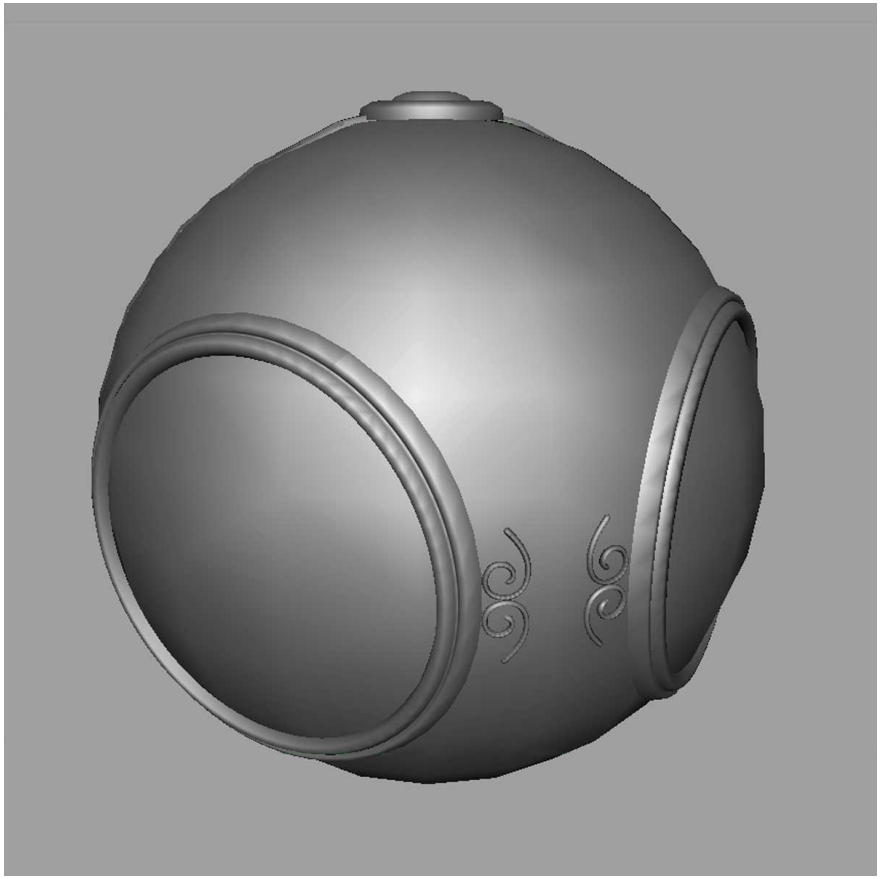
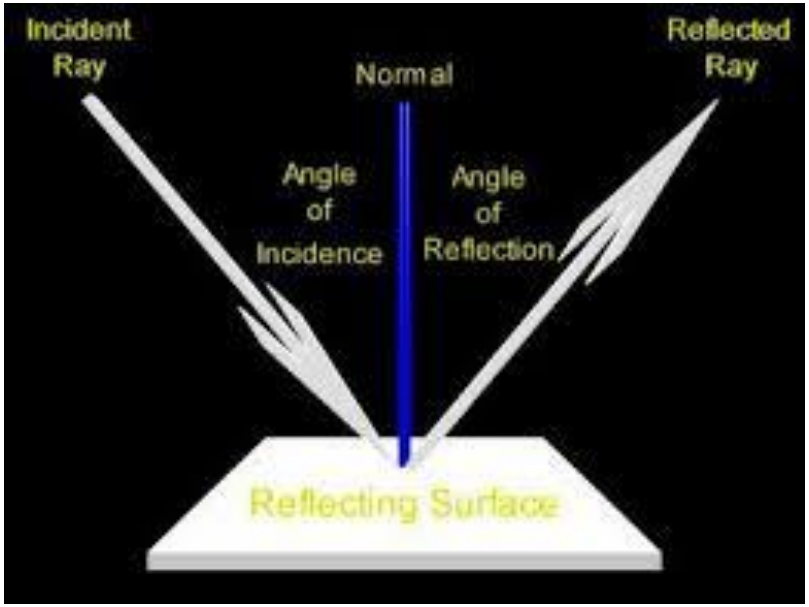
Environmental Mapping

