Texture Mapping

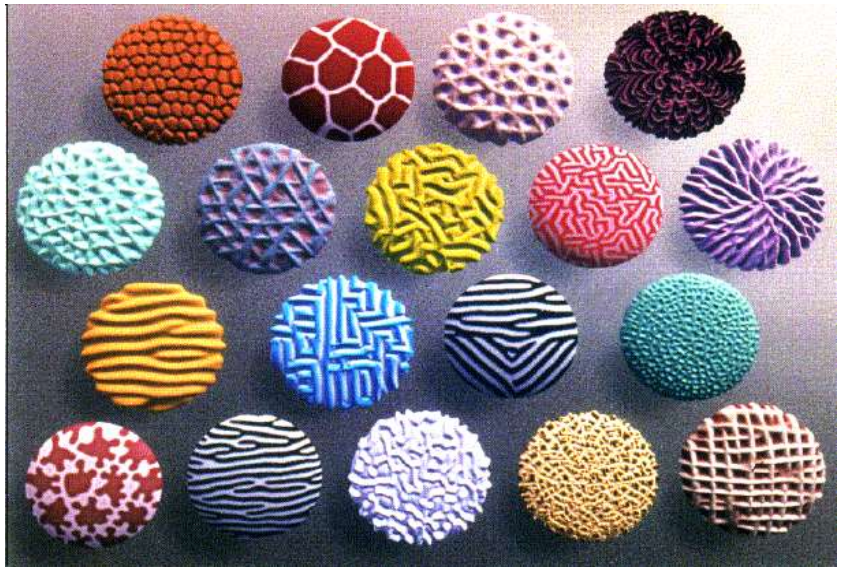
COLLEGE OF COMPUTING

HANYANG ERICA CAMPUS

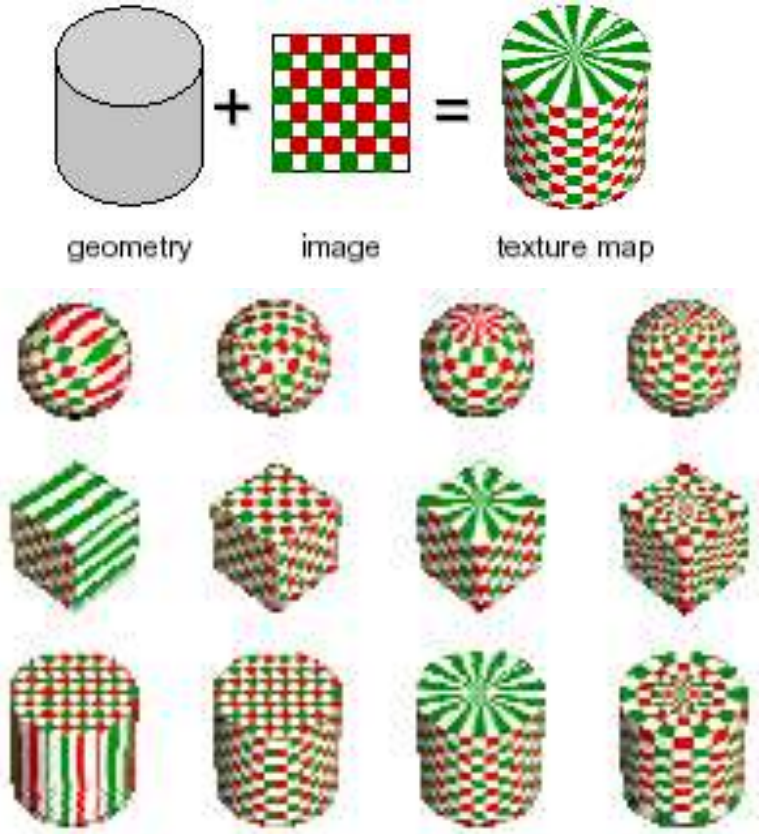
Q YOUN HONG (홍규연)

Texture Mapping

- 현실의 물체들은 색이나 법선 방향등이 균일하지 않음
- High frequency 디테일은 모델링하기 어려움
- 사실적인 shading을 위해서 색상, 법선에 변화를 주는 것이 필요함 → **Texture (텍스쳐)** 사용함
- 복잡한 디테일의 모델링을 대체할 수 있음



Texture Mapping



Texture Mapping

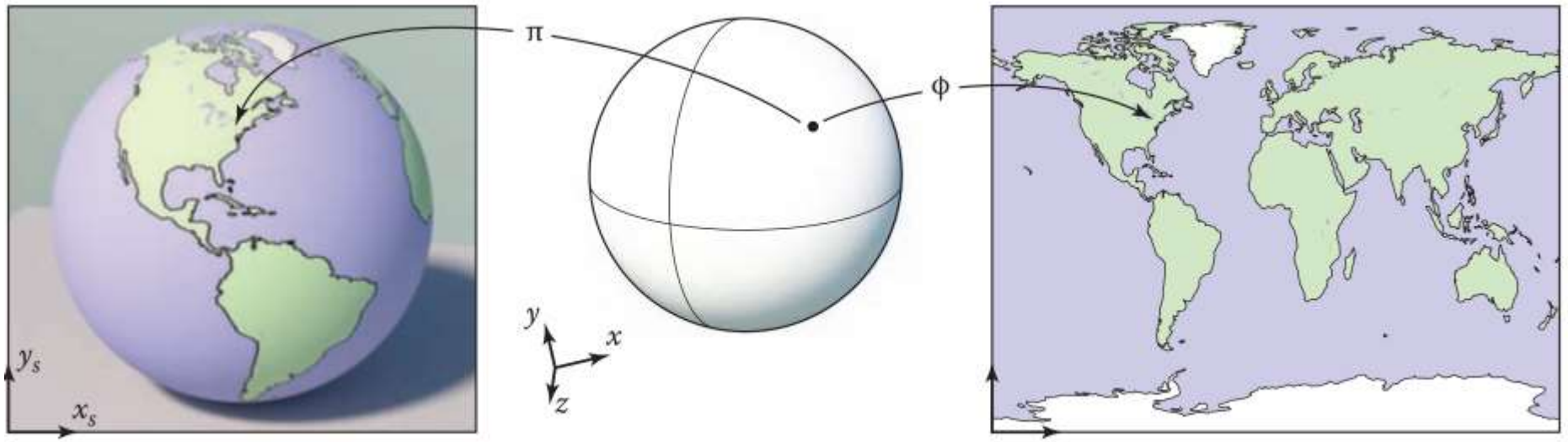
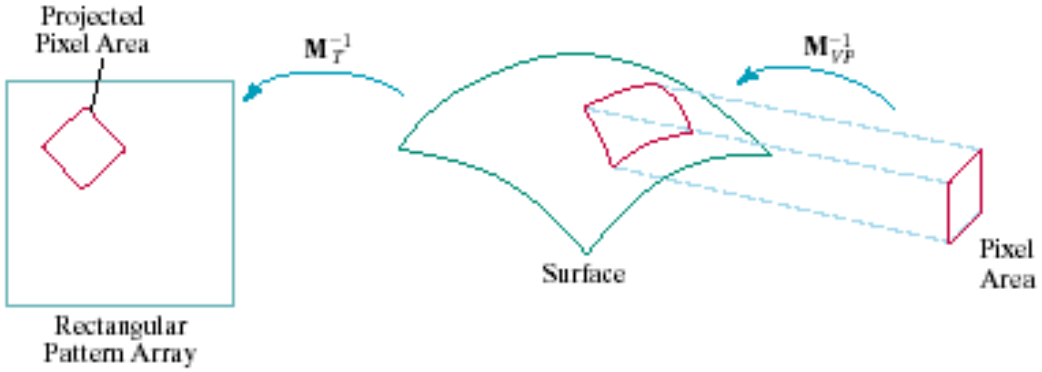


Image space　　Surface S in world space　　Texture space, T

- $\Phi = M_T^{-1}: S \rightarrow T$
  $: (x, y, z) \rightarrow (s, t)$
  Surface S에서 Texture T로 가는 매핑
- $\Pi = M_{VP}$: viewing projection



