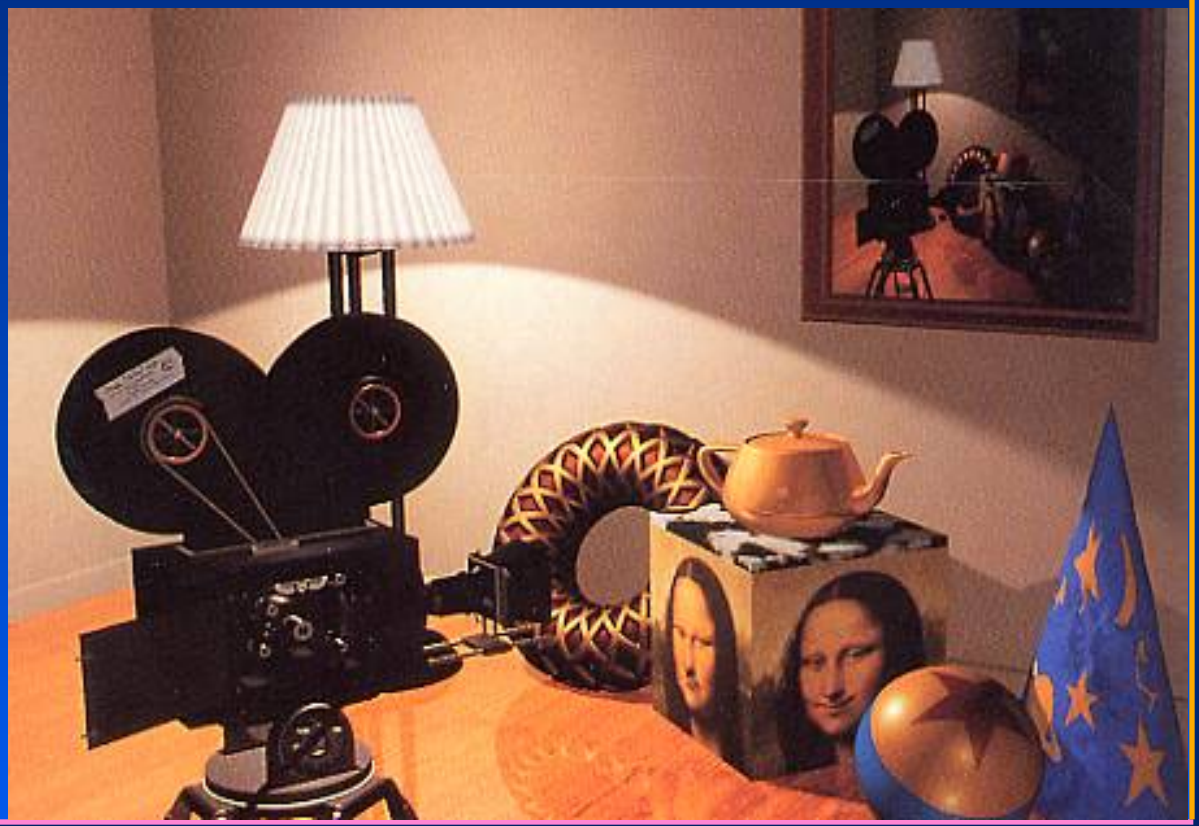
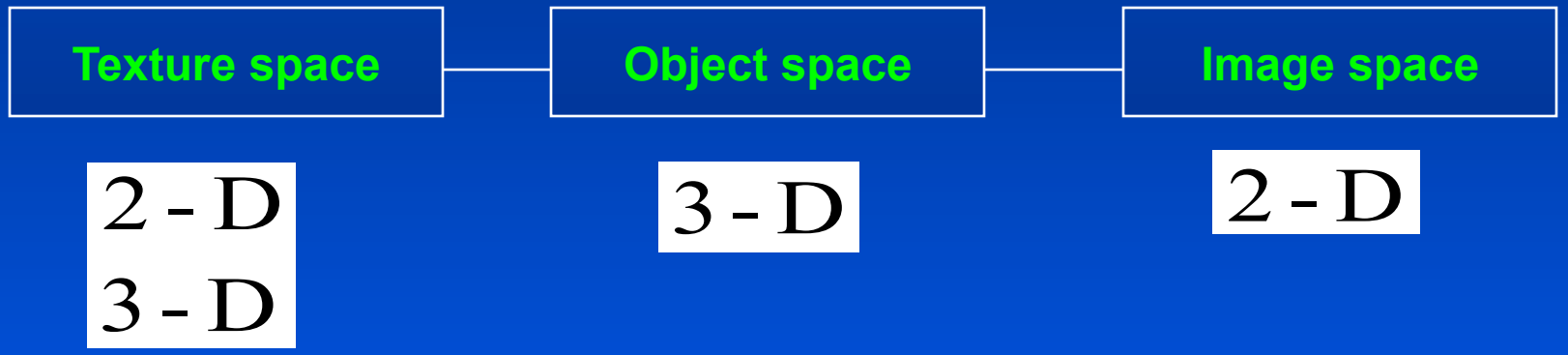
Texture Mapping