Environmental Mapping

Example



Key observation: Parameterization

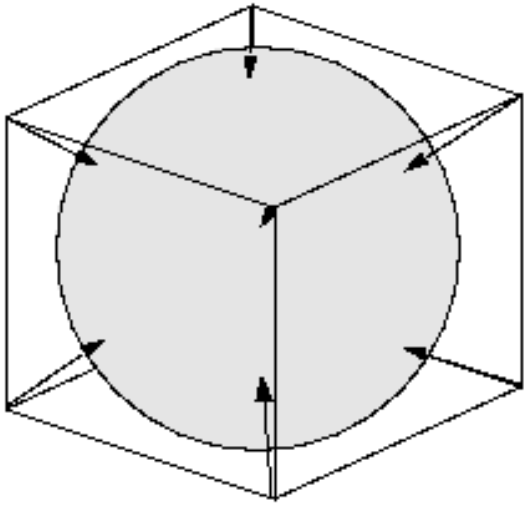


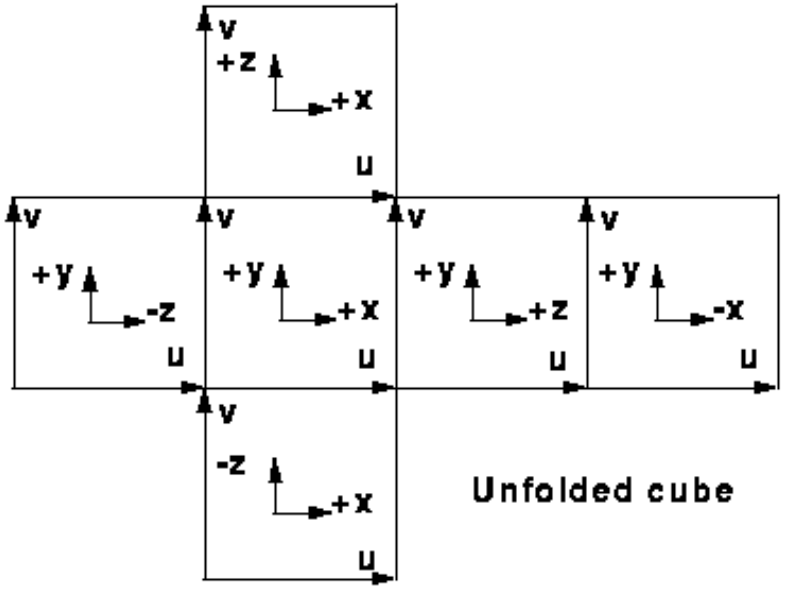
Directions

Texture coordinates

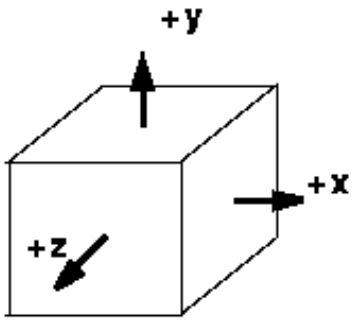
Cube Mapping

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



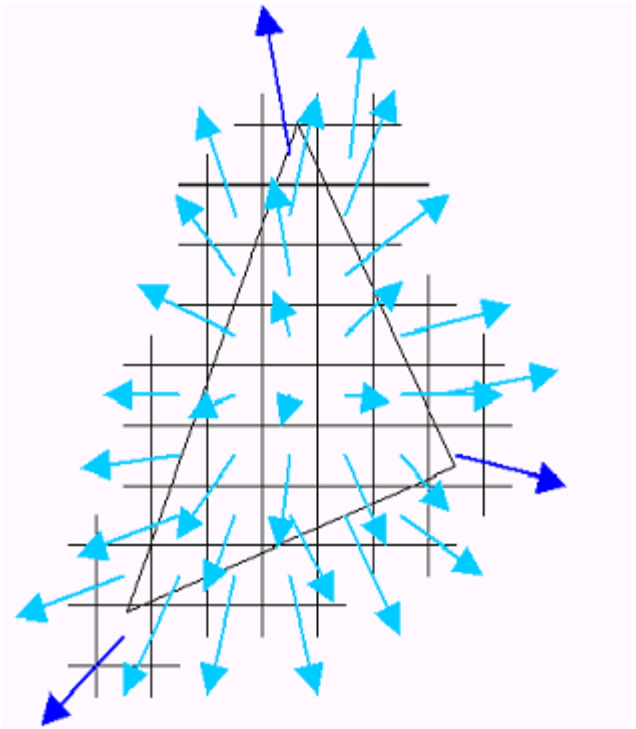


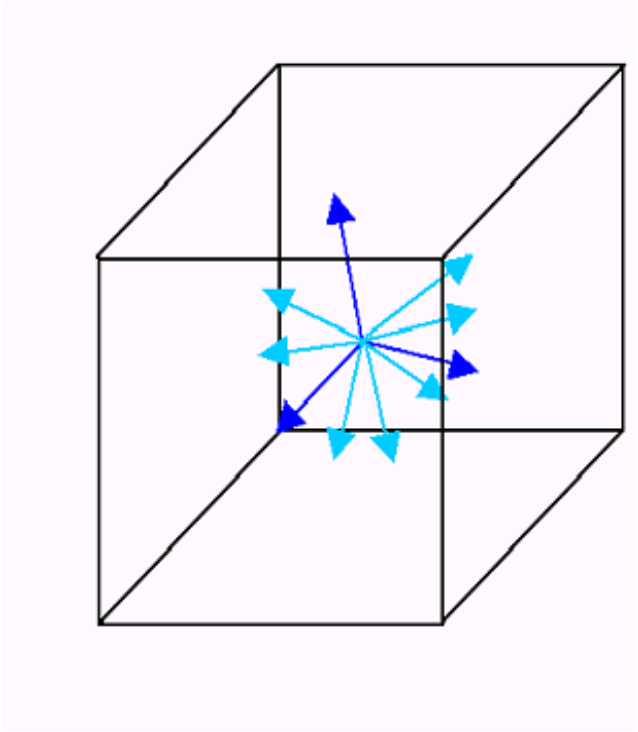
Unfolded cube



Cube Environment Mapping

Cube Maps assumption





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"

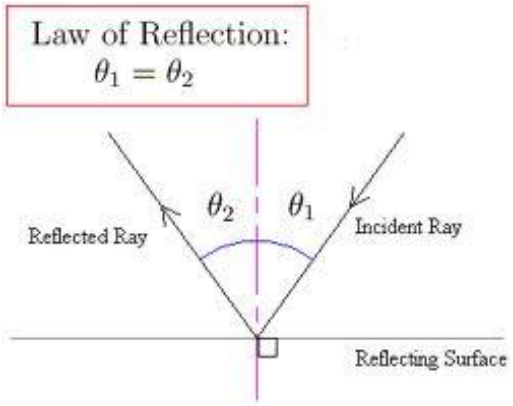
```
#version 430

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);
```

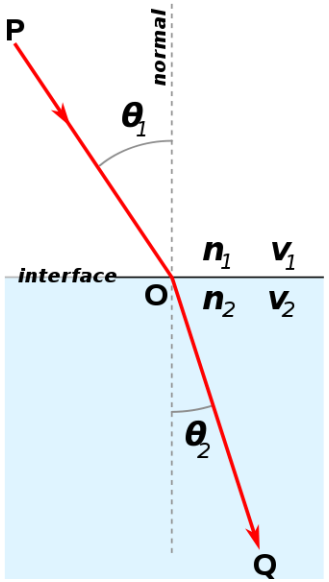
- view_dir: direction from the eye to the position in world coord.

