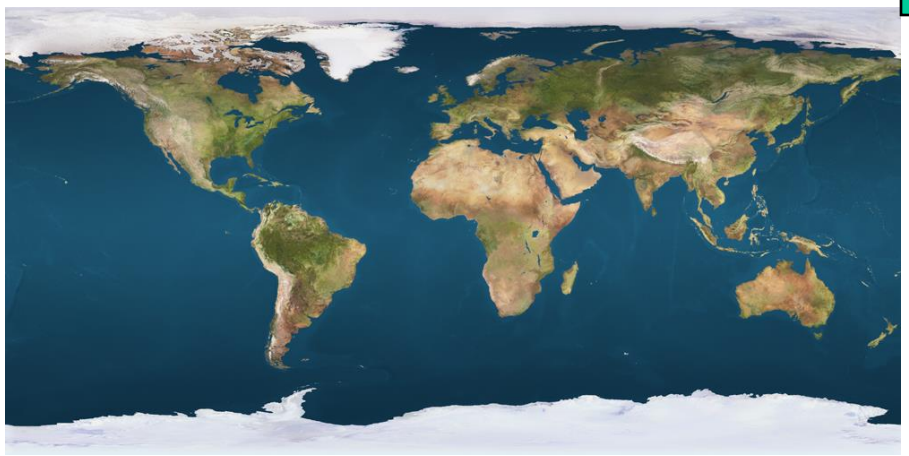
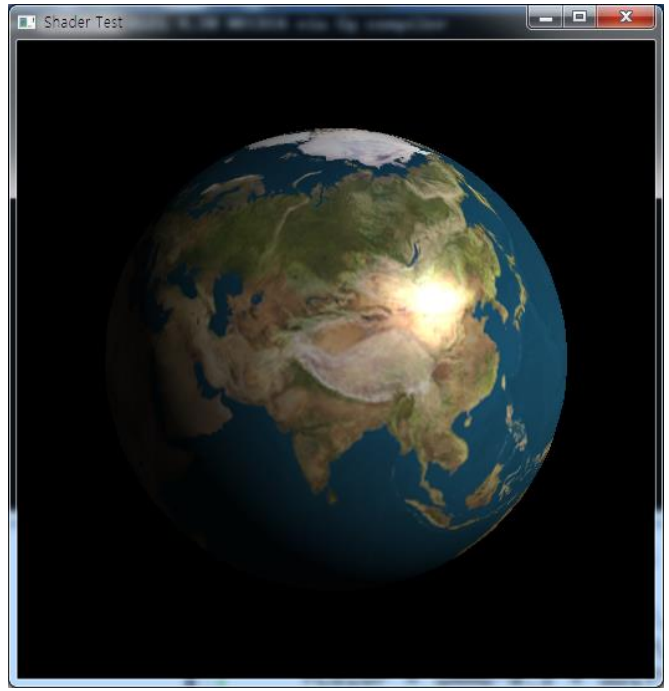
# Texture Mapping & Environmental Mapping