Parameters to be modulated

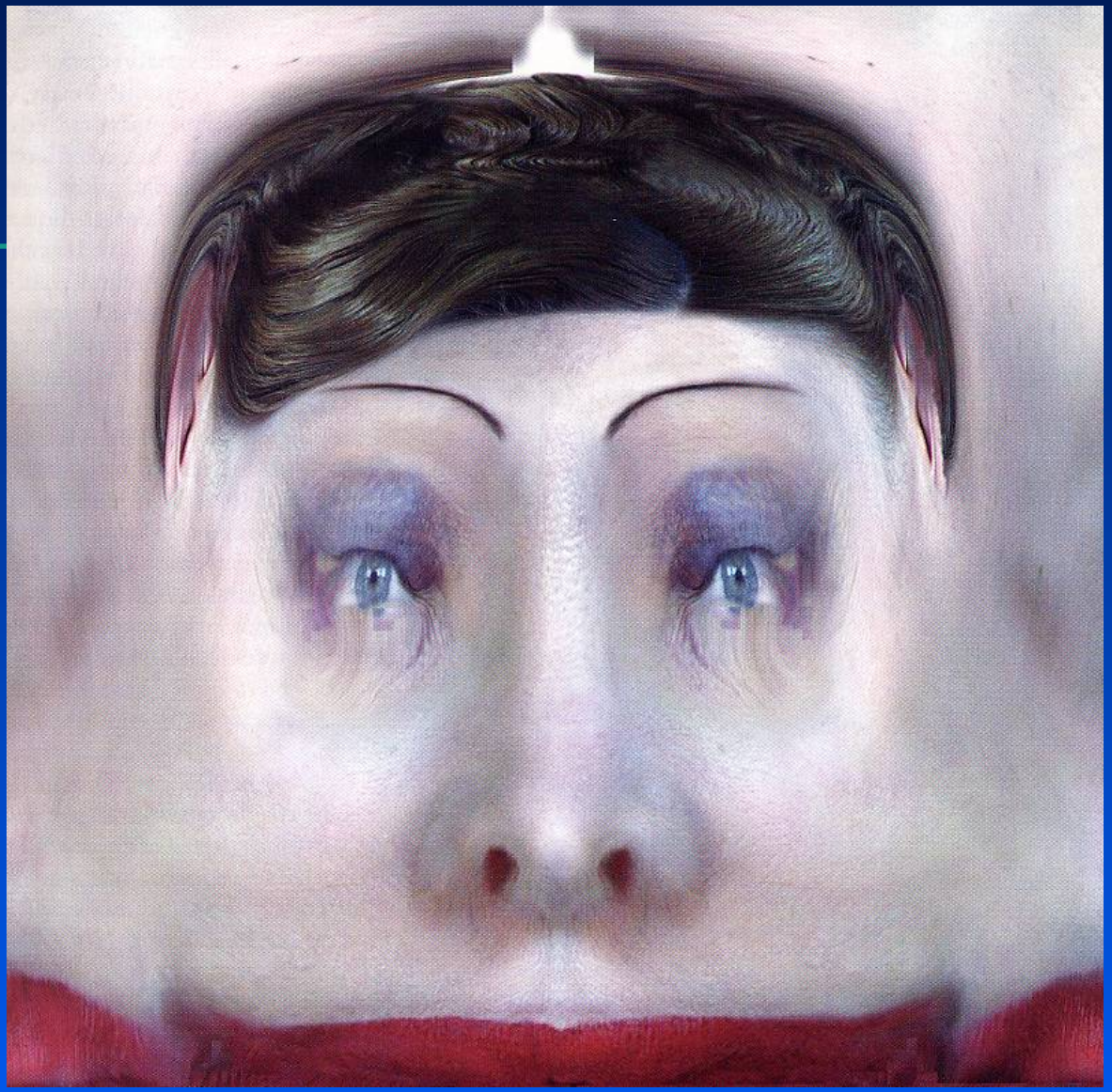
- Surface color
- Specular and diffuse reflection
  - E.g. Environment mapping
- Bump mapping
  - Perturb surface normal
- Specularity – surface roughness
- Transparency
- Displacement