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

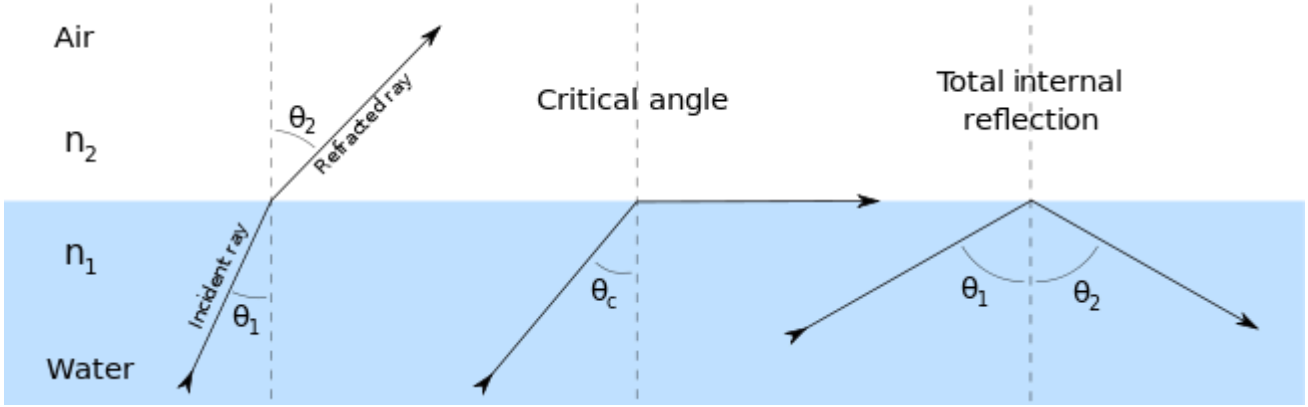
- Using the function “refract”

```
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



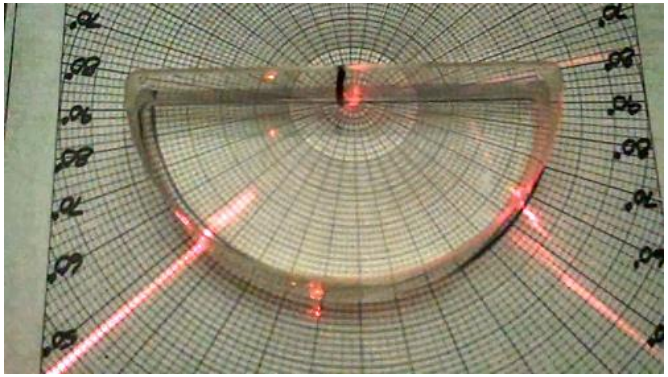
$$n_1 \sin\theta_1 = n_2 \sin\theta_2 \ .$$

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

Refraction becomes Reflection

Issue:

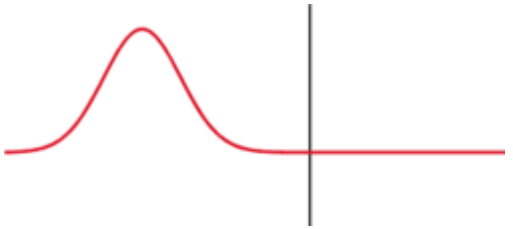
- Critical angle of refraction in real world



Fresnel Effect

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

Fresnel Equation

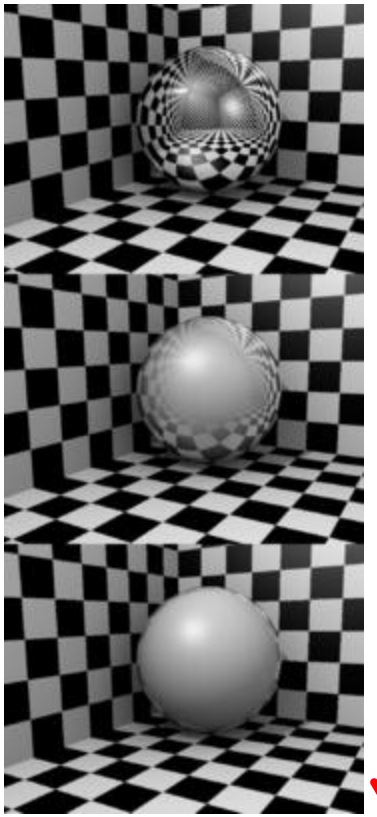


- Equation about how strong the reflect effect is:

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

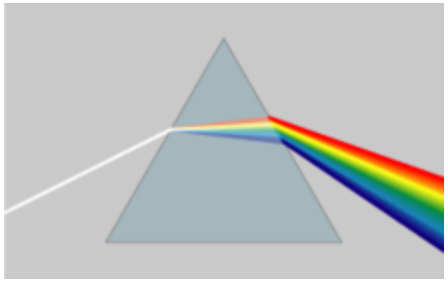
- Coefficient $n \approx (0.5\sim10)$
  larger n : smaller reflect
  smaller n : stronger reflect

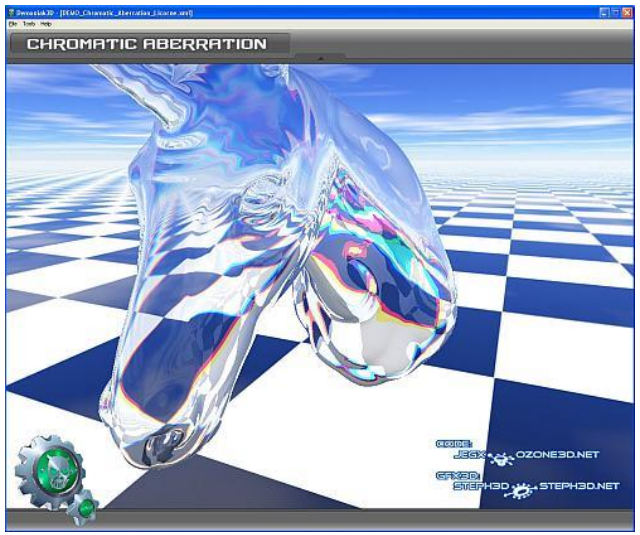
- The reflection strength depends on the incident angle!

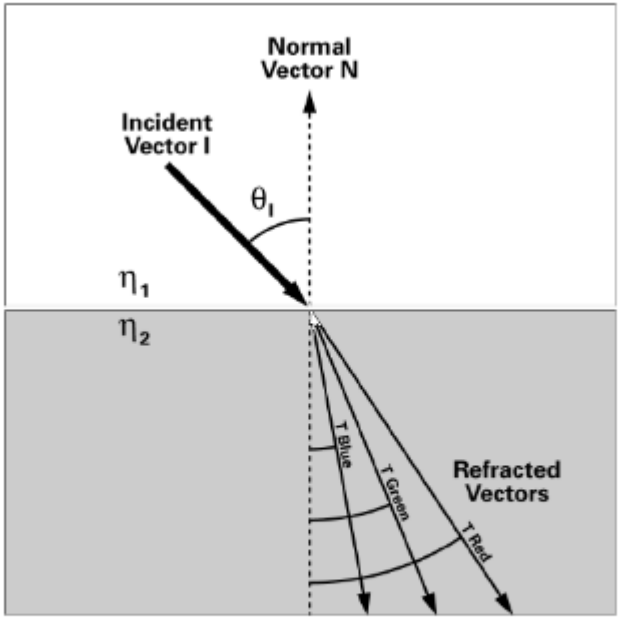