# Coding practice: planet earth

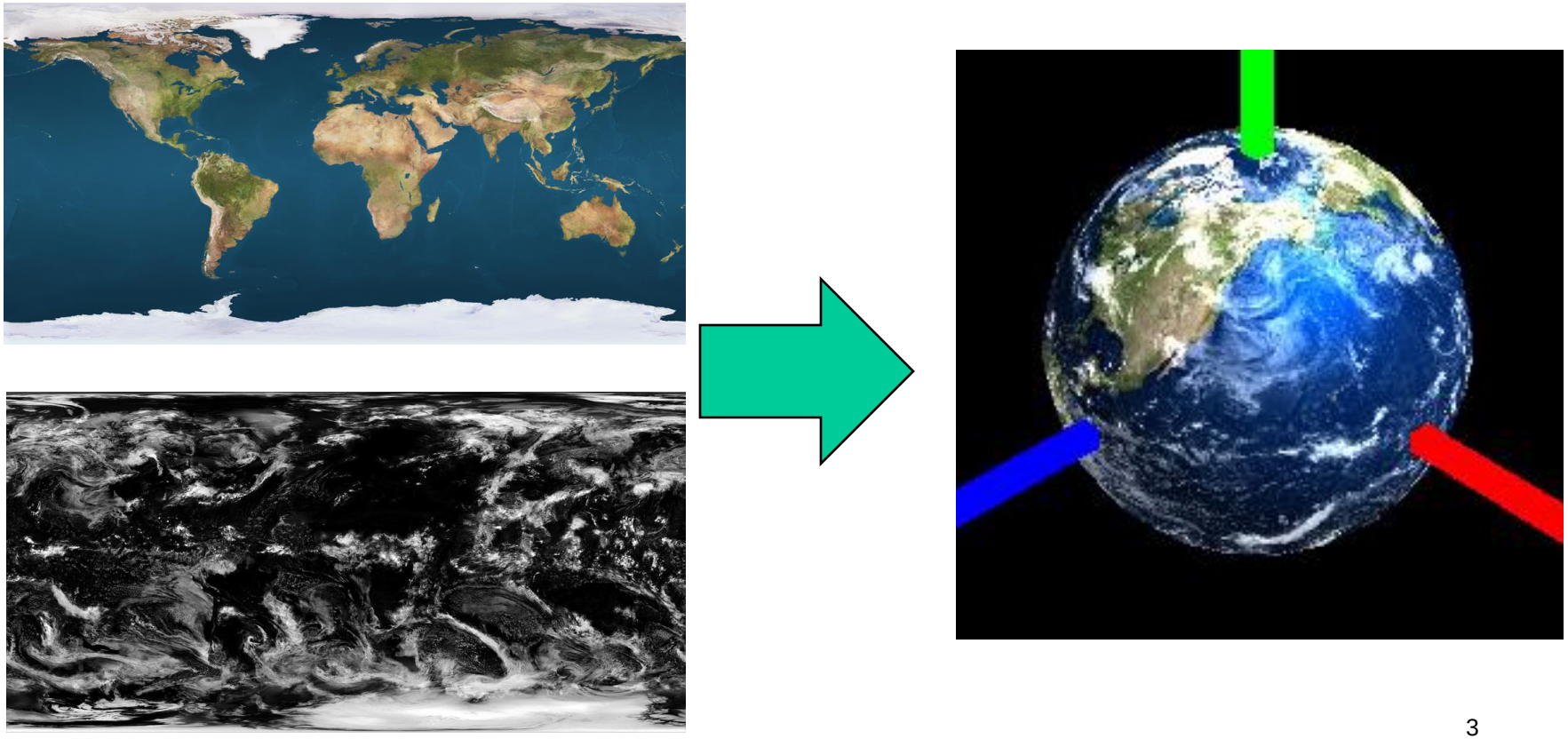




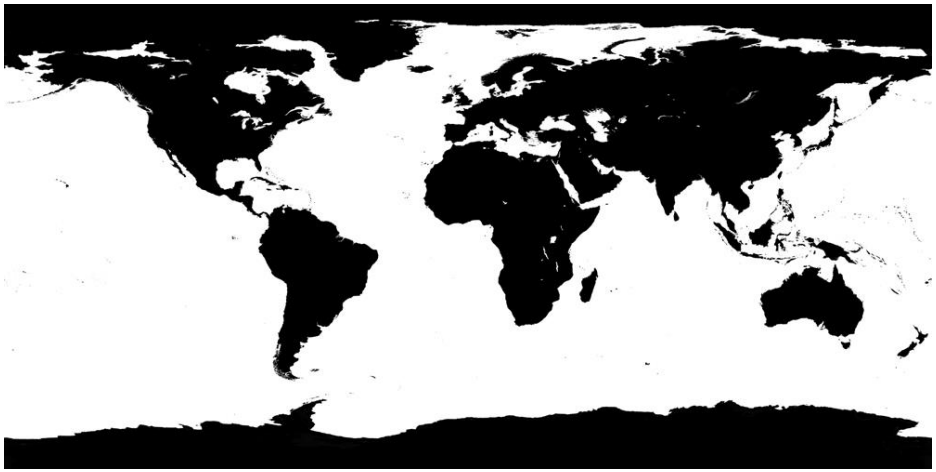
# Multi-Texturing

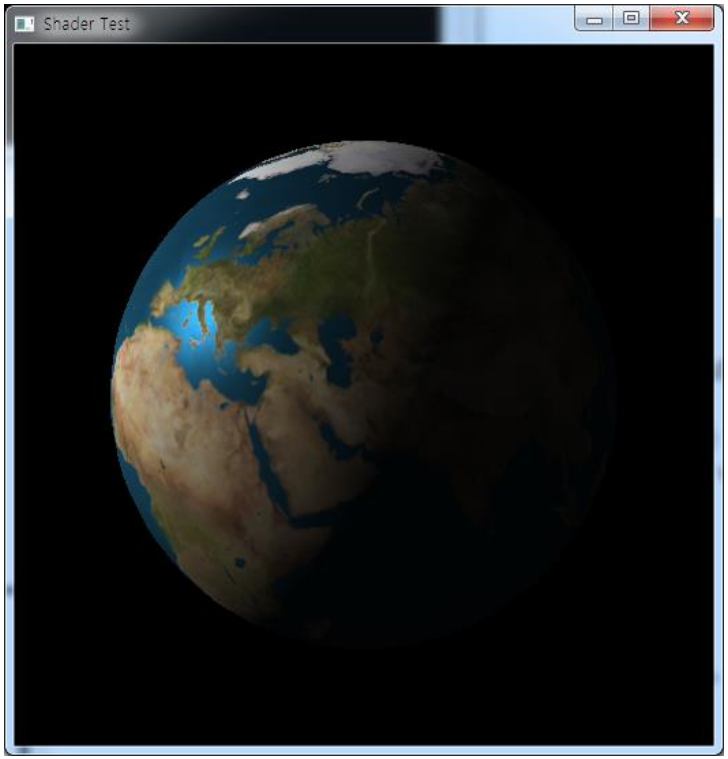
- Apply multiple textures at the same surface:



# Multi-texturing: Specular Map:

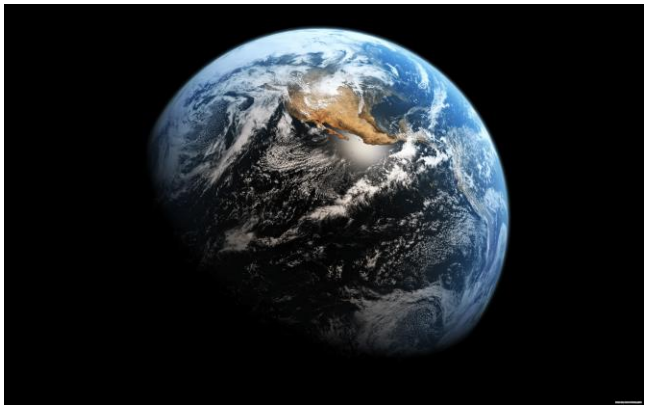
- Specify the specular reflectance by giving the additional texture (specular map)