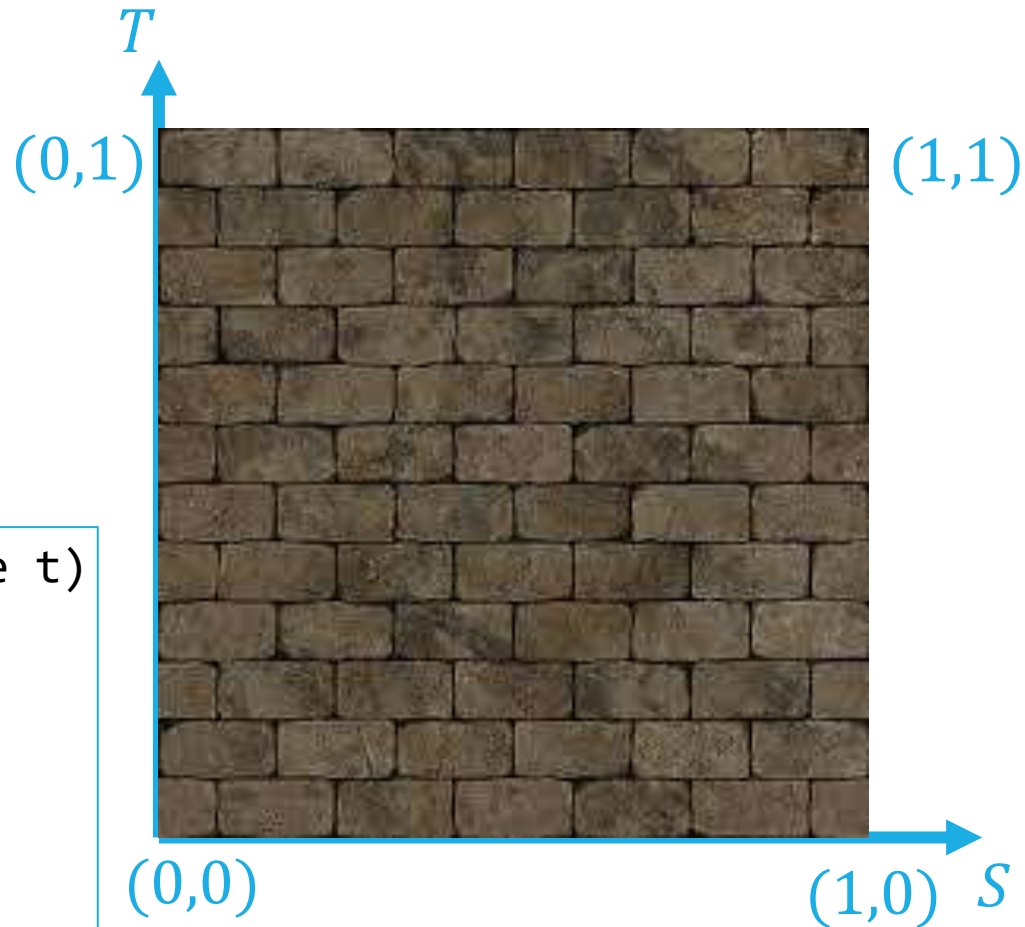
Looking Up Texture Values

- Surface의 texture는 texture coordinate $(s,t), s,t \in [0,1]$에 의해 정의
- Texture lookup function

```
Color texture_lookup(Texture t, float s, float t)
{
    int i = round(s * t.width() – 0.5 );
    int j = round(t * t.height() – 0.5);
    return t.get_pixels(i,j);
}
```

- Shading surface with a texture

```
Color shade_surface_point(Surface s, Point p, Texture t)
{
    Vector normal = s.get_normal(p);
    (s,t) = s.get_texcoord(p);
    Color diffuse_color = texture_lookup(s,t);
    //compute shading using diffuse_color and normal
    //return shading result
}
```



Texture Coordinate Functions

- Q) Surface에서 Texture로 매핑하는 함수 Φ를 어떻게 정의하는가?

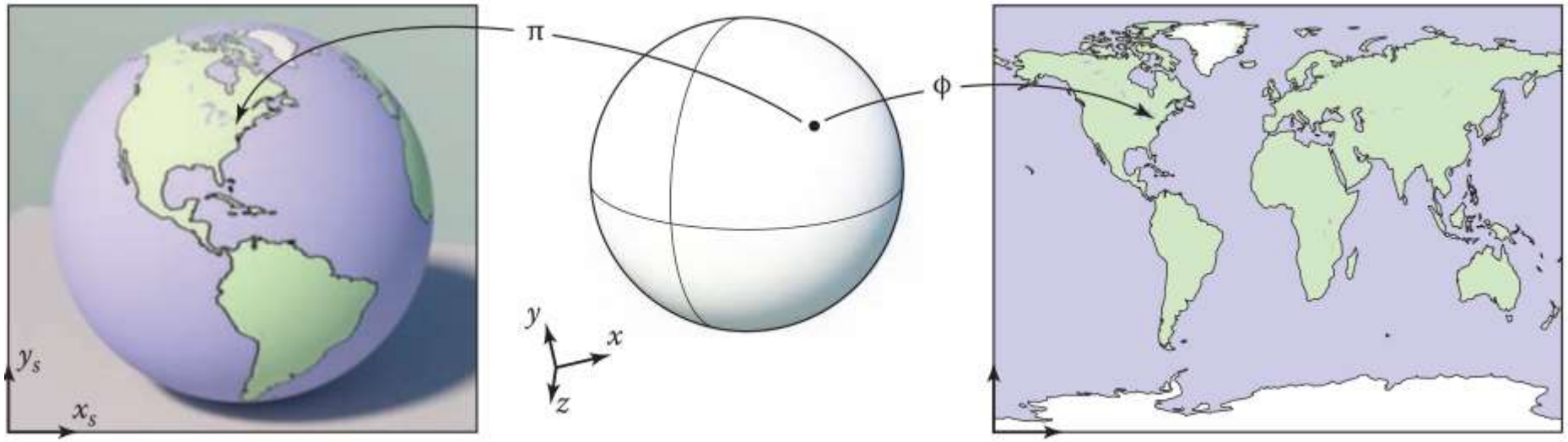


Image space　　Surface *S* in world space　　Texture space, *T*

Texture Coordinate Functions

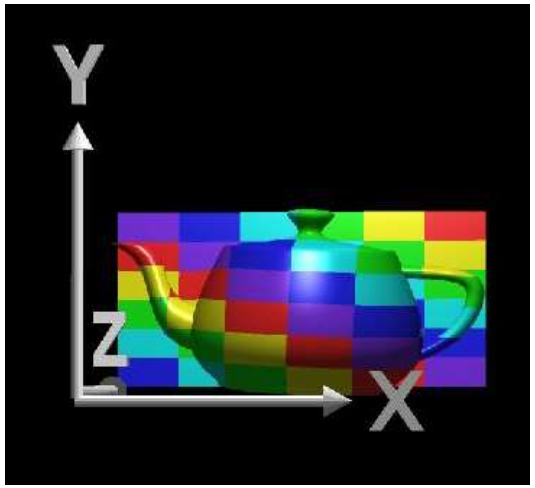
- Surface가 평면인 경우 (예: 벽, 바닥 등등)
  - $\Phi(x, y, z) = (\mathrm{ax}, \mathrm{by})$ (xy-평면에 평행인 바닥의 경우)

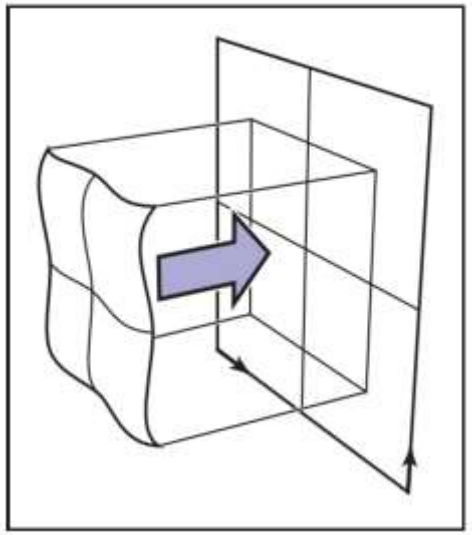
- Surface가 parametric surface (예: Bezier surface, B-spline surface)인 경우
  - Surface의 parameter인 (u, v)를 texture coordinate로 사용할 수 있음

- 일반적인 Surface의 경우, 다음의 특징을 가지는 texture coordinate functions을 정의해야 함
  - Bijectivity: surface의 각 점은 texture위의 다른 점으로 mapping
  - Minimizaing distortion: mapping된 texture의 scale이 일정해야 함
  - Continuity: texture는 되도록 연속적으로 mapping

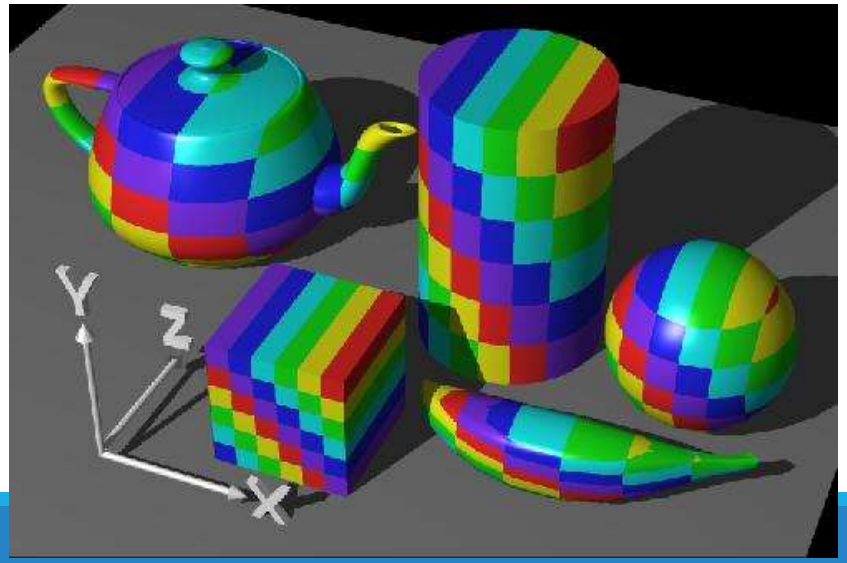
Texture Coordinate Functions

- Planar Projection





$$\Phi(x, y, z) = (s, t), \text{where} \begin{bmatrix} s \\ t \\ * \\ 1 \end{bmatrix} = M_t \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