Mapping



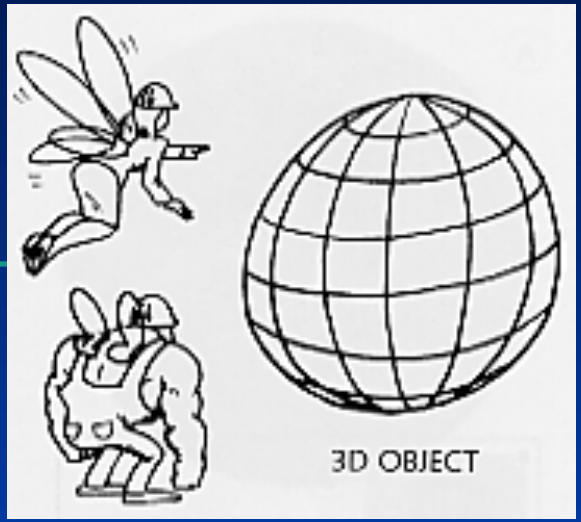
Texture map

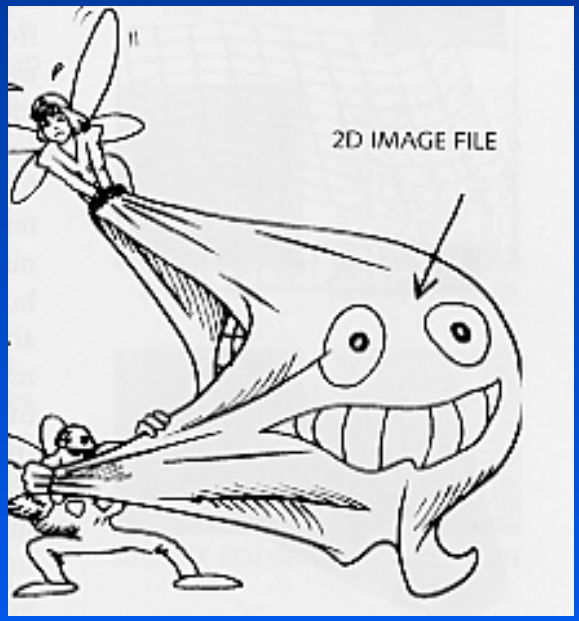
Texture map



2-D texture mapping onto polygons

- Texture space $T(u,v)$, $u, v \in [0,1]$
- Define mapping (projection function) from object space to texture space
  - $(u, v) = F(x, y, z)$
  - usually for vertices of polygons
- Interpolate to get (u, v) for points interior to the polygon





Two stage mapping

1. Map from 2-D texture space to simple 3-D intermediate surface (and vice versa)

$$T(u,v) \xrightarrow[s]{} T'(x_i, y_i, z_i)$$

2. Map from intermediate surface onto object surface (and vice versa)

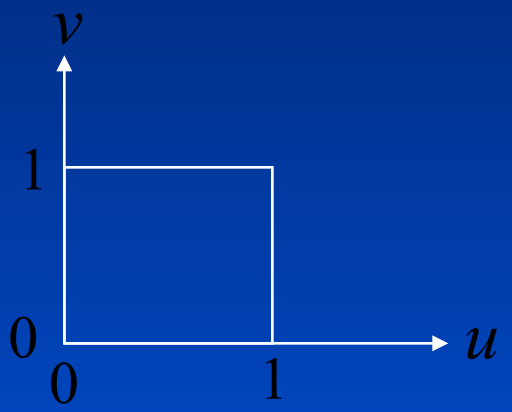
$$T'(x_i, y_i, z_i) \xrightarrow[o]{} O(x, y, z)$$