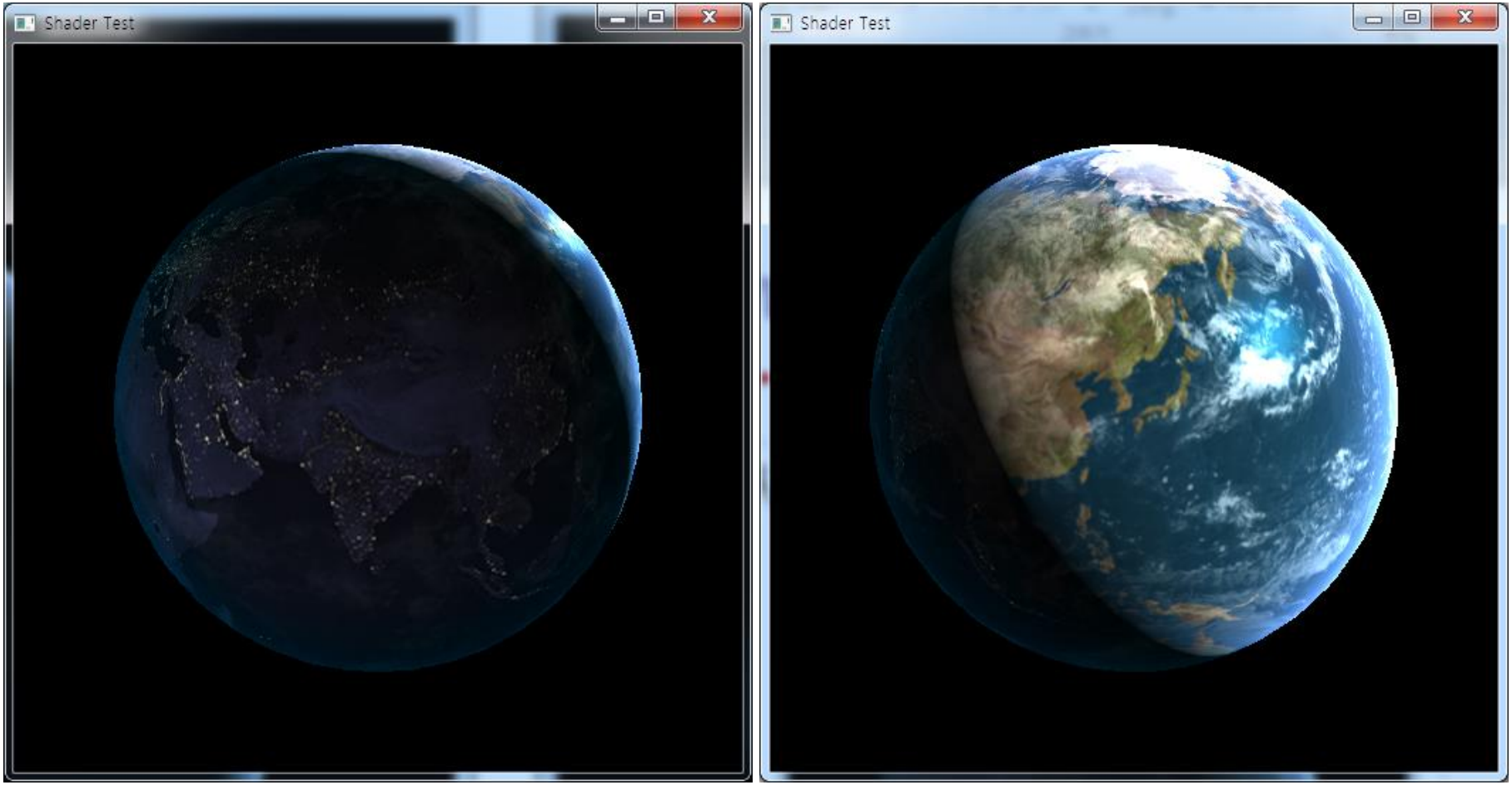


# Think more:

- Observe the photos of the earth
- What effects you need more and how to do implement?