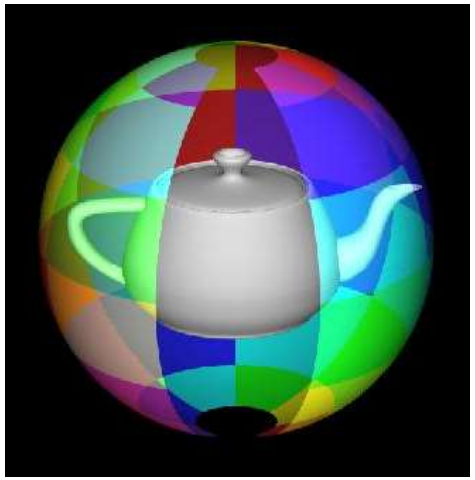


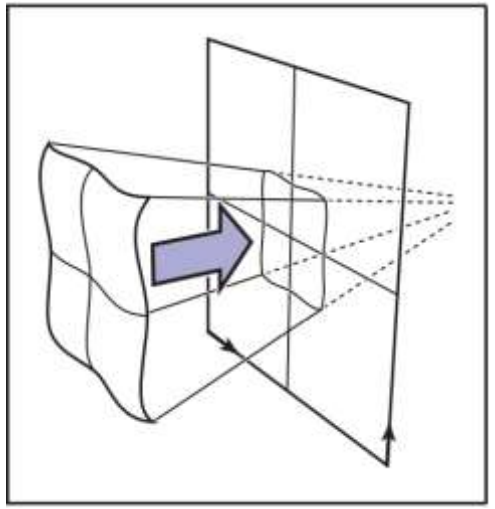
Φ

09	19	29	39	49	59	69	79	89	99
08	18	28	38	48	58	68	78	88	98
07	17	27	37	47	57	67	77	87	97
06	16	26	36	46	56	66	76	86	96
05	15	25	35	45	55	65	75	85	95
04	14	24	34	44	54	64	74	84	94
03	13	23	33	43	53	63	73	83	93
02	12	22	32	42	52	62	72	82	92
01	11	21	31	41	51	61	71	81	91
00	10	20	30	40	50	60	70	80	90

Texture Coordinate Functions

- Spherical Coordinates

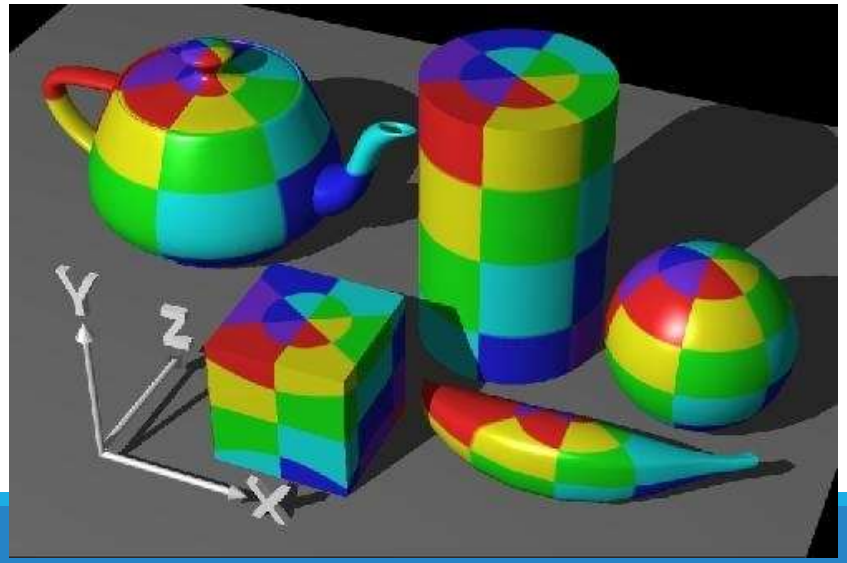




Spherical Coordinate:

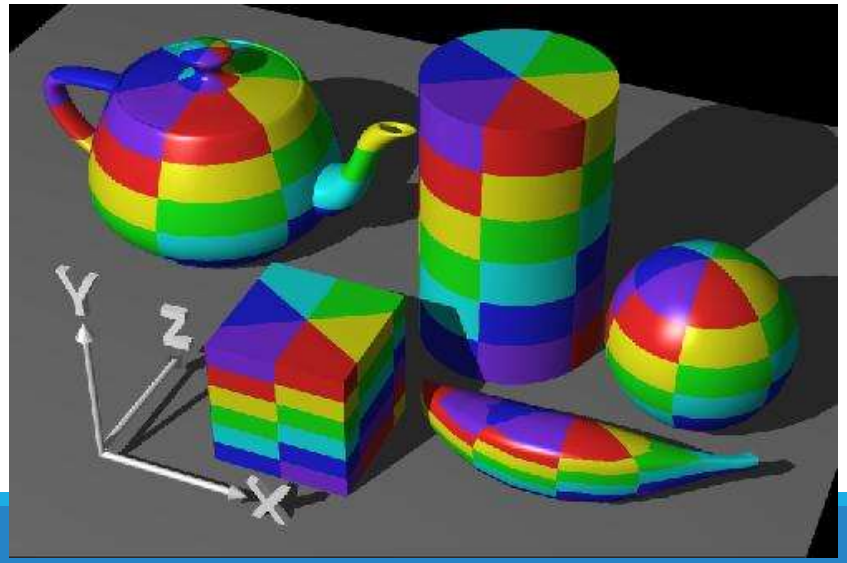
$$\Phi(x,y,z) = \left(\frac{\pi + \text{atan2}(y,x)}{2\pi}, \frac{\pi - \text{acos}\left(\frac{z}{\|x\|}\right)}{\pi}\right)$$

- 곡면의 구면좌표 (ρ, θ, φ)에서 θ, φ를 [0,1]로 mapping
- 경도, 위도 mapping
- 극점(pole)들을 제외하면 bijective mapping



Texture Coordinate Functions

- Cylindrical Coordinates

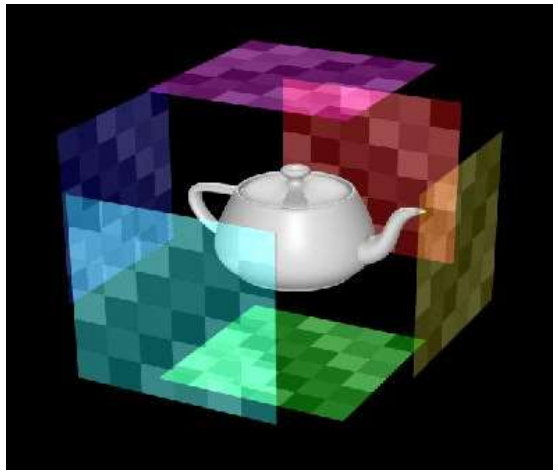


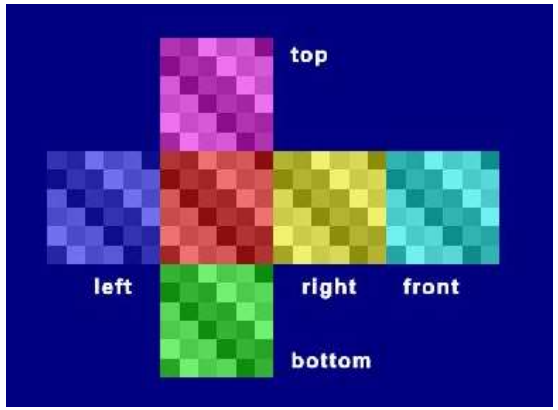
Cylindrical Coordinate:

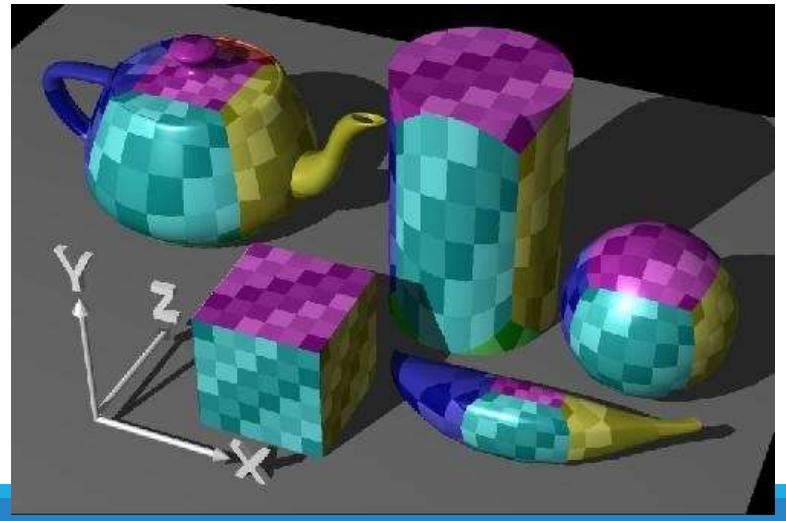
$$\Phi(x, y, z) = \left(\frac{\pi + \text{atan2}(y,x)}{2\pi}, \frac{1+z}{2}\right)$$