Chromatic dispersion (aberration)

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



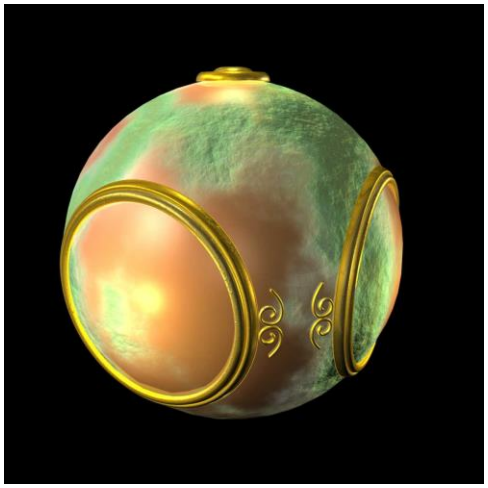




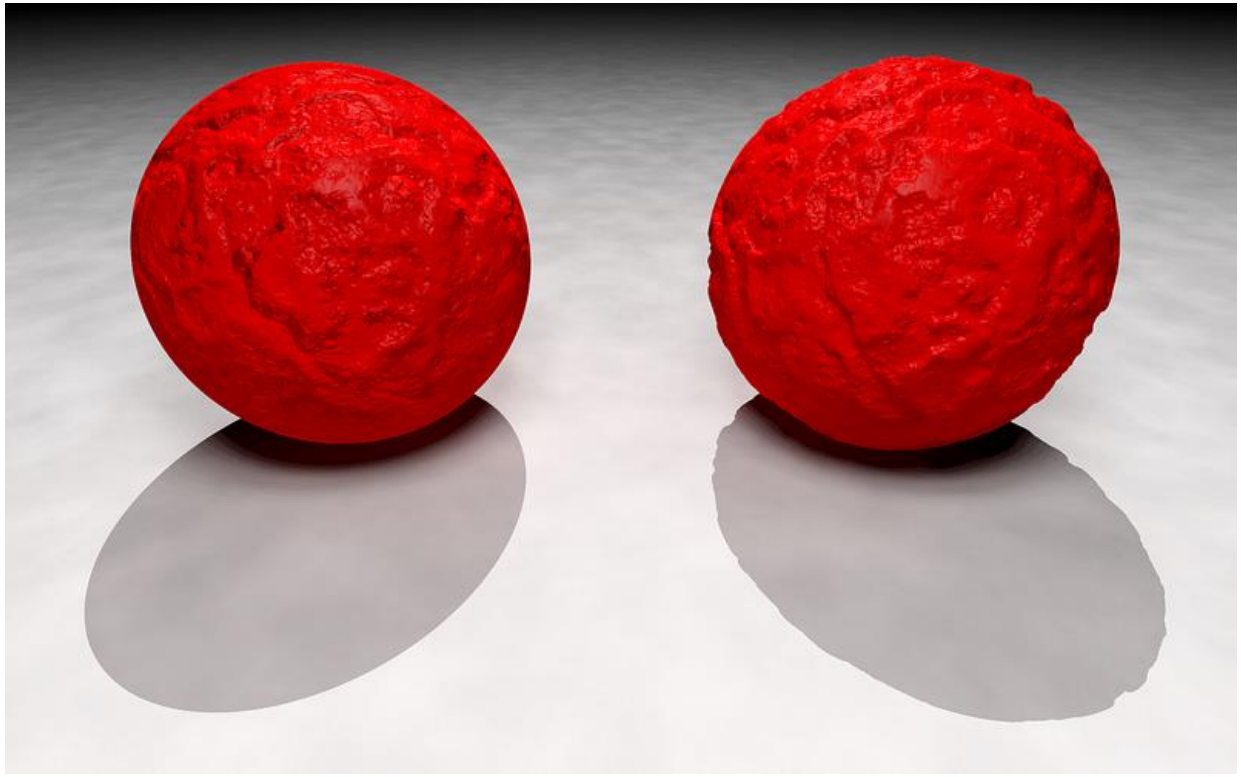
Bump Mapping:
Creating complexity without triangles

Bump Mapping

- Adding Complexity to the surface without adding geometry.
- Idea:
  - Perturb normal for each fragment.
  - Store perturbation as textures.



Which one is fake?

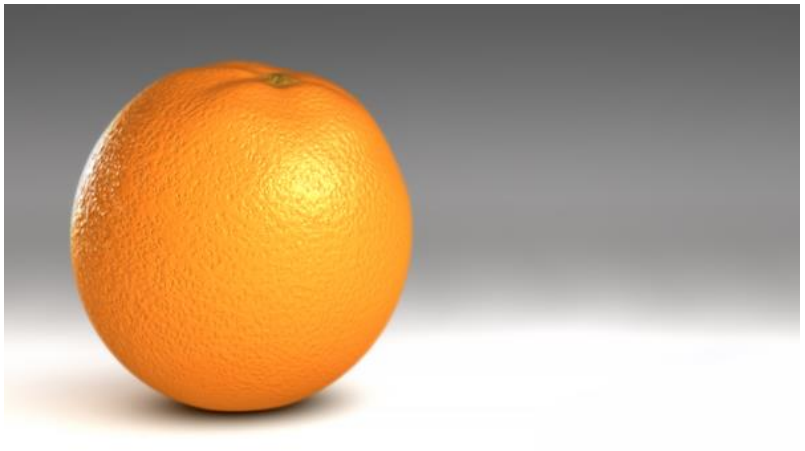


With bump mapping
(not many triangles)

With actual geometry
(many triangles)

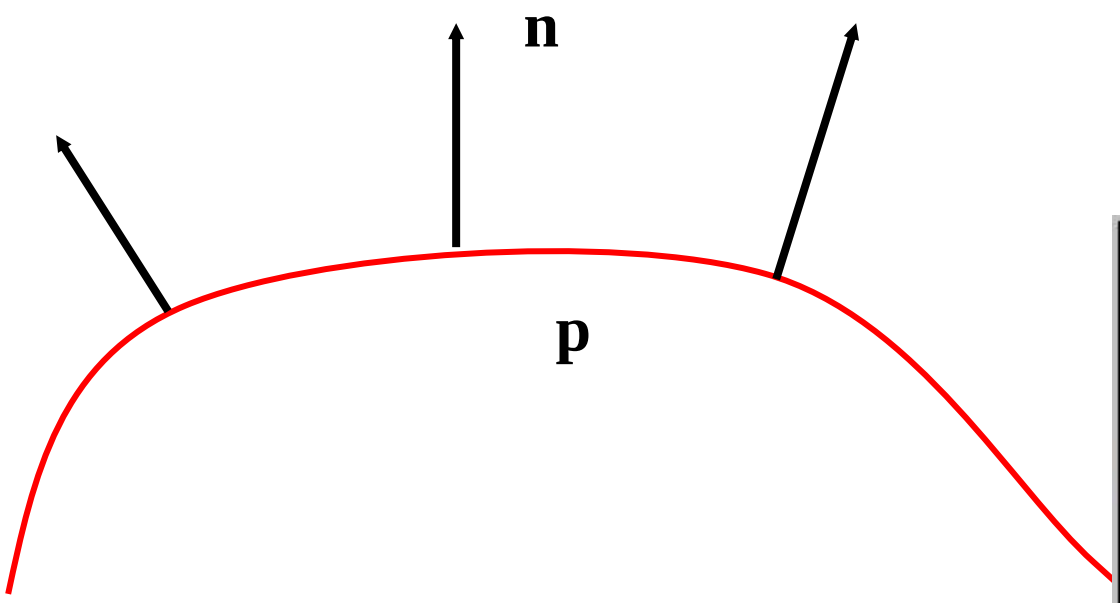
Modeling an Orange

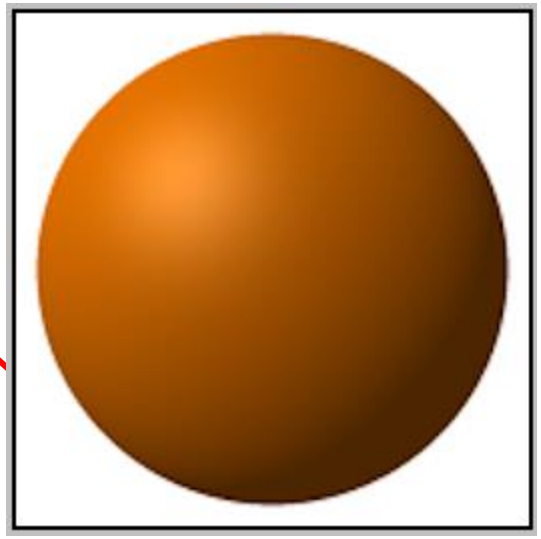
- Texture map a photo of an orange onto a sphere
  - Normal vectors are smooth.
  - Shades of dimples are not correct if viewer or light is moving.
- How to compute (store) the correct normal?



Bump Mapping Process

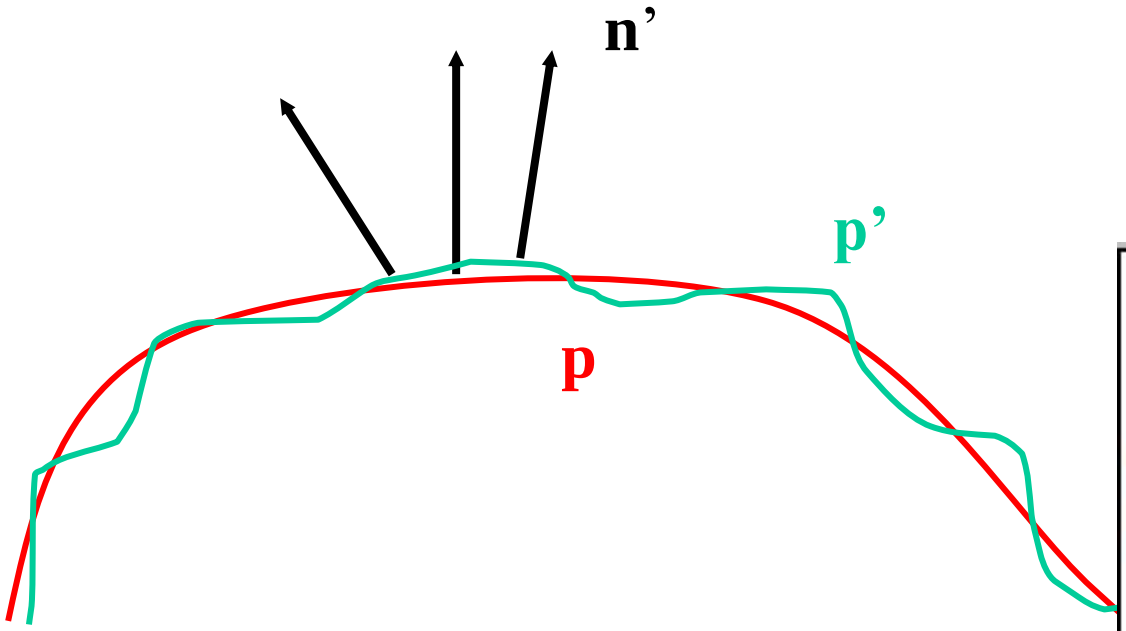
- Consider a smooth surface

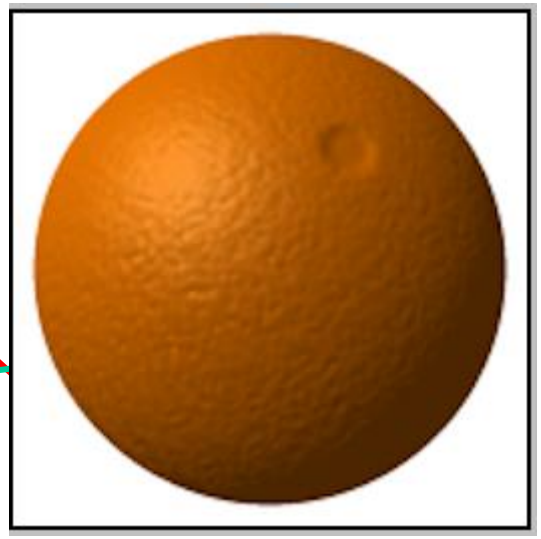