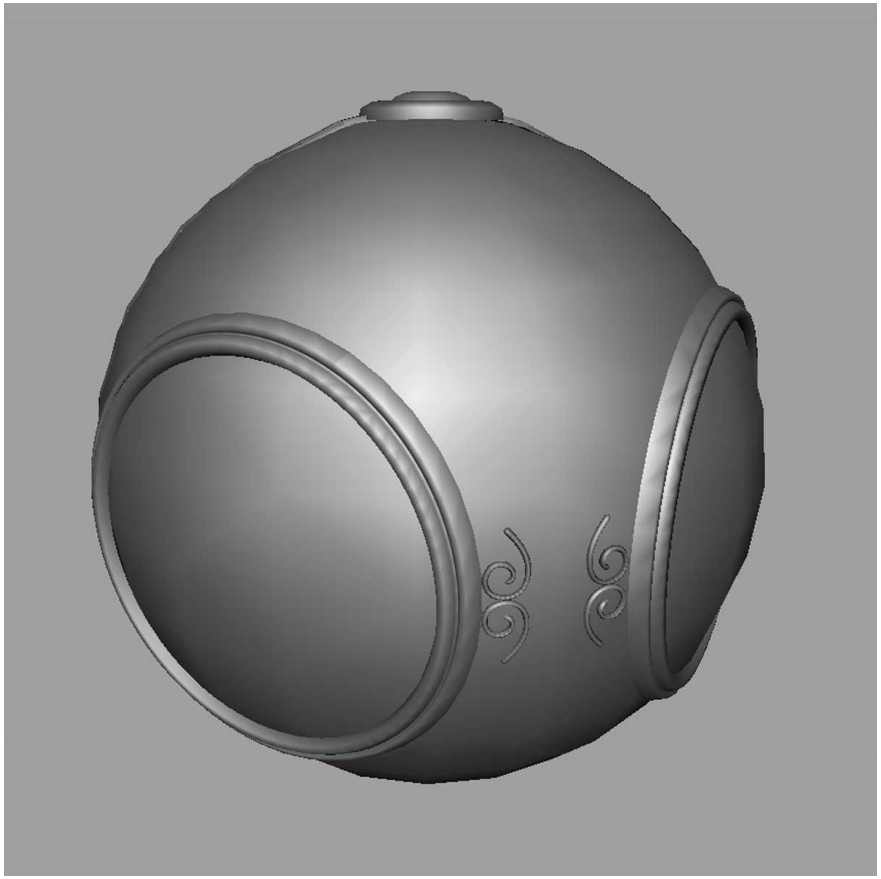


# Make your own Planet Earth:

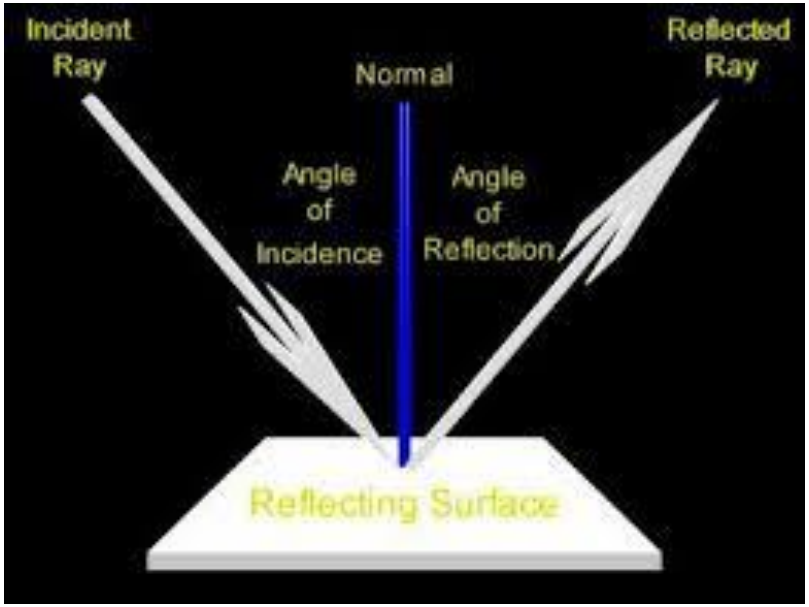


# Environmental Mapping

# Example



# Key observation: Parameterization



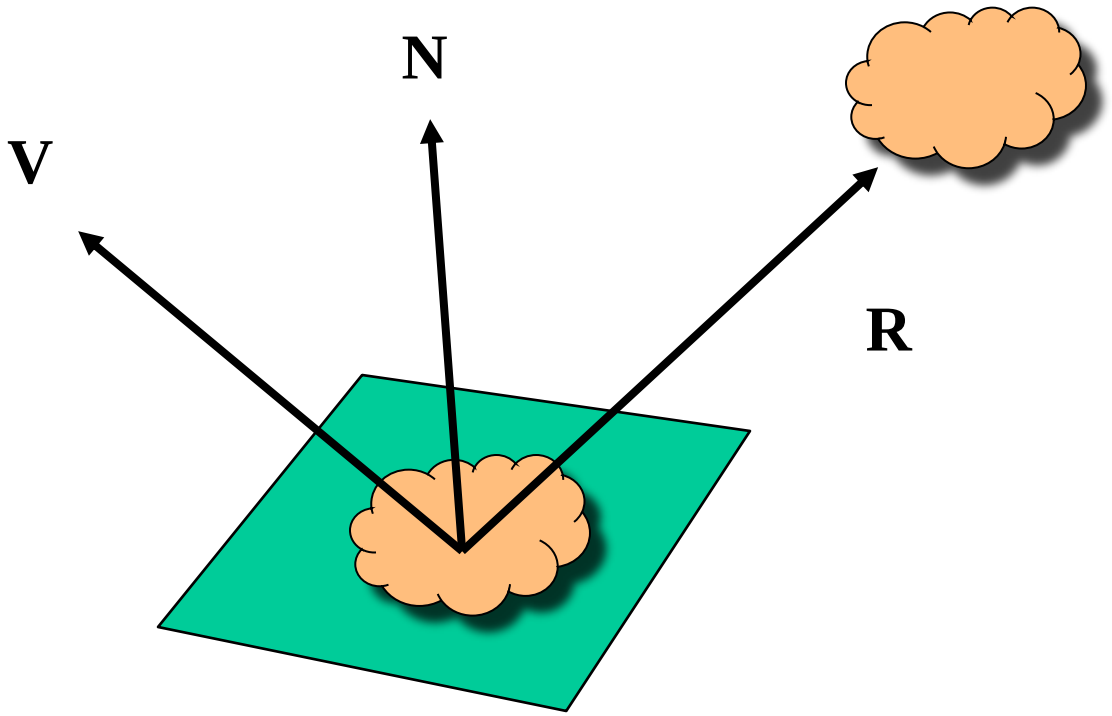

Directions

Texture coordinates

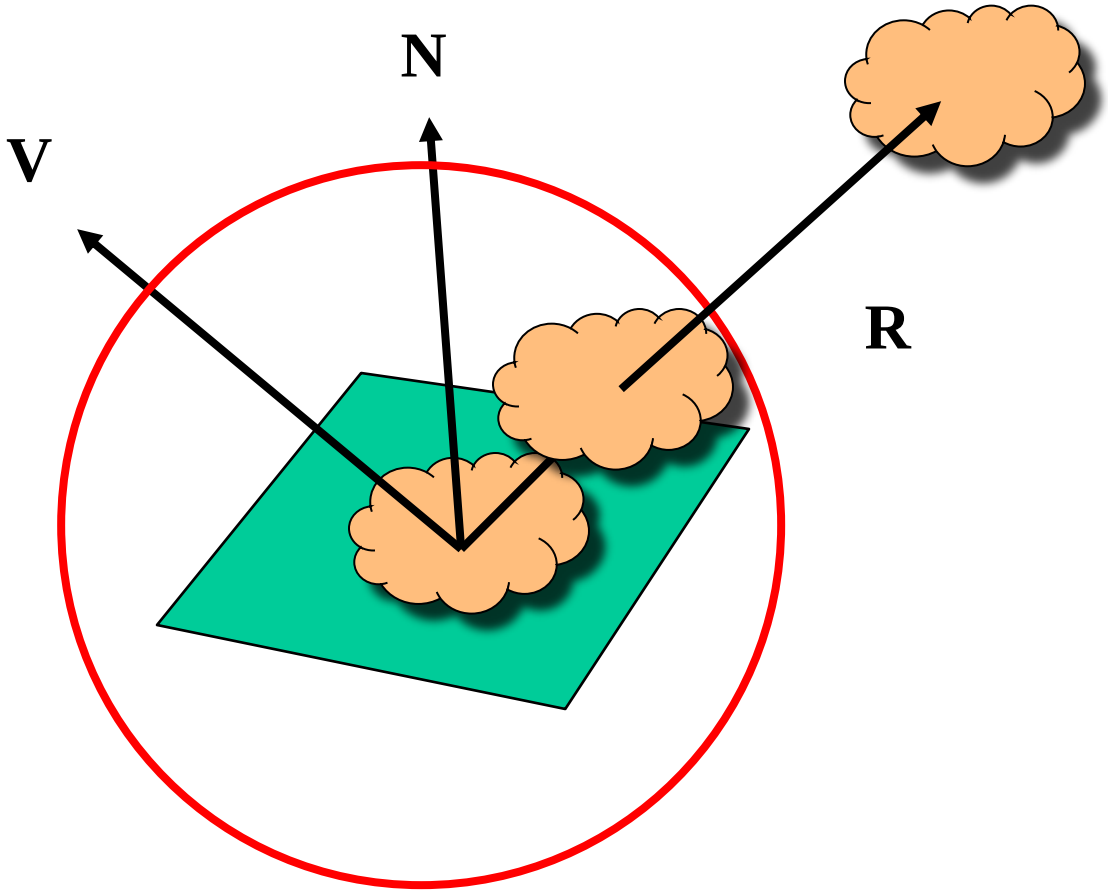
# Parameterization

- **Spherical**
- Cubic