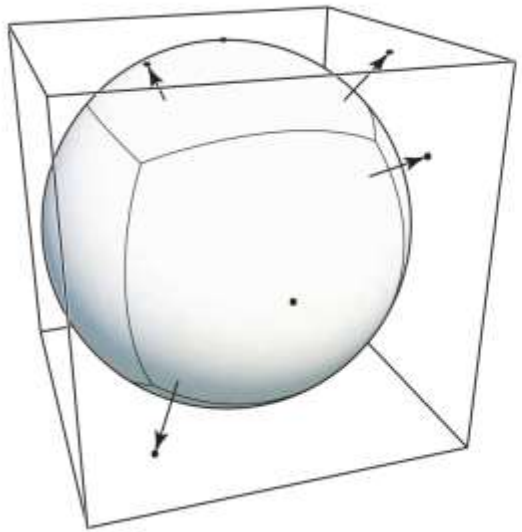
Texture Coordinate Functions

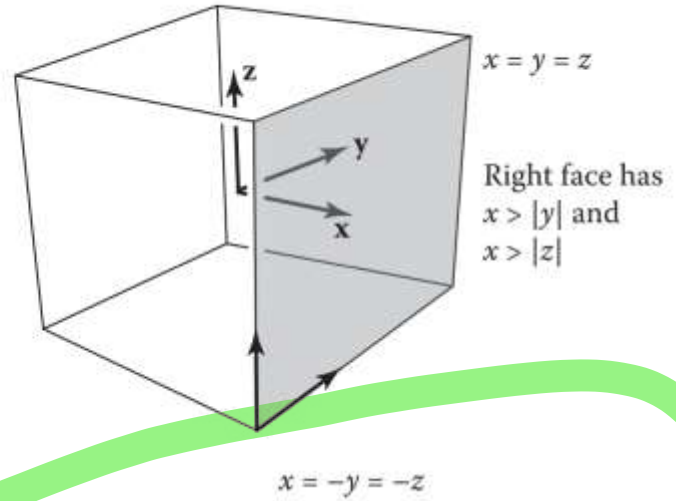
- Cubemap







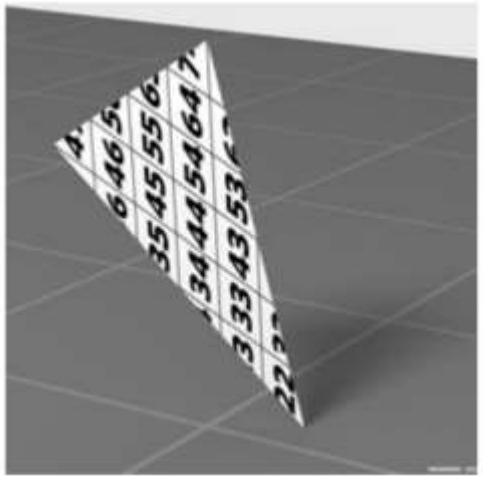




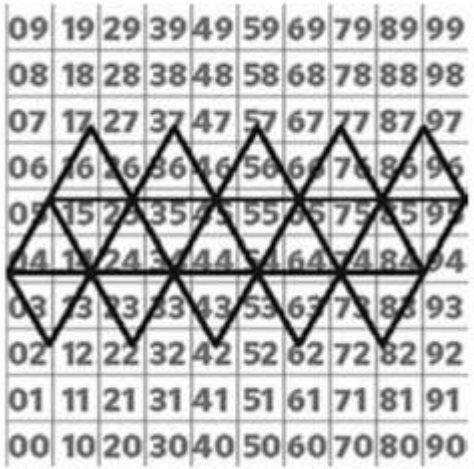
$$\phi_{-x}(x,y,z) = \tfrac{1}{2}\left[1 + (+z,-y)/|x|\right],$$
$$\phi_{+x}(x,y,z) = \tfrac{1}{2}\left[1 + (-z,-y)/|x|\right],$$
$$\phi_{-y}(x,y,z) = \tfrac{1}{2}\left[1 + (+x,-z)/|y|\right],$$
$$\phi_{+y}(x,y,z) = \tfrac{1}{2}\left[1 + (+x,+z)/|y|\right],$$
$$\phi_{-z}(x,y,z) = \tfrac{1}{2}\left[1 + (-x,-y)/|z|\right],$$
$$\phi_{+z}(x,y,z) = \tfrac{1}{2}\left[1 + (+x,-y)/|z|\right].$$

Interpolated Texture Coordinates

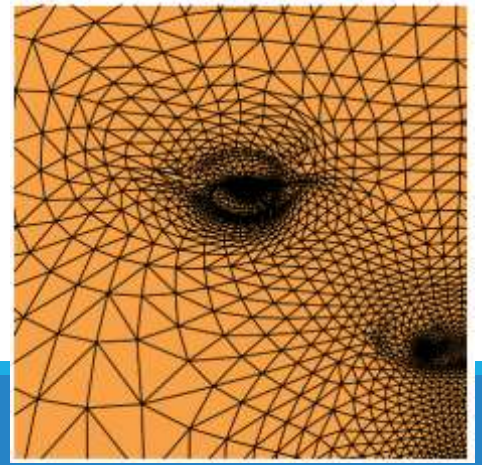
- Use barycentric interpolation using texture coordinates stored in vertices