S-mapping: Cylinder

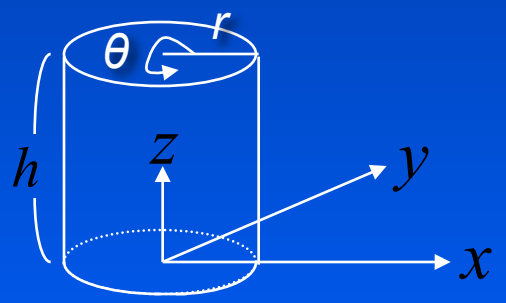
- $(u, v) = (\frac{\theta}{2\pi}, z)$

- A point on a cylinder

$$(r\cos\theta, r\sin\theta, hz)$$
$$0 < \theta < 2\pi, \quad 0 < z < 1$$



- For a point on a cylinder, find θ and z then plug into first equation to find u and v



S-mapping: Sphere (part of)

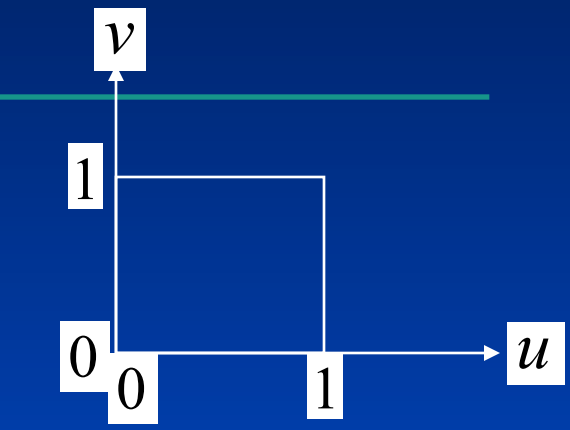
- $$(u,v) = \left(\frac{\theta}{\pi/2}, \frac{(\pi/2)-\phi}{\pi/4}\right)$$

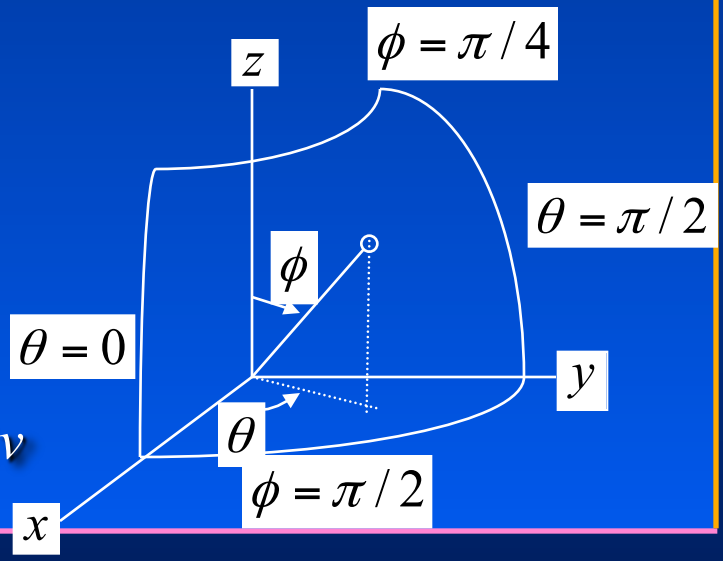
- A point on a sphere:

$$(r\cos\theta\sin\varphi, r\sin\theta\sin\varphi, r\cos\varphi)$$

$$0 \le \theta \le \frac{\pi}{2}, \qquad \frac{\pi}{4} \le \varphi \le \frac{\pi}{2}$$

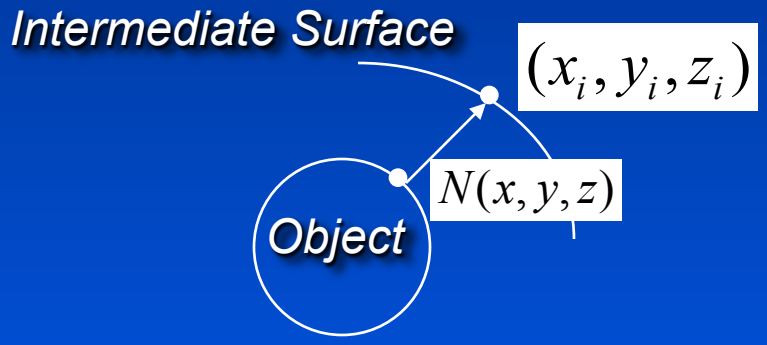
- For a point on a sphere, find θ and Φ then plug into first equation to find u and v