# Reflecting the Environment



# Mapping to a Sphere

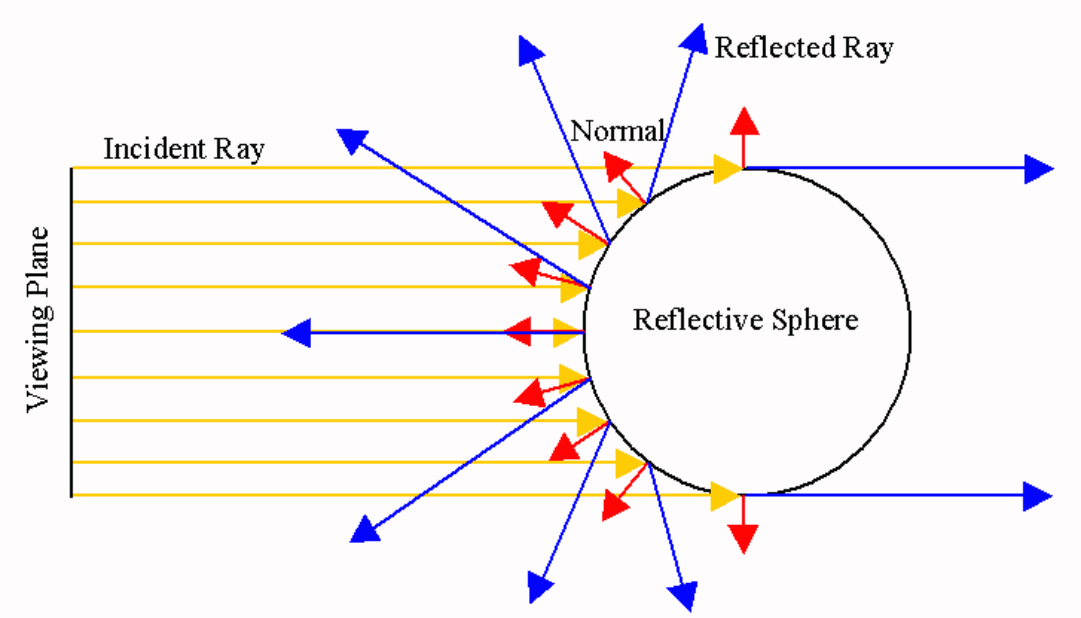


# Spherical Map

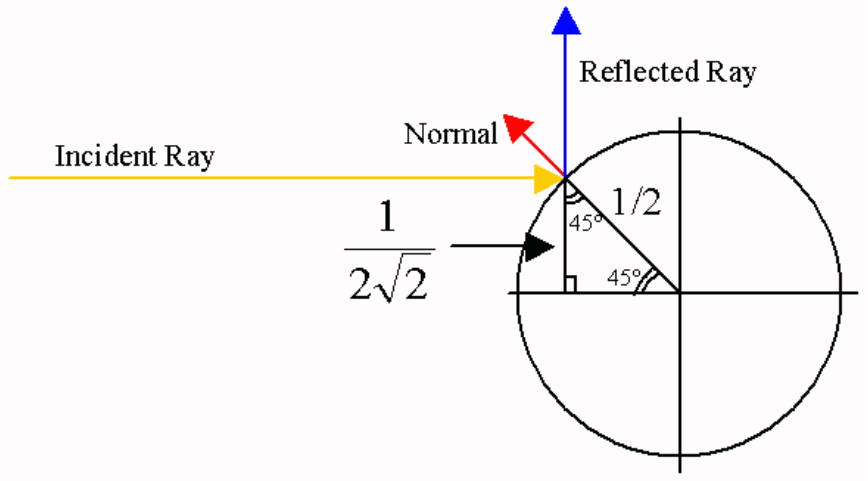
- Original environmental mapping technique proposed by Blinn and Newell.

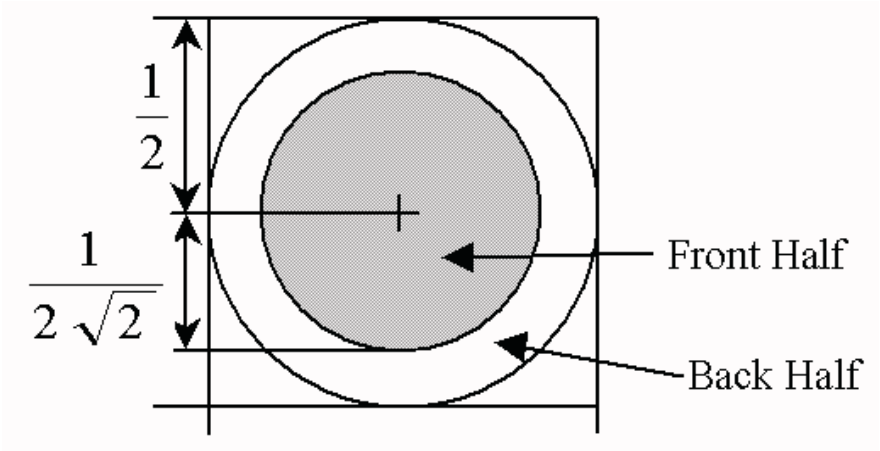
# Spherical Environ. Mapping

image of perfect sphere reflector seen from infinity (orthographic)