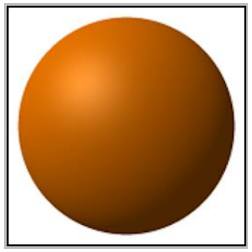
Perturb the normals

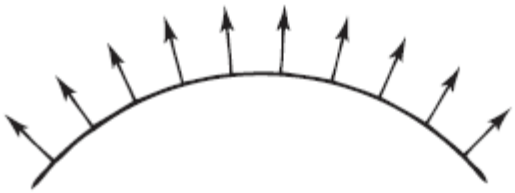
- Change the normal as if it is rough.



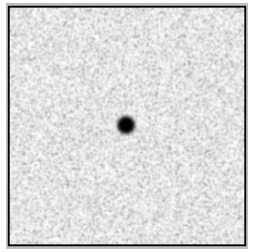


Process outline



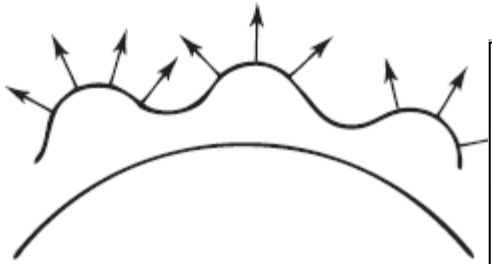


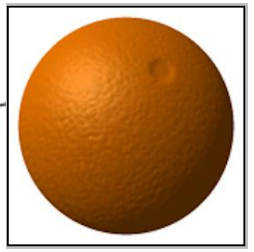
(a) original surface



(b) height map

(c) new surface





(d) perturbed normals

Process:

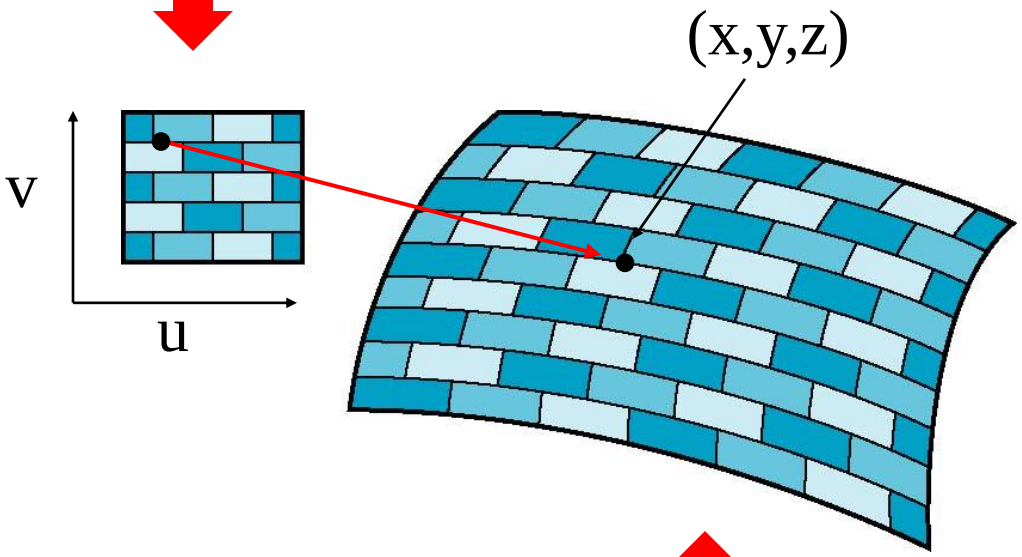
1. Find Texture Local Coord. (u,v,n) on the surface

2. Change normal into u, and v direction

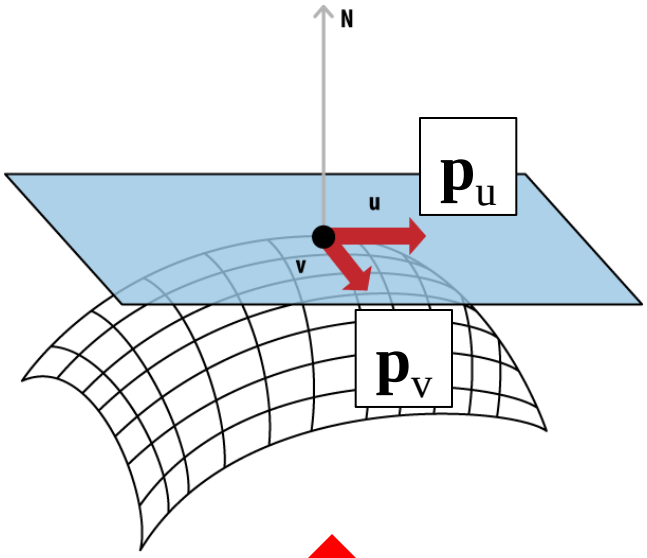
3. Apply the phong shading with the new normal

Tangent Plane: texture coord. in world coord.

Texture uv coord.

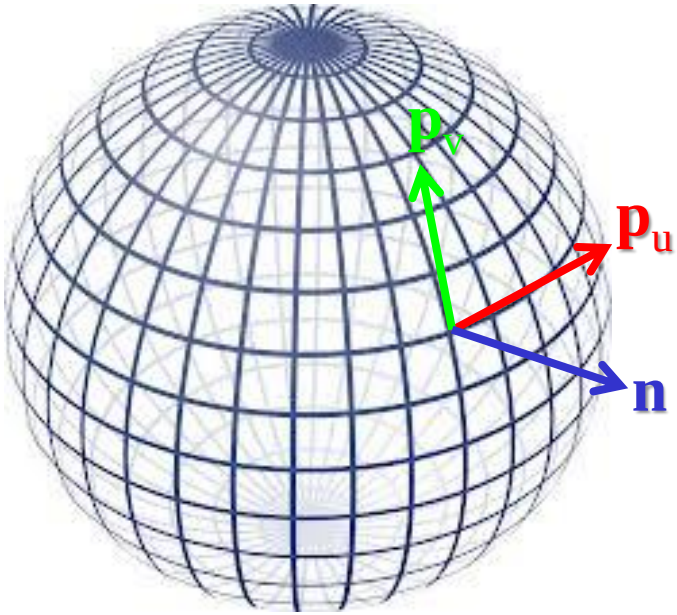


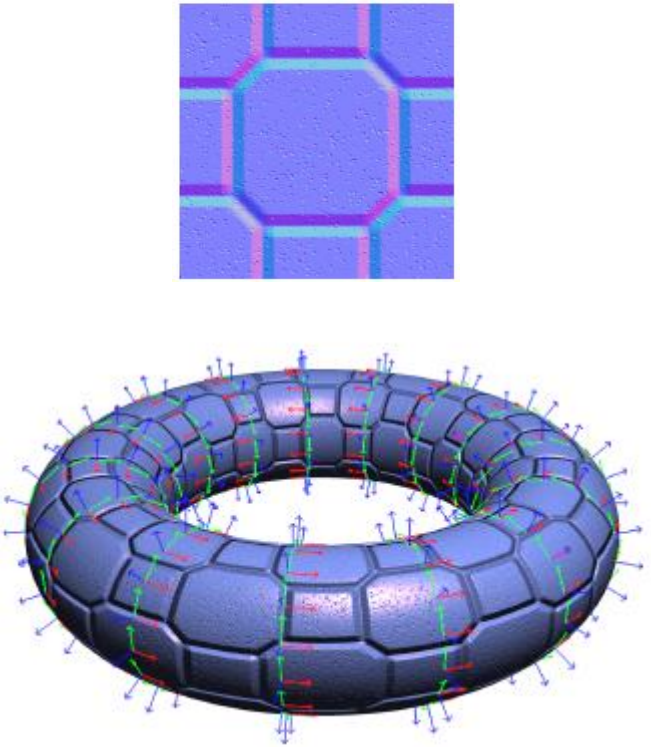
World coord.



Texture coord.
In world coord.

Tangent Coordinate System





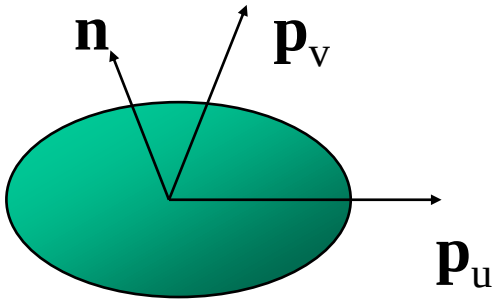
Equations to find the texture plane

좌표: $\mathbf{p}(u,v) = [x(u,v), y(u,v), z(u,v)]^T$