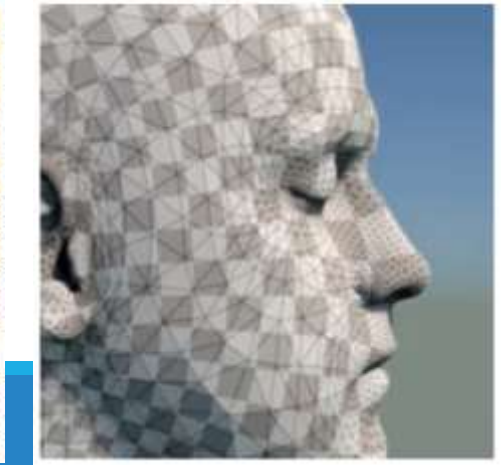


For a single triangle



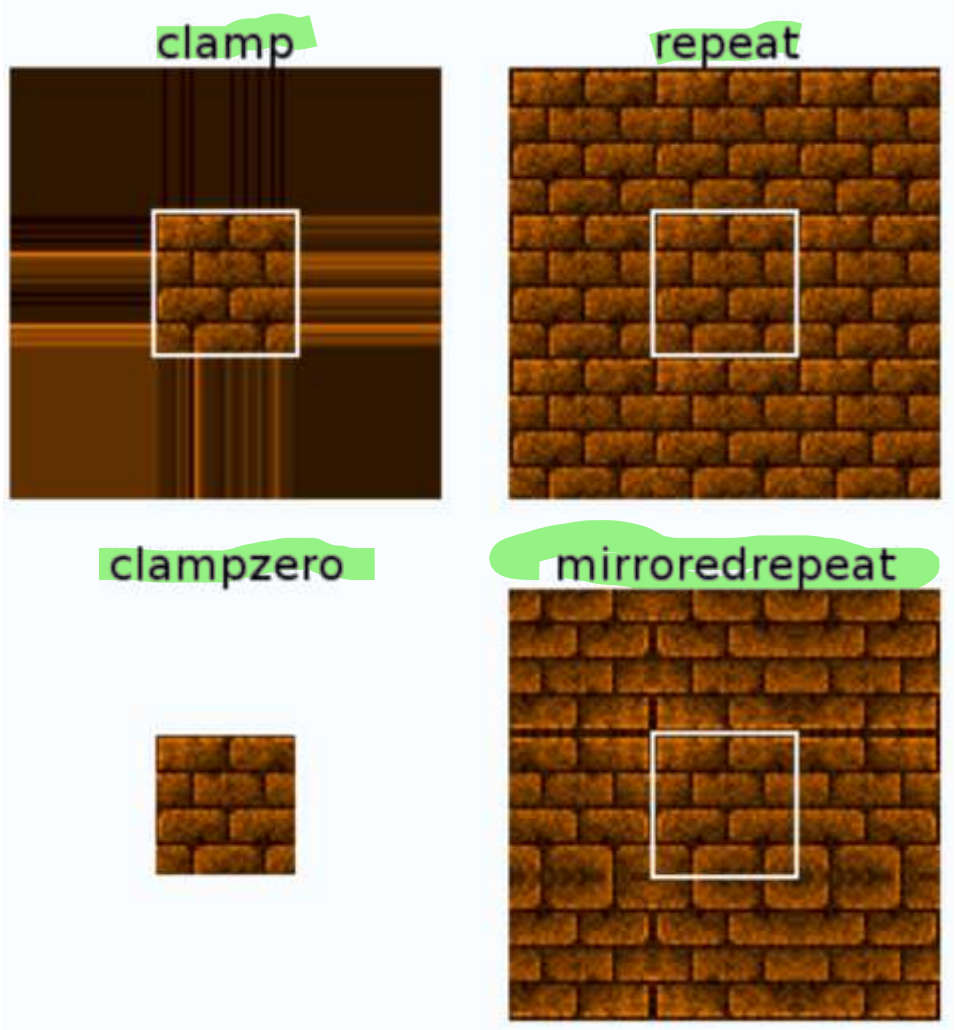
For an icosahedron





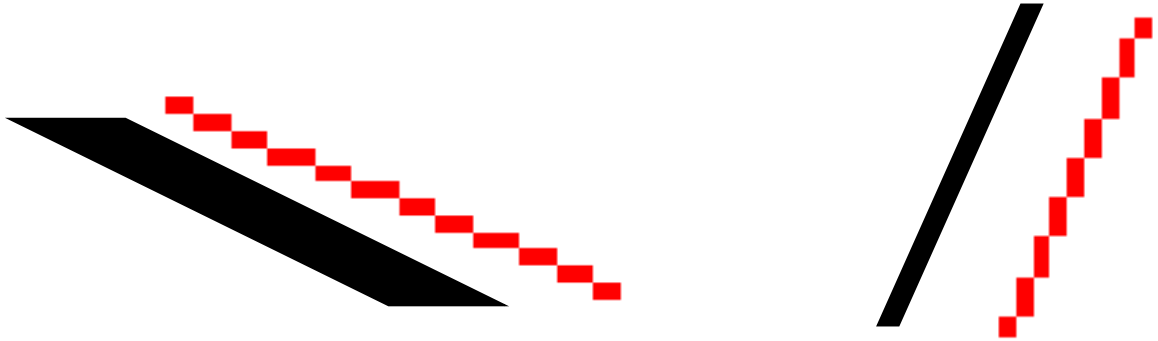
For a face model
: Texture coordinates are assigned to reduce size distortion

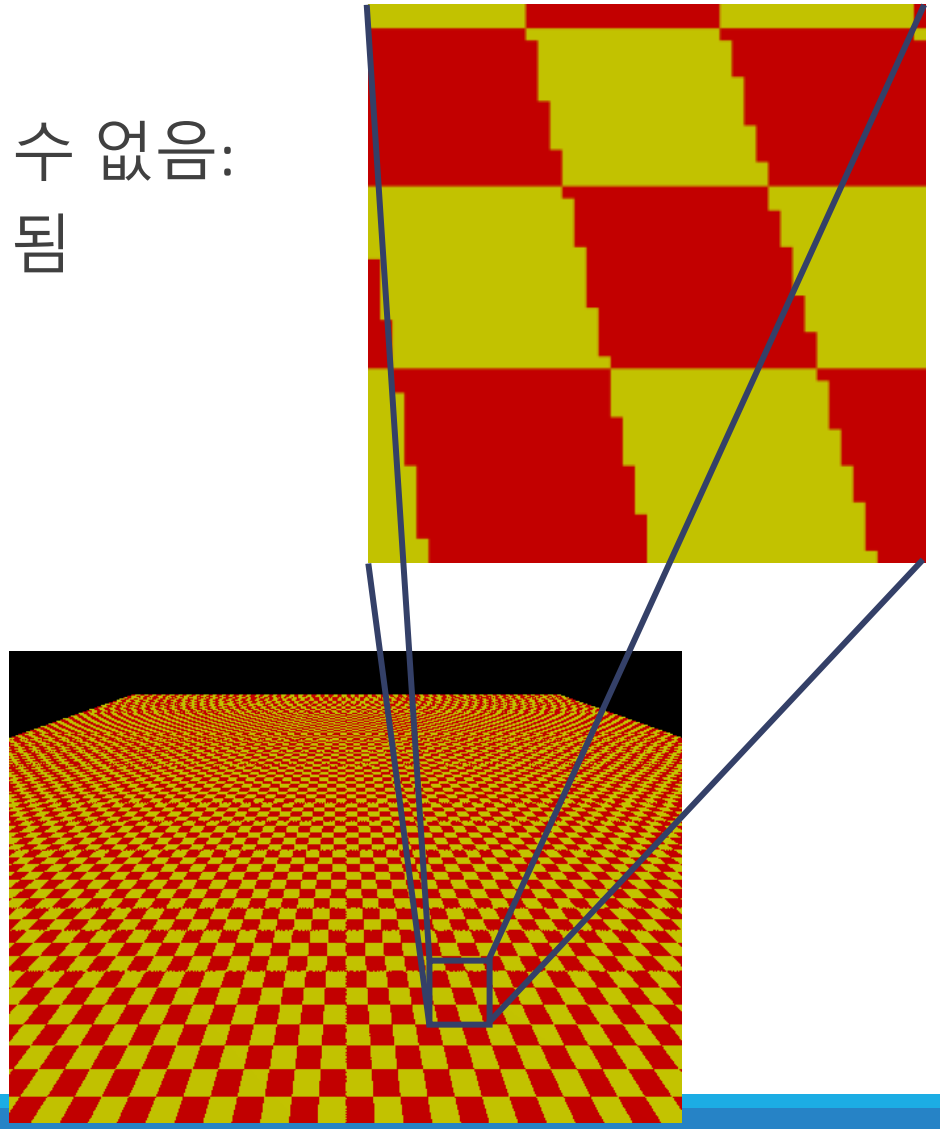
Texture Wrapping Modes



Antialiasing Texture Lookup

- Aliasing
  - Discrete raster device에선 부드러운 선을 그릴 수 없음: 계단형의(staircased) 라인 (jaggies)들을 그리게 됨



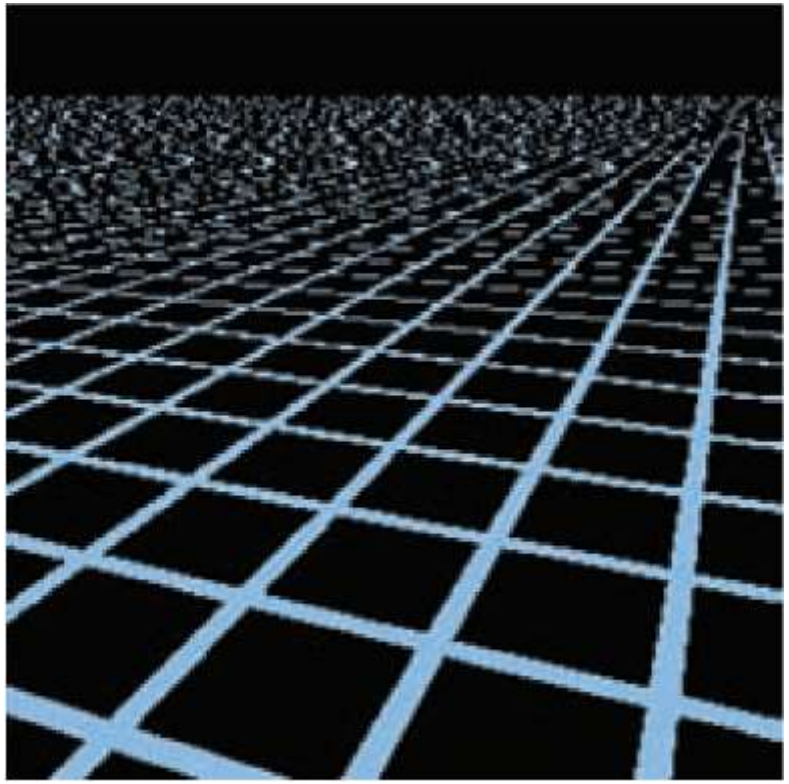


- 너무 미세한 detail들은 artifact들을 생성함 (Moire patterns)

Antialiasing Texture Lookup

- Aliasing in Texture Mapping
  - ⇒ 텍스쳐 매핑된 이미지를 그리는 것은 샘플링임
  - ⇒ 텍스쳐 매핑된 곡면을 2D 이미지 화면에 투영한 후, 각 pixel에서 샘플링

- high contrast된 texture를 기울여서 (저해상도의) 이미지 화면에서 렌더링할 경우, aliasing artifact (staircased pixels, moire patterns)이 보임