# Non-uniform Sampling



# Today: Spherical Cameras are coming!

Nokia Ozo

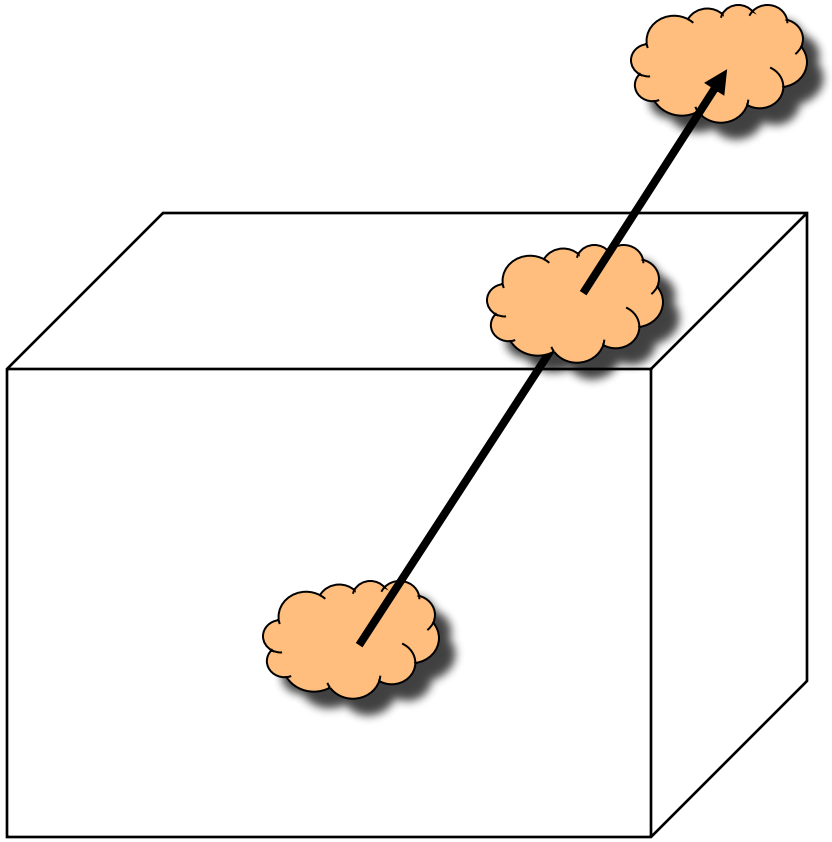
panono

Samsung project beyond

# A image from a 360 camera

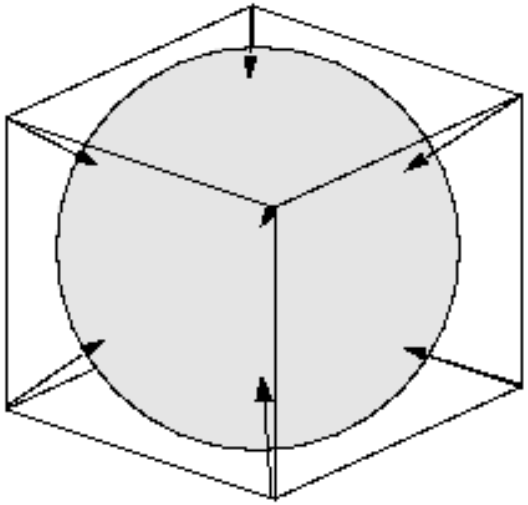
panorama image

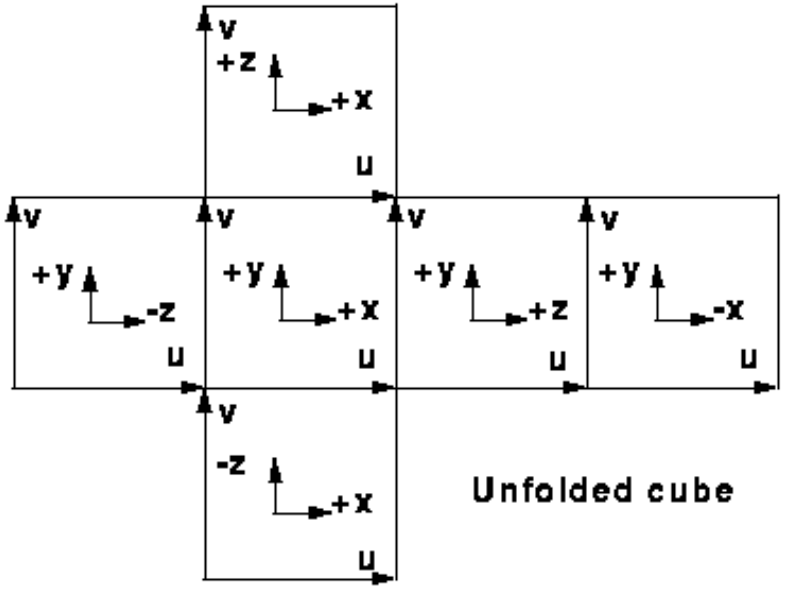
# Cube Map



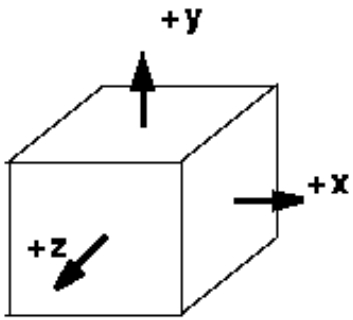
# Cube Mapping

- a simplified method uses the surface normal as an index for the texel on the cube surface

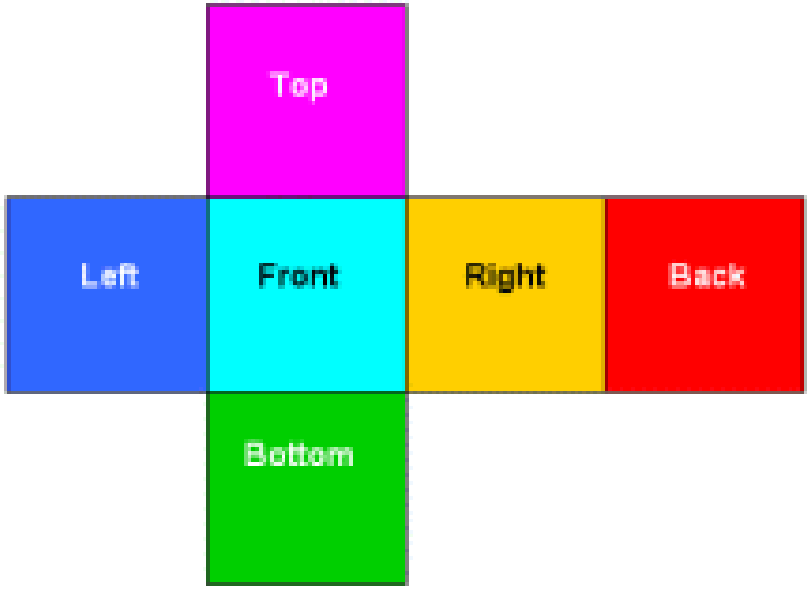


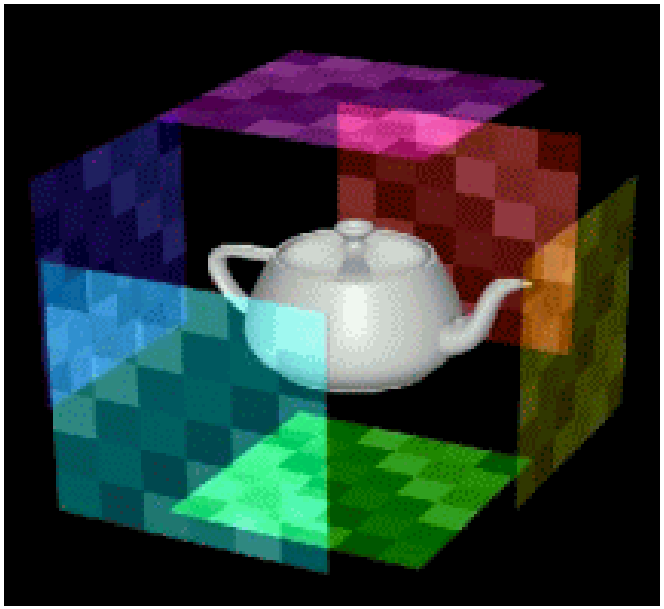



Unfolded cube



# Cube Environment Mapping

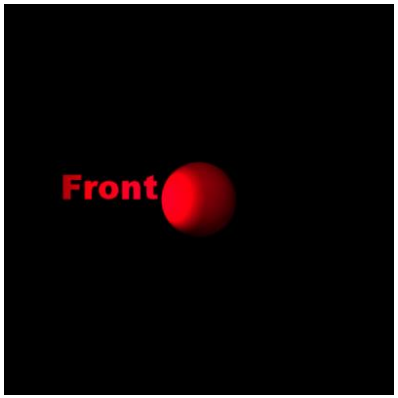


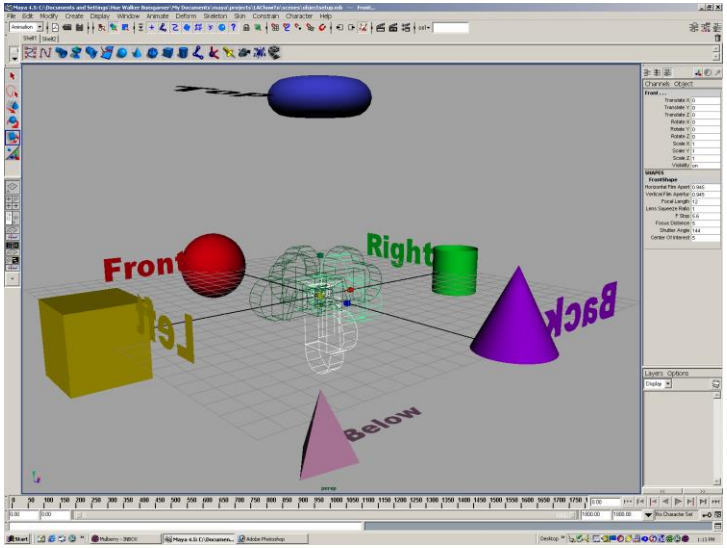



# Cube Environment Mapping

# Forming Cube Map

- Use six cameras, each with a 90 degree angle of view.