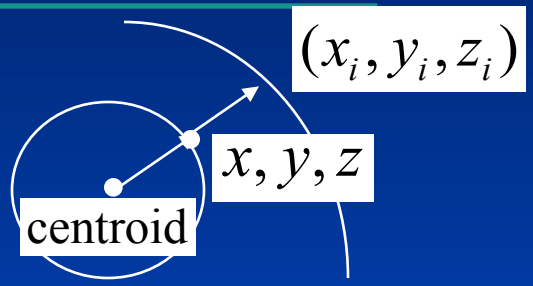


O-mapping (from object to intermediate surface)

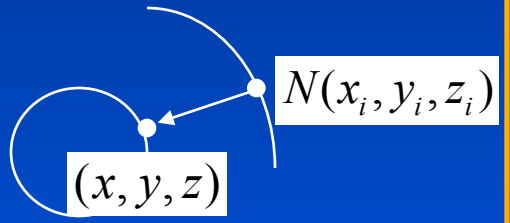
- Intersection of surface normal at *(x,y,z)* w/ intermediate surface
  - Like shrink wrapping texture on object



- Intersection of line through *(x,y,z)* and centroid of object with intermediate surface
  - Same as last one for sphere



- Intersection of line from *(x,y,z)* to intermediate surface
  - Orientation given by surface normal at (x_i,y_i,z_i)
  - Example: if intermediate surface is y-z plane projection mapping
    S-mapping is:

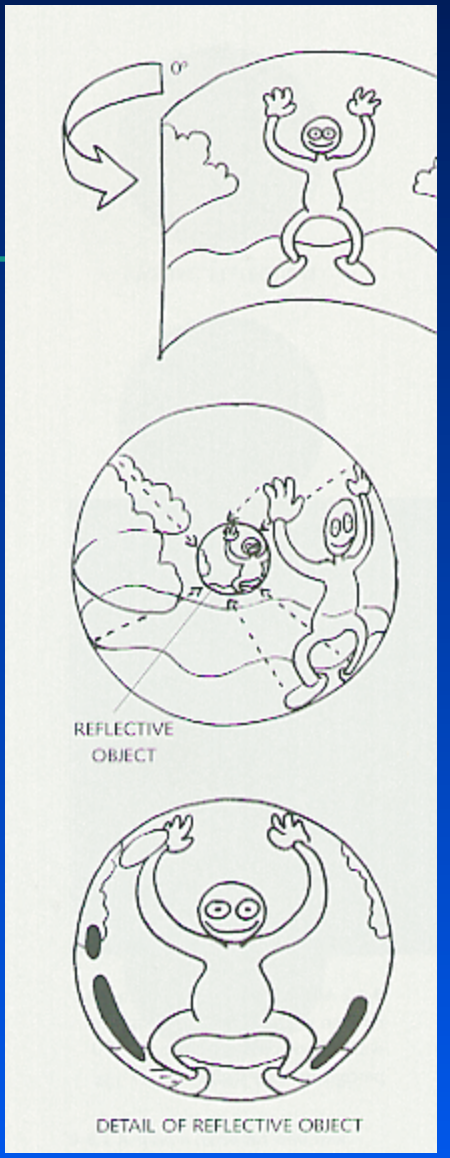


$$(y_l \le y \le y_r, z_l \le z \le z_r)$$

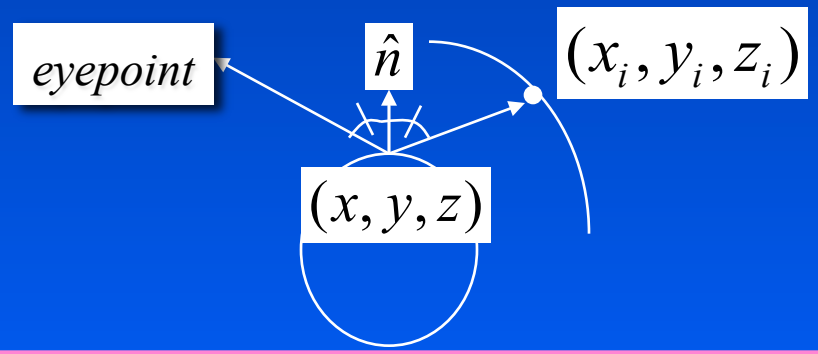
$$(u, v) = \left(\frac{y - y_l}{y_r - y_l}, \frac{z - z_l}{z_r - z_l}\right)$$