$$\mathbf{p}_u = [\partial x/\partial u, \partial y/\partial u, \partial z/\partial u]^T$$

$$\mathbf{p}_v = [\partial x/\partial v, \partial y/\partial v, \partial z/\partial v]^T$$

$$\mathbf{n} = (\mathbf{p}_u \times \mathbf{p}_v) / |\mathbf{p}_u \times \mathbf{p}_v|$$

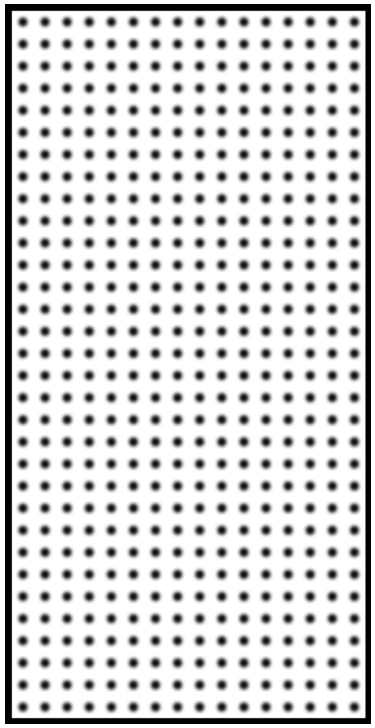


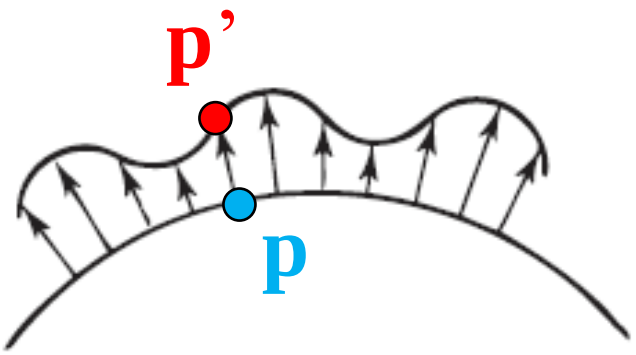
Displacement Function

$$\mathbf{p}' = \mathbf{p} + d(u,v)\,\mathbf{n}$$

d(u,v) is the bump
or displacement (height).

$|d(u,v)| << 1$





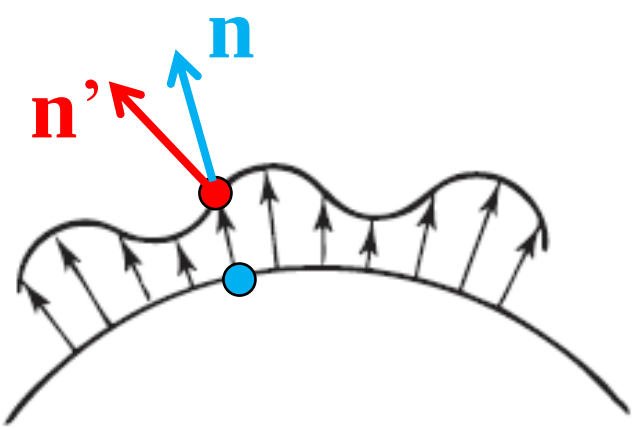
Perturbed New Normal

$$\mathbf{n'} = \mathbf{p'}_u \times \mathbf{p'}_v$$

$$\mathbf{p'}_u = \mathbf{p}_u + (\partial d/\partial u)\mathbf{n} + d(u,v)\mathbf{n}_u$$

$$\mathbf{p'}_v = \mathbf{p}_v + (\partial d/\partial v)\mathbf{n} + d(u,v)\mathbf{n}_v$$

If d is small, we can neglect last term



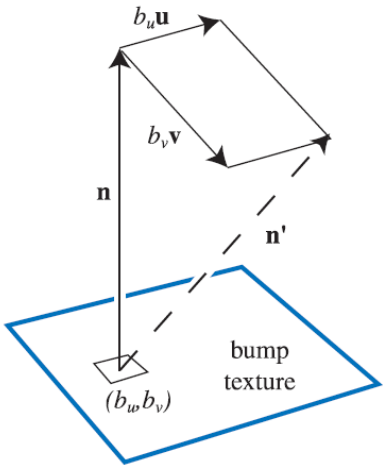
Approximating the Normal

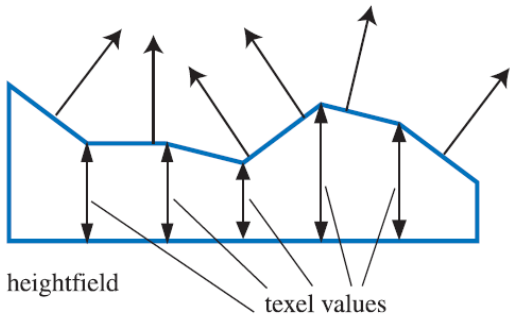
$$\mathbf{n}' = \mathbf{p}'_u \times \mathbf{p}'_v$$