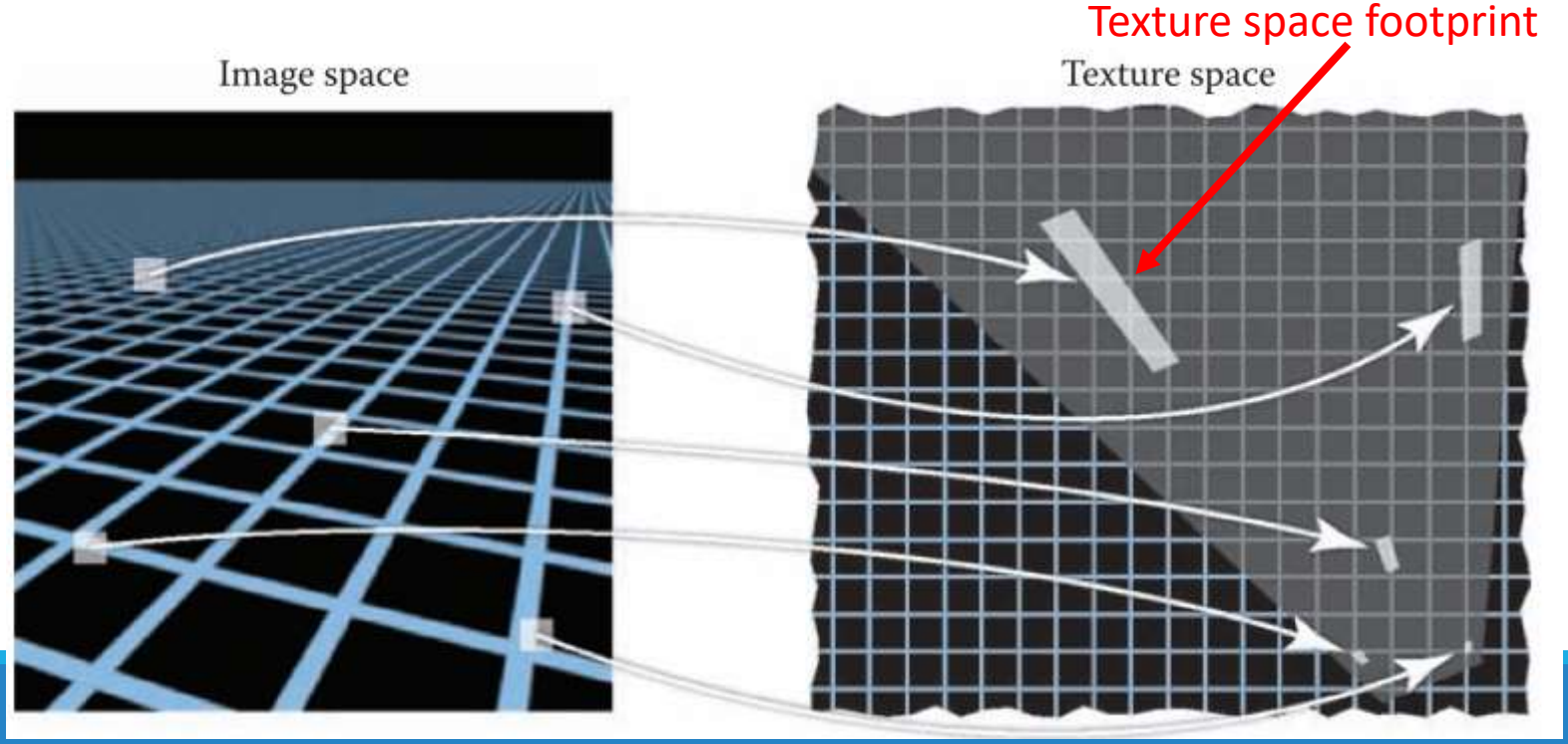


Antialiasing Texture Lookup

- How to resolve aliasing artifacts in image?
  - Supersampling: 각 pixel을 칠할 때, point sample대신, point 주변의 area의 평균 값으로 pixel의 색상을 결정
  - Antialiasing textures is more complex!!
    - Rendered image와 texture와의 관계가 물체의 형태와 화면에서의 위치에 따라 계속 변함

- π: 3D point → image space
- ϕ: 3D point → texture space
- $\varphi = \phi \circ \pi^{-1}$: image pixel → texture space pixel
- Supersampling in textures: $\varphi(\Delta x, \Delta y)$의 평균값을 구해야함

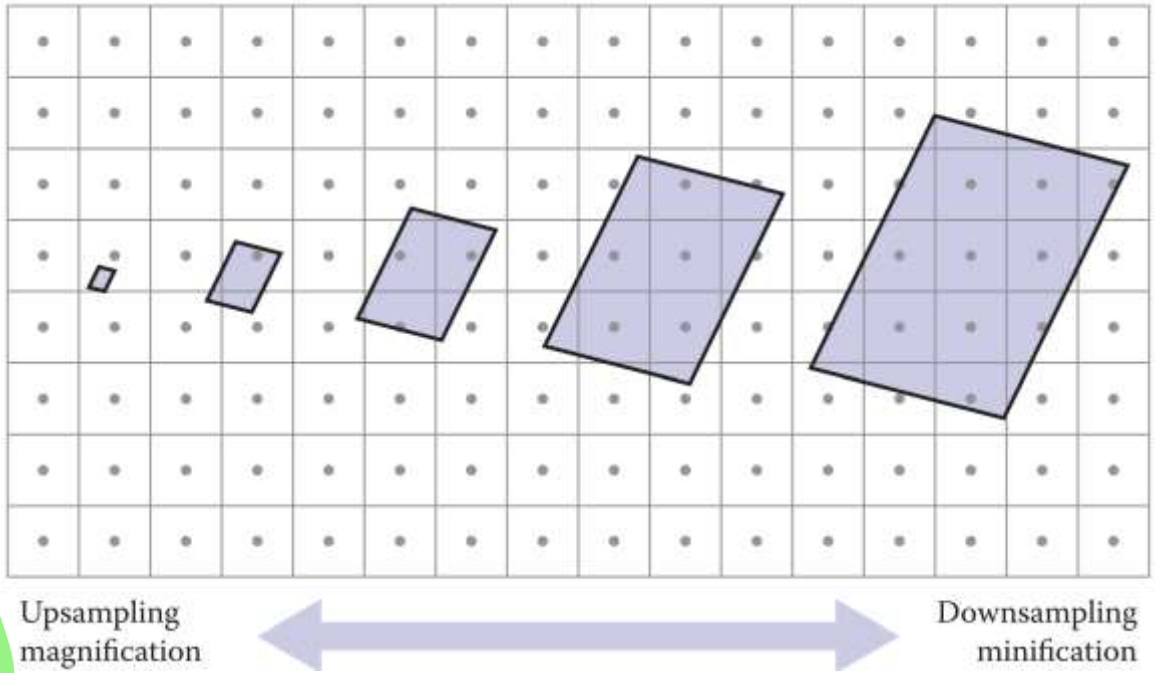
⇒ Approximation 사용!



Antialiasing Texture Lookup

- Reconstruction
  - Texel보다 texture space footprint의 크기가 작을 때: texel들 간의 interpolation 적용
  - Texel보다 texture space footprint의 크기가 클 때: 여러 texel들 간의 평균 값을 효율적으로 계산하는 것이 중요

```
Color tex_sample_bilinear(Texture t, float u, float v) {
    u_p = u * t.width - 0.5
    v_p = v * t.height - 0.5
    iu0 = floor(u_p); iu1 = iu0 + 1
    iv0 = floor(v_p); iv1 = iv0 + 1
    a_u = (iu1 - u_p); b_u = 1 - a_u
    a_v = (iv1 - v_p); b_v = 1 - a_v
    return a_u * a_v * t[iu0][iv0] + a_u * b_v * t[iu0][iv1] +
           b_u * a_v * t[iu1][iv0] + b_u * b_v * t[iu1][iv1]
}
```



Antialiasing Texture Lookup

- Mipmapping
  - Mipmap – 같은 이미지를 downsampling해서 저장한 textures들의 집합

Ex) Level-0: 512 x 512 (original)