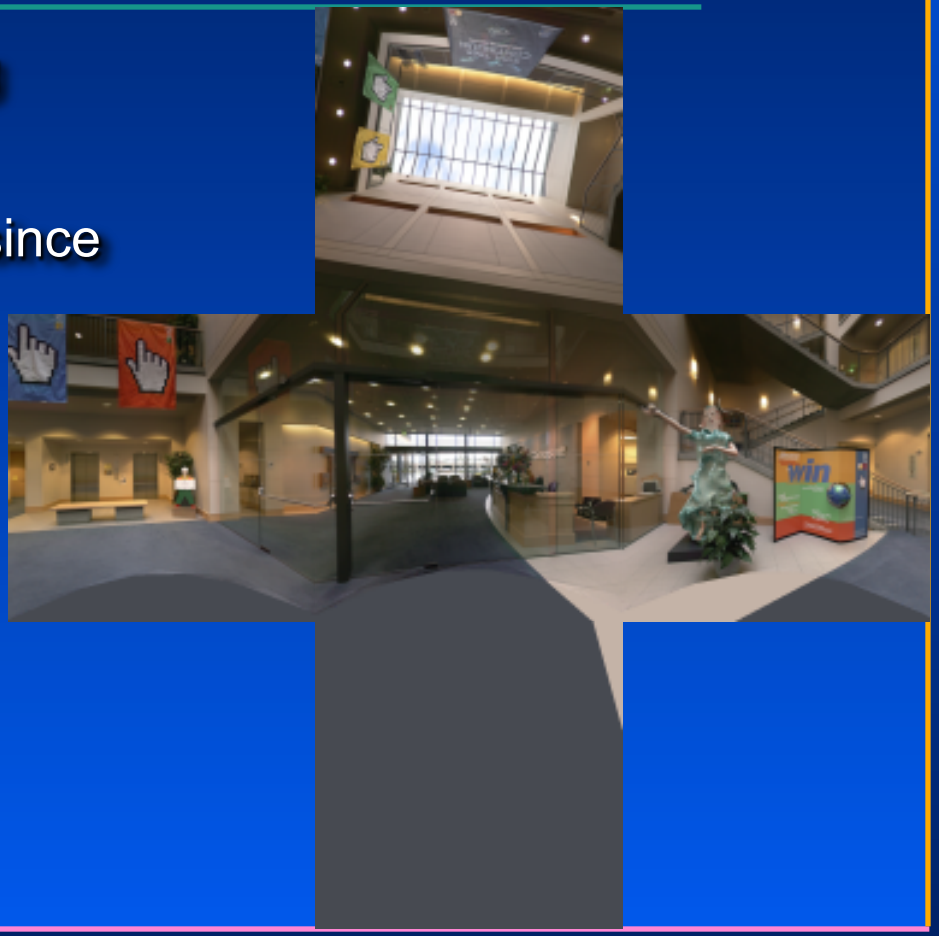
Environment map

- Poor man's ray-tracing
  - Similar effect as ray-tracing
  - Cheaper than ray-tracing
- Create environment map
  - Image of the environment from one viewpoint
- Assume environment map made with center at the point on surface (cannot be true for every point)



- Map onto surface of objects using reflection direction
  - Actually vector direction only since we assume that center of projection used to make the map is at the surface♪





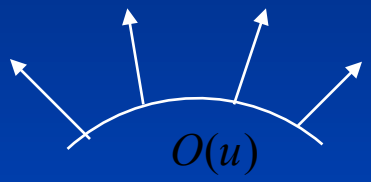
Mapping polygon interior points

- Using mapping from object space to texture space, assign u,v at each vertex of polygon
  - Object space is usually local space for each object
- Interpolate u,v at vertices of polygons across polygon face
- Easiest way: tri-linear interpolation at scan conversion (just like phong or gouraud shading)
  - Usual problems w/ linearly interpolation in image space
  - Rotational variance