$$\approx \mathbf{n} + (\partial d/\partial u)\mathbf{n} \times \mathbf{p}_v + (\partial d/\partial v)\mathbf{n} \times \mathbf{p}_u$$

$$\approx \mathbf{n} + (\partial d/\partial u)\mathbf{p}_u + (\partial d/\partial v)\mathbf{p}_v$$

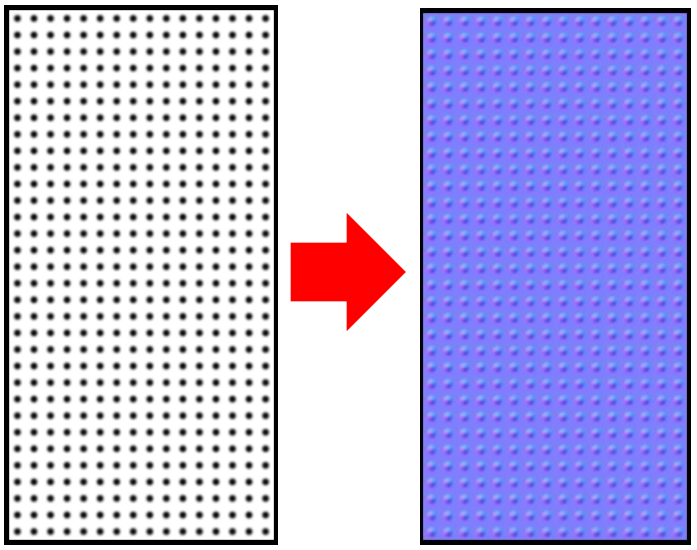
Perturb the normal during shading as much as $(\partial d/\partial u, \partial d/\partial v, 0)$!!





Store the perturbation as an image

- Suppose that we start with a function $d(u,v)$
- We can sample it to form an array $D=[d_{ij}]$
- Then $\partial d/\partial u \approx d_{ij} - d_{i-1,j}$
  and $\partial d/\partial v \approx d_{ij} - d_{i,j-1}$
- Save it as a texture:
  $(\partial d/\partial u\ ,\ 0,\ \partial d/\partial v\ ) \rightarrow (r,g,b)$



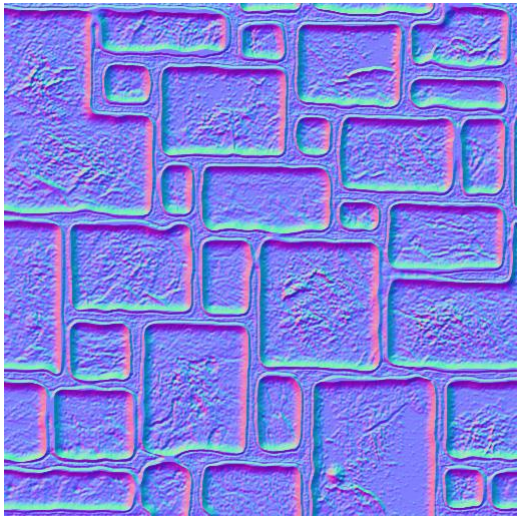
$d(u,v)$

$(\partial d/\partial u,\ 0,\ \partial d/\partial v)$

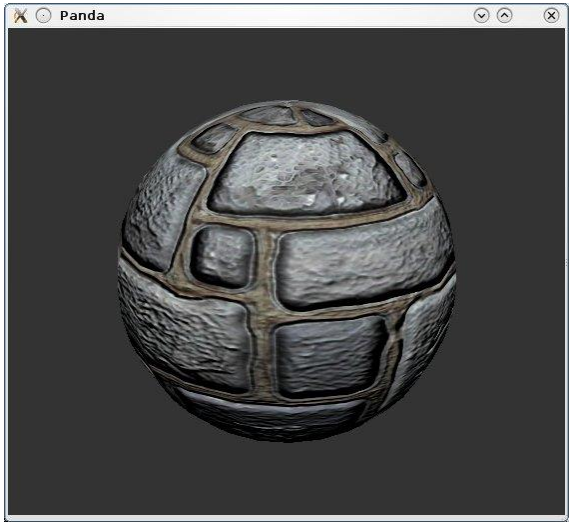
red blue

Example

Regular texture



Normal
Perturbation map

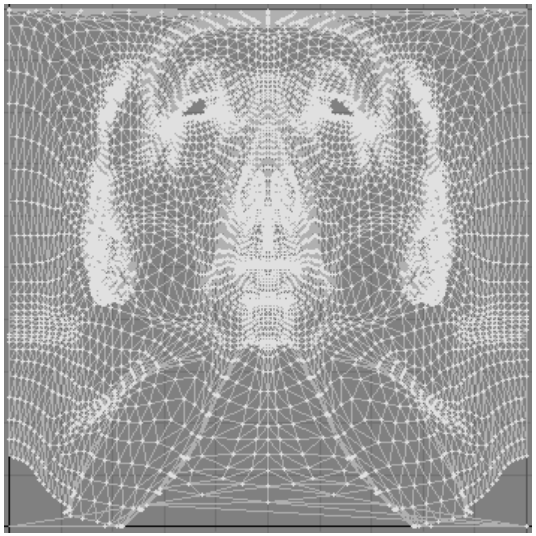