Level-1: 256 x 256

Level-2: 128 x 128…

  - Texture filtering with mipmaps

: pixel footprint의 크기와 비슷한 texel을 가진 mipmap에서 pixel 값을 읽음

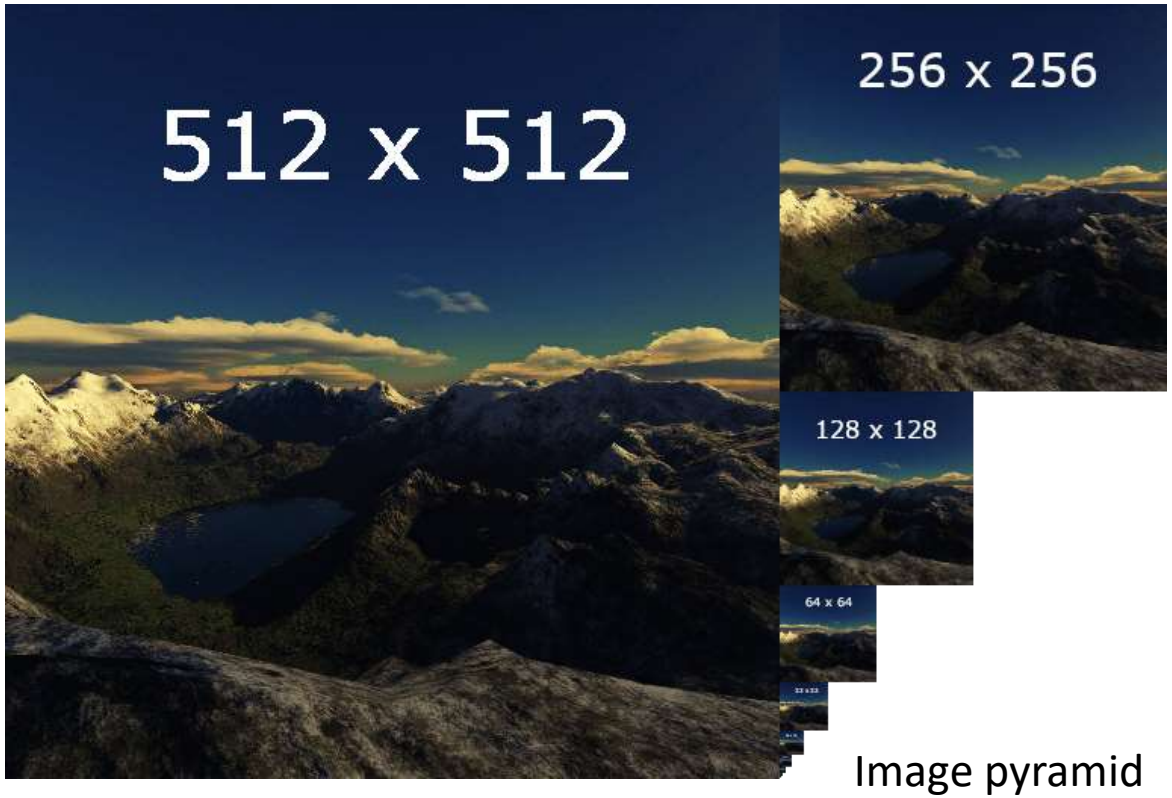


Image pyramid

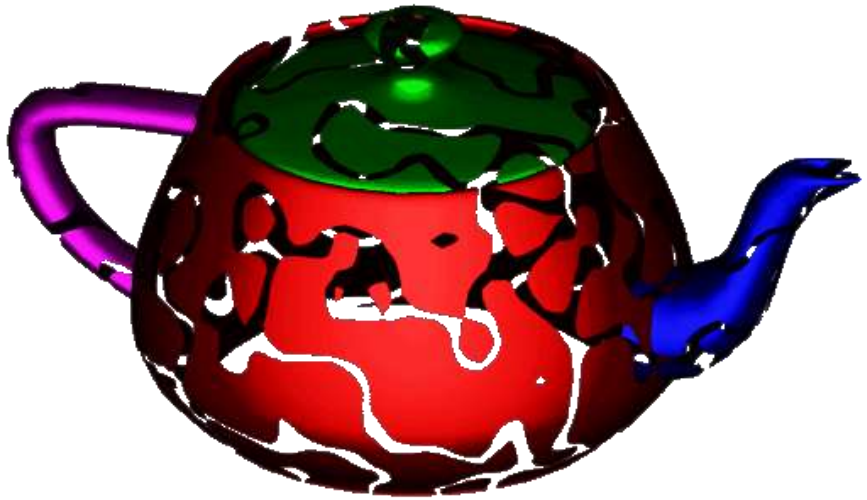
Texture Mapping Applications

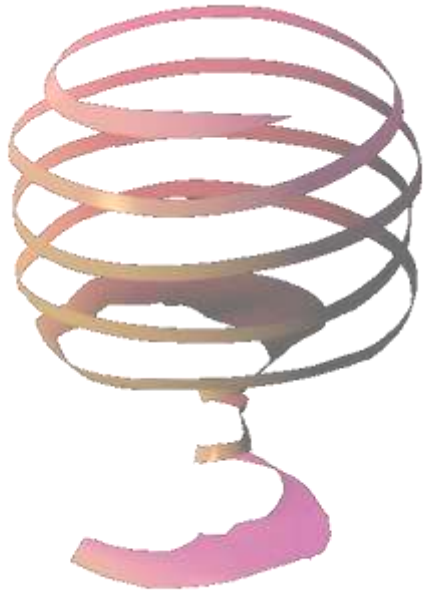
Color와 intensity이외의 여러 material/object properties들 또한 texture mapping을 통해 제어할 수 있음

- Reflectance (diffuse or specular color)
- Surface normal (bump mapping) and (normal) displacement
- Transparency
- Reflected color (environment mapping)

Controlling Shading Parameters

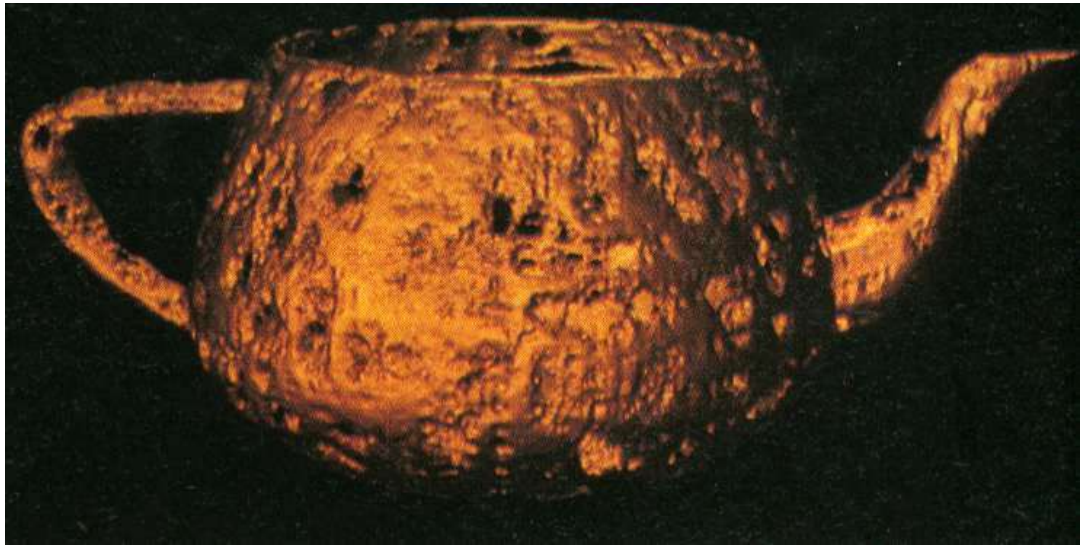
- Controlling Shading Parameters
  - Diffuse color, specular color, specular roughness 등을 texture로 부터 읽음
  - 예: 테이프가 붙어있는 박스의 rendering, 스티커가 붙어 있는 컵의 표현
- Transparency/Opacity mapping





Normal – Bump Mapping

- Object surface는 보통 부드럽지 않음 – 이를 재현하기 위해서 복잡한 기하 모델이 필요
- Object shape의 normal을 국소적으로 흔들어서 표면의 재질 표현
  - Random perturbation
  - Directional change over region

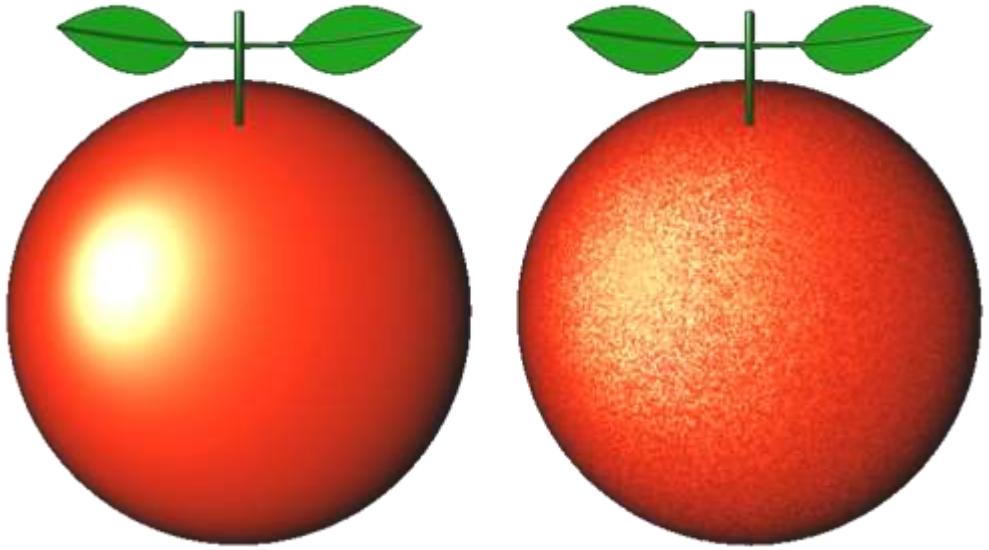


Normal – Bump Mapping

- Shading equation:

$$I = I_a k_a + I_p (k_d \left(\left(N + N(x, y, z) \right) \cdot L \right) + k_s (R \cdot V)^n)$$