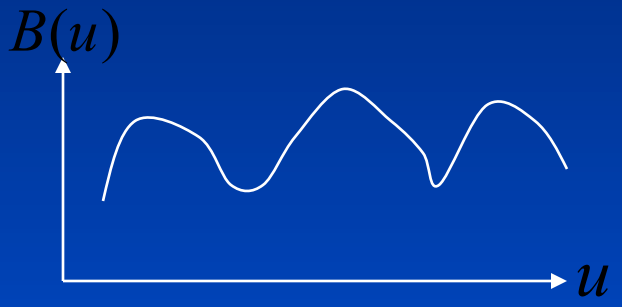
2-D Texture mapping onto bicubic parametric patches

- Trivial mapping, patch parameterized in 2-D over surface
- $s \rightarrow u,\ t \rightarrow v$

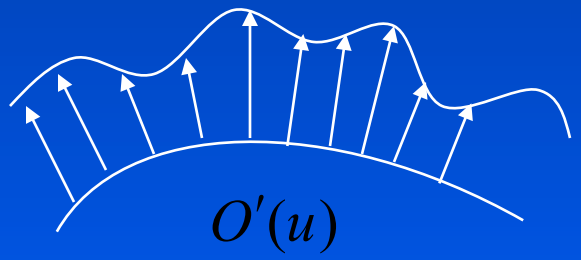
Bump Mapping



original surface



bump map



$$O'(u,v) = O(u,v) + B(u,v)\frac{\overline{N}}{|\overline{N}|}$$

- Differentiate w.r.t u and v (subscript indicate partial derivative)
- If B small last terms approx. 0
- Cross product to get normal:

$$O'_u = O_u + B_u \frac{N}{|N|} + B(\frac{N}{|N|})_u$$

$$O'_v = O_v + B_v \frac{N}{|N|} + B(\frac{N}{|N|})_v$$

$$N' = O'_u \times O'_v$$

$$= O_u \times O_v + B_u(\frac{N}{|N|} \times O_v) + B_v(O_u \times \frac{N}{|N|}) + 0$$

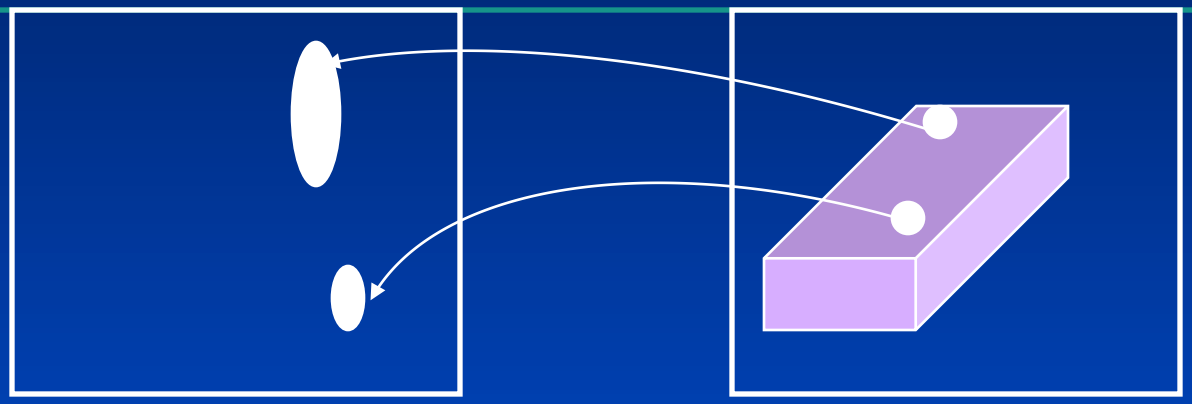
$$= N + B_u(\hat{N} \times O_v) - B_v(\hat{N} \times O_u) \equiv N + D$$

- Perturb surface normal by D
- Problem at silhouette
- Displacement map - perturb surface

Texture Mapping and Anti-Aliasing

- To perform weighted area sampling transform filter base from image to texture space (space variant)
- Two approach : Direct convolution, Pre-filtering