Applied on a sphere

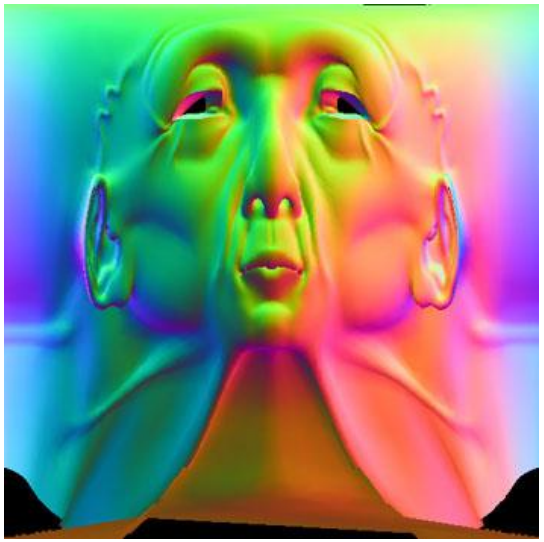
이미지 출처: http://bcchang.com/immersive_blog/?p=589

Normal map

- Compute the normal in the model coord. space (not in the texture tangential coord.)
- Encode it as a texture and override the vertex normal attributes. (x, y, z) ➔ (r, g, b)



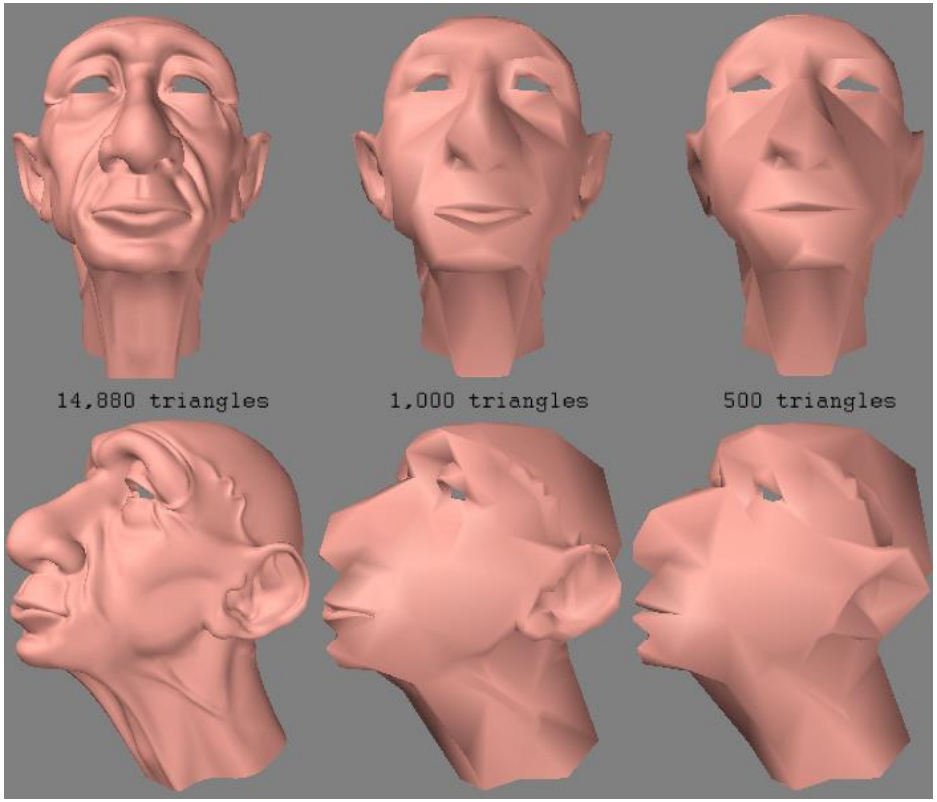
Texture uv coord.

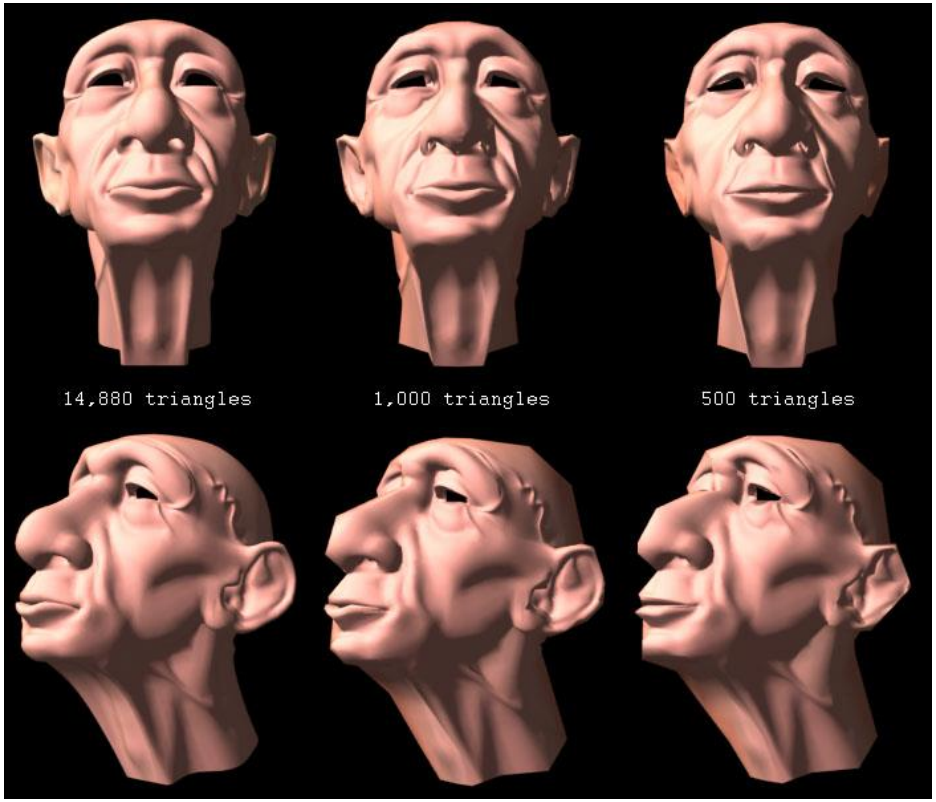


Normal vector as a texture

이미지 출처: http://amber.rc.arizona.edu/lw/normalmaps.html

Normal map results:



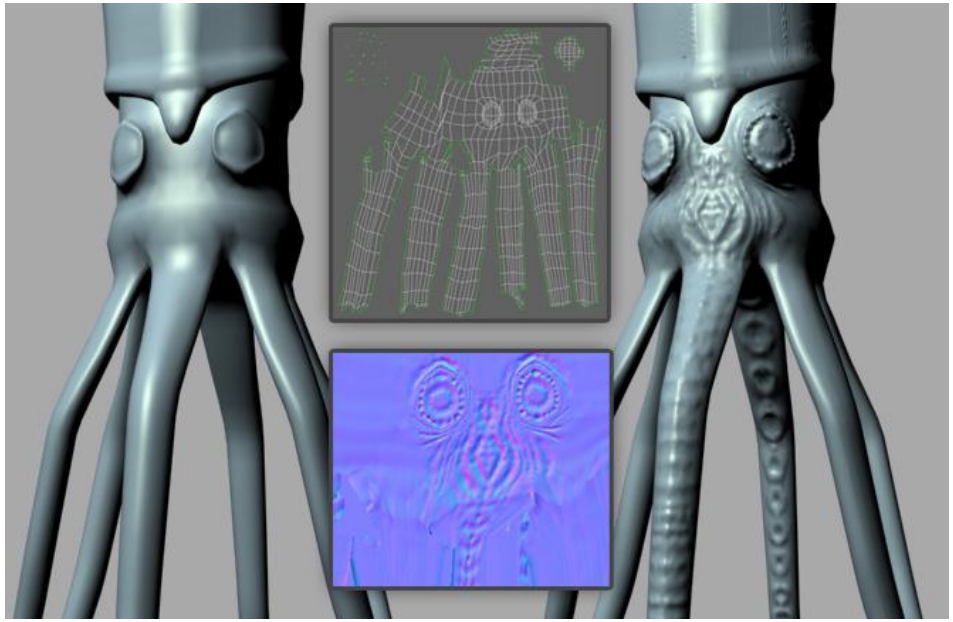


Before applying normal map

After applying normal map

이미지 출처: http://amber.rc.arizona.edu/lw/normalmaps.html

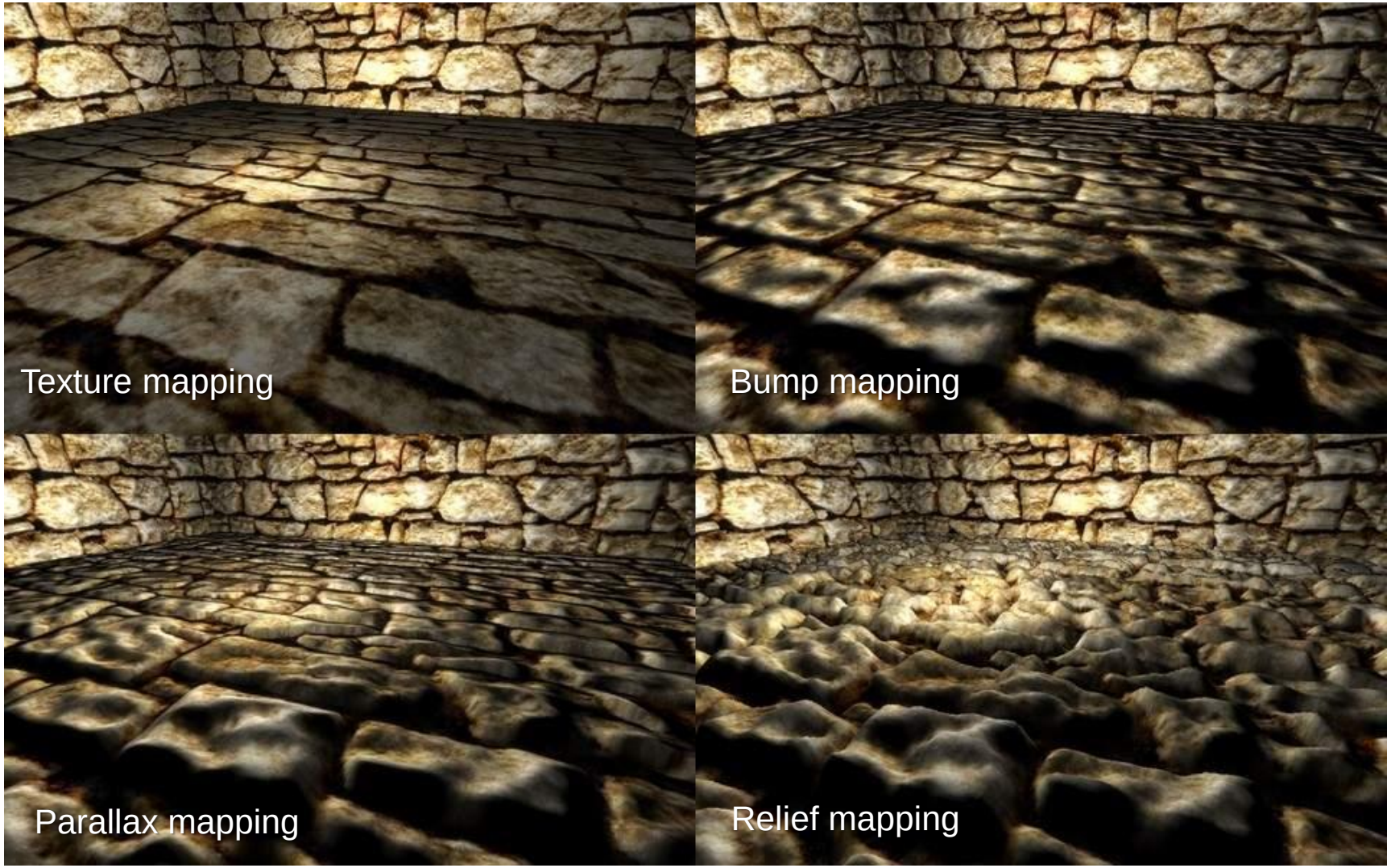
Normal map results:



이미지 출처:
http://tom.drastic.net/stuff/mudbox--3ds-max/

이미지 출처:
http://ve3d.ign.com/images/fullsize/46667/PC/Crysis/Screenshots/CryEngine-3-Image

The State-of-the-art techniques:



이미지 출처: http://www.esenthel.com/?id=feature/rendering