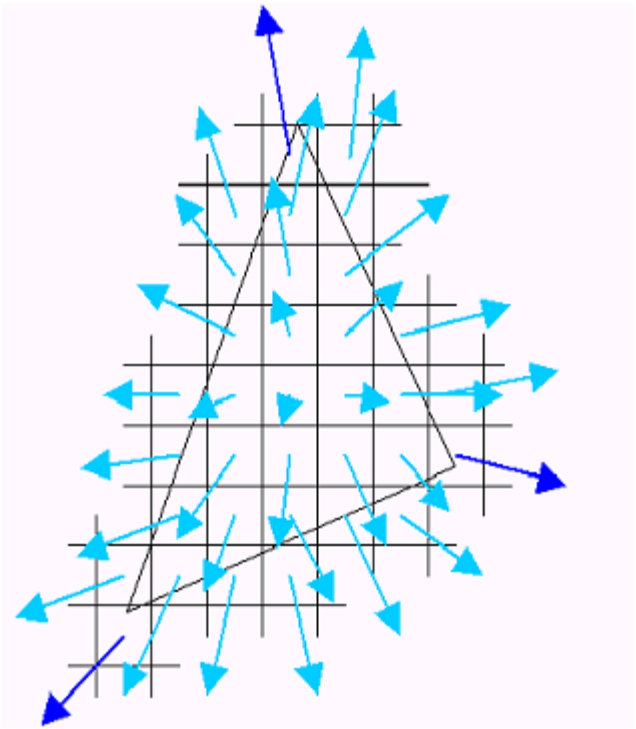



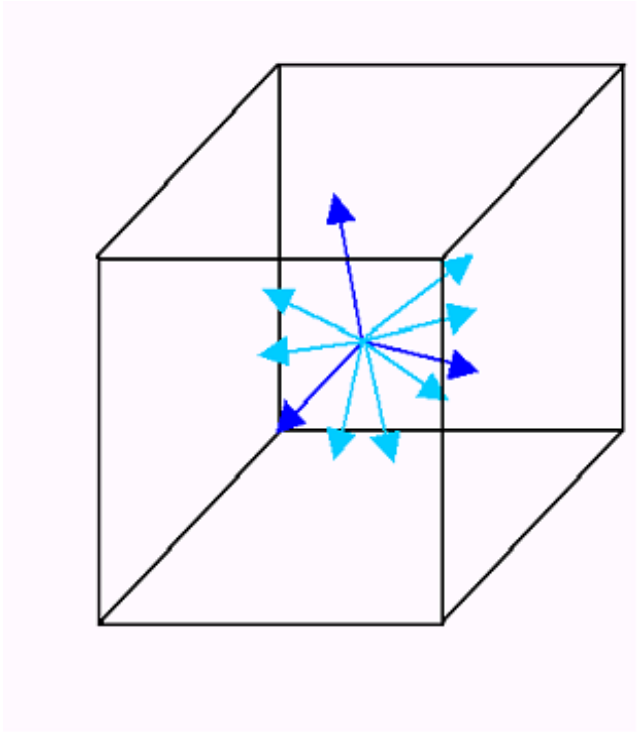


- Good place to find cubemaps:

http://www.humus.name/index.php?page=Textures

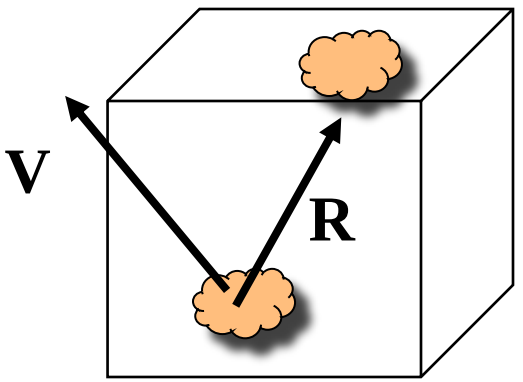
# Cube Maps assumption





# Indexing into Cube Map

- Compute $\mathbf{R} = 2(\mathbf{N}\cdot\mathbf{V})\mathbf{N}-\mathbf{V}$
- Object at origin.




- Use largest magnitude component of R to determine face of cube.

- Other two components give texture coordinates.

# Example

- **R** = (-4, 3, -1). Same as **R** = (-1, 0.75, -0.25)
- Use face x = -1 and  y = 0.75,  z = -0.25
- Not quite right since cube defined by x, y, z = $\pm$ 1 rather than [0, 1] range needed for texture coordinates.
- Remap by s = ½ + ½ y,  t = ½ + ½ z
- Hence, s =0.875, t = 0.375

# Issues

- Must assume environment is very far from object (equivalent to the difference between near and distant lights).
- Object cannot be concave (no self reflections possible).
- No reflections between objects.
- Need a reflection map for each object.

# OpenGL Implementation

- OpenGL supports spherical and cube maps.
- First, form map :
  - Use images from a real camera.
  - Form images with OpenGL.
- Texture map it on to object.

# Cubemap in OpenGL

- **Preparation:** Make one texture object out of the six images.

```
GLuint cubeMap;
glGenTextures(1, &cubeMap);

glActiveTexture(GL_TEXTURE0);
glBindTexture(GL_TEXTURE_CUBE_MAP, cubeMap);
```

- **Load 6 Images (image1, image2, … image 6) and send it to GPU**

```
glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_X,
             level, GL_RGB, width, height, border,
             GL_RGB, GL_UNSIGNED_BYTE, image1);
```

- **Same for other five images.**

# OpenGL Cube Map (cont)

- Parameters apply to all six images.

```
glTexParameteri(GL_TEXTURE_CUBE_MAP,
                GL_TEXTURE_MAP_WRAP_S, GL_REPEAT);
```

- Same for t and r.
- Note that texture coordinates are in 3D space (s, t, r).
- Set the filtering options too.

```
glTexParameteri(GL_TEXTURE_CUBE_MAP,
                    GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_CUBE_MAP,
                    GL_TEXTURE_MIN_FILTER, GL_LINEAR);
```

# In Fragment Shader

- Using uniform variable of “samplerCube”

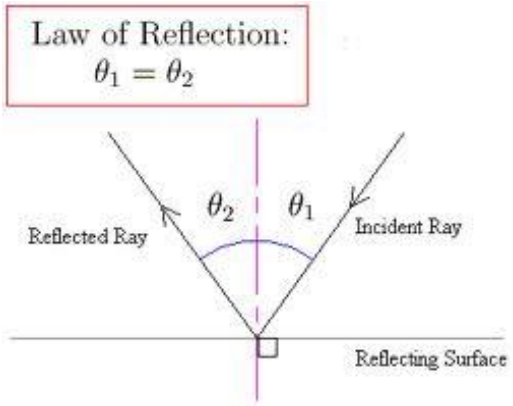
```
#version 430

in  vec4 dir;
out vec4 fColor;

uniform samplerCube uCubeTex;

void main()
{
      fColor = texture(uCubeTex, dir);
}
```