- 위의 식에서 R 또한 (perturbed) normal을 포함



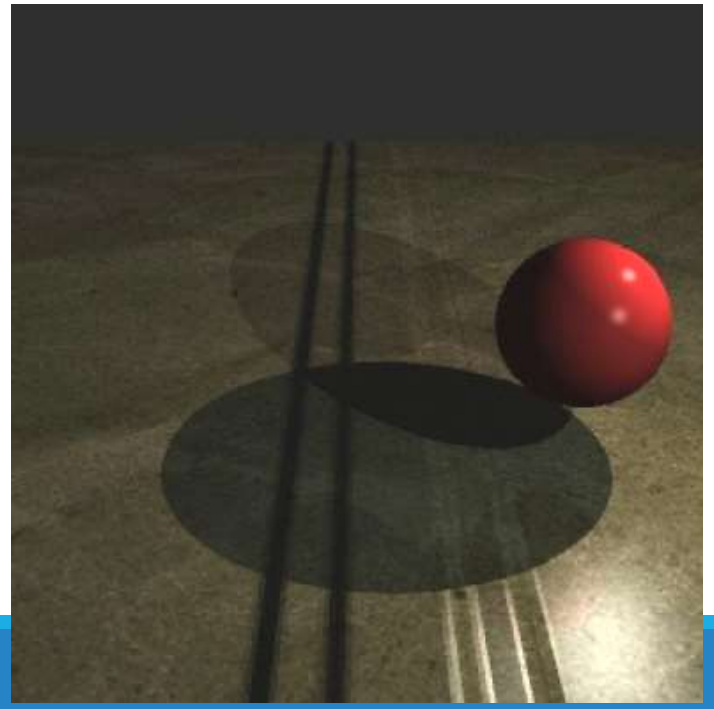
Normal – Bump Mapping

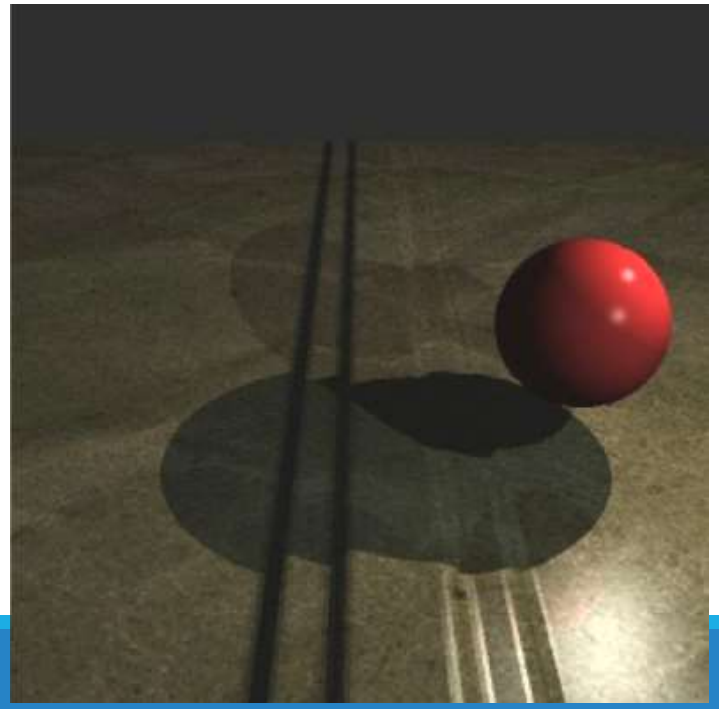
Bump/Displacement Mapping

- Normal/Bump map은 실제 surface의 형태를 변화시키지 않는 shading trick

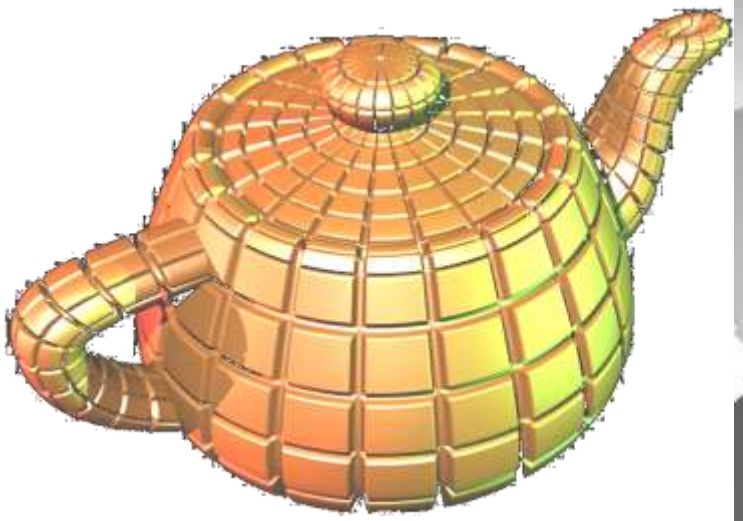
⇒실루엣을 보면 여전히 부드럽게 보임

- Displacement mapping: normal 방향의 height map 저장하고 실제로 surface geometry에 영향을 줌



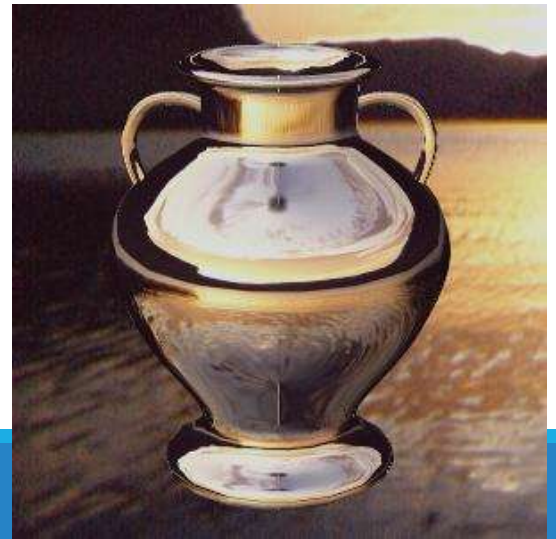


Displacement Mapping



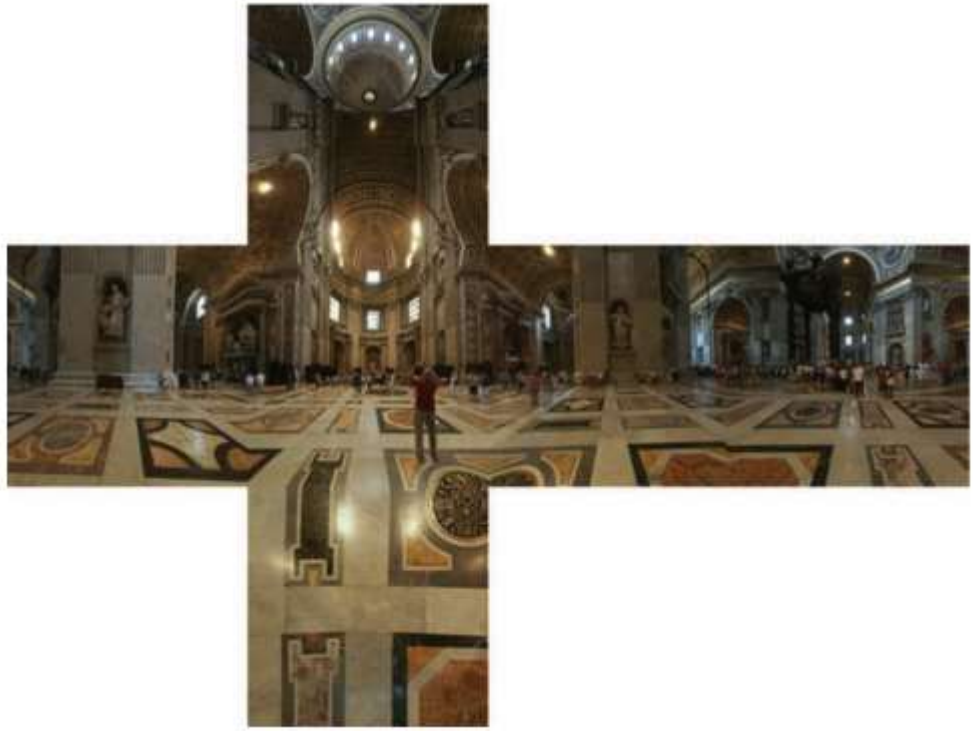
Environmental Mapping

- 공간에서 반사되어 나오는 빛도 texture로 저장



Environmental Mapping

- 반사 효과를 나타내는 가장 저렴한 방법
  - 주변을 파노라마 이미지로 생성
  - 이미지를 object에 texture로 mapping
  - 사람의 눈은 reflectance에 대해 둔감함



Volumetric Textures

- Texture patter을 3D domai에서 정의
  - Digitized or procedural texture function
  - Object의 각각의 점에 대해 3D locatio으로 texture 계산
- natural material/irregular texture (stone, wood..)