# Reflection




$$\mathbf{R} = 2(\mathbf{N}\cdot\mathbf{V})\mathbf{N}-\mathbf{V}$$

# In Fragment Shader

- Using the function "reflect"

```
vec3 dir = reflect(view_dir, normal);
fColor = texture(uCubeTex, dir);
```

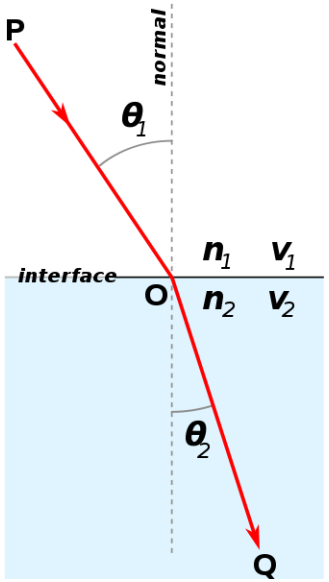
- view_dir: direction from the eye to the position in world coord.

view_dir = position – eye_positoin

- Normal: normal vector of the surface in world coord.

# Refraction

- Snell's law $n_1 \sin\theta_1 = n_2 \sin\theta_2$ .




n: speed of light
(refractive index)

- air = 1
- water = 1.333

http://www.youtube.com/watch?v=gwggONU0QZQ

# In Fragment Shader

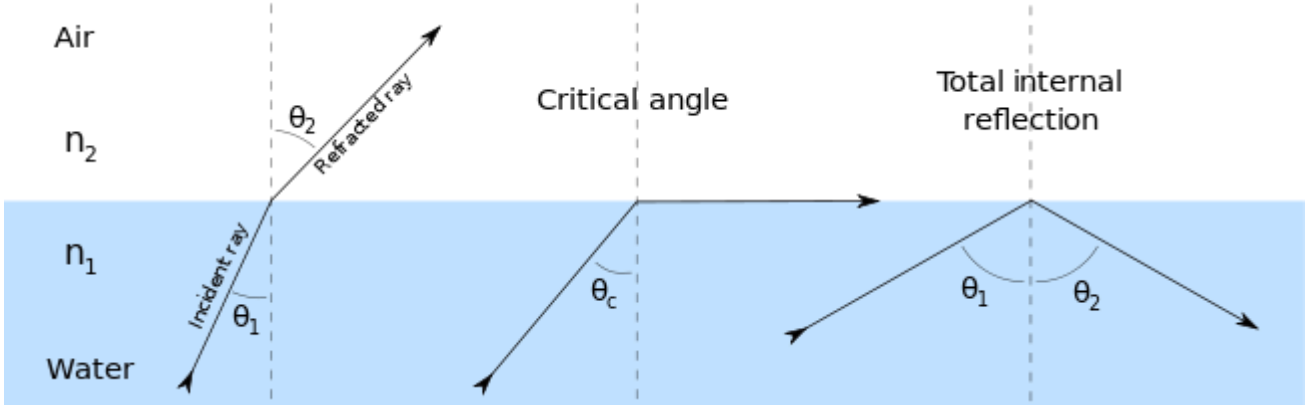
- Using the function “refract”

```
vec3 dir = refract(view_dir, normal, index);
fColor = texture(uCubeTex, dir);
```

- Refractive index:
  - Example:
    from air to water = 1/1.3
    from water to air = 1.3/1

# Issue:

- Critical angle of refraction



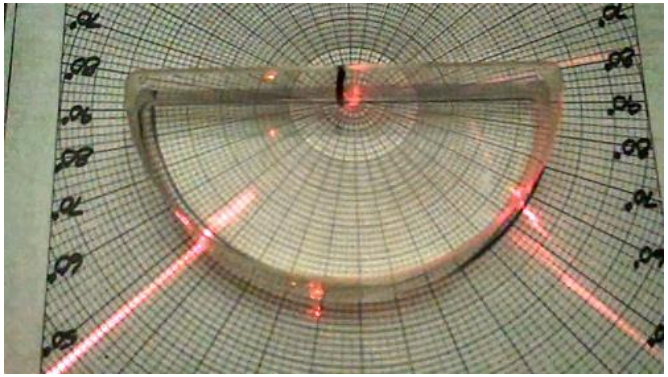

$$n_1 \sin\theta_1 = n_2 \sin\theta_2 \ .$$

$$\sin\theta_2 = \frac{n_1}{n_2}\sin\theta_1 = \frac{1.333}{1}\cdot\sin(50^\circ) = 1.333\cdot 0.766 = 1.021, \quad ????$$

Refraction becomes Reflection

# Issue:

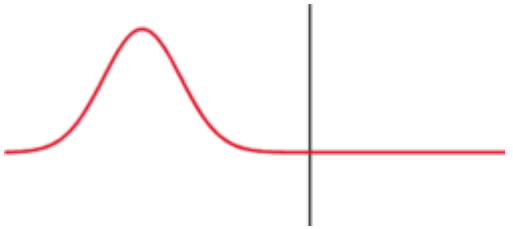
- Critical angle of refraction in real world



# Fresnel Effect

- Fresnel equation describe the movement of light in different media.

# Fresnel Equation

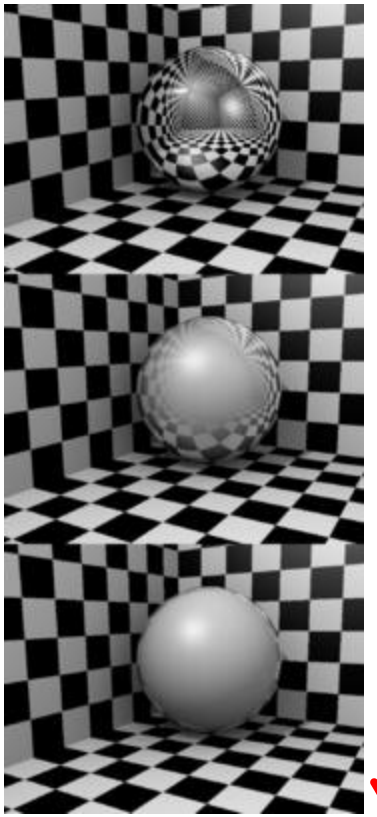


- Equation about how strong the reflect effect is:

$$fr = \underbrace{offset}_{\text{기본값}\approx 0} + \underbrace{scale}_{\text{가중치}\approx(0.1\sim0.5)} * (1 + V \cdot N)^{n}$$

- Coefficient $n \approx (0.5\sim10)$
  larger $n$ : smaller reflect
  smaller $n$ : stronger reflect

- The reflection strength depends on the incident angle!



# Chromatic dispersion (aberration)

- Light separates in many colors because of the refraction of each color is different.