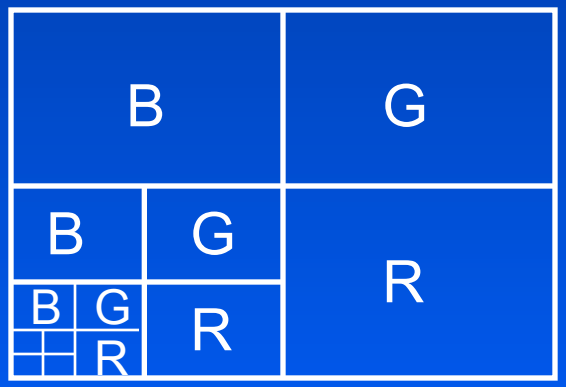
Direct Convolution

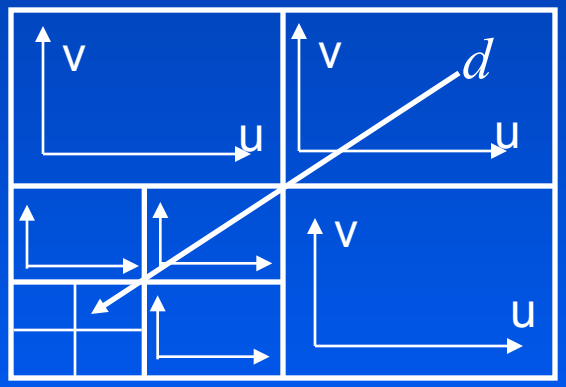


- Transform filter kernel (~pixel) to texture space (“Space variant filter”)
- Bring appropriate texels to screen space
- Perform weighted sum in image space
- Very expensive operation for each pixel

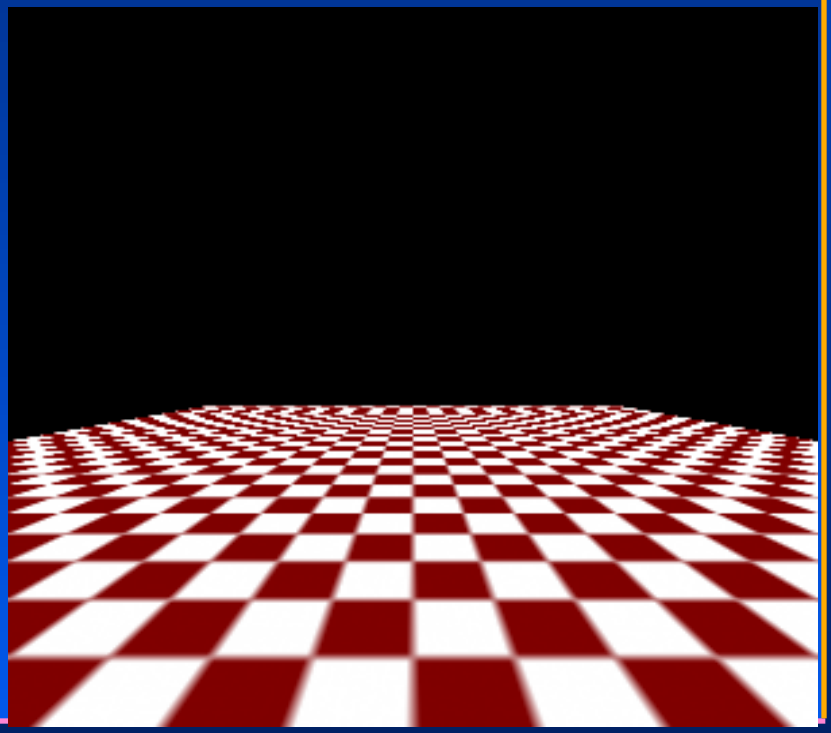
Pre-filtering: Mip-Map

- Ignore shape variation of filter in texture space- assume square box
- Allow area of filter kernel to vary
- “level of detail” of texture form a pyramid
- Each level ½ resolution of previews level





- When zoom in, texel size large compared to pixel size
- When zoom out, texel size small (must average many texels to get a pixel value)
  - If we do not average, aliasing



- Pre average the texels by combining 4 neighboring texels into one texel recursively
- “level of detail” of texture form a pyramid
- Each level ½ resolution of previews level

- Displayed at same resolution

- d-level of the pyramid
- Measure of how much texel area “compressed” under a pixel $\propto$ area of kernel in texture space
- Bilinear interpolation within 2 adjacent levels to get 2 texel values for a particular u,v
- Linear interpolation between levels to get a texel value for a particular d
- If d too large blurring, if d too small aliasing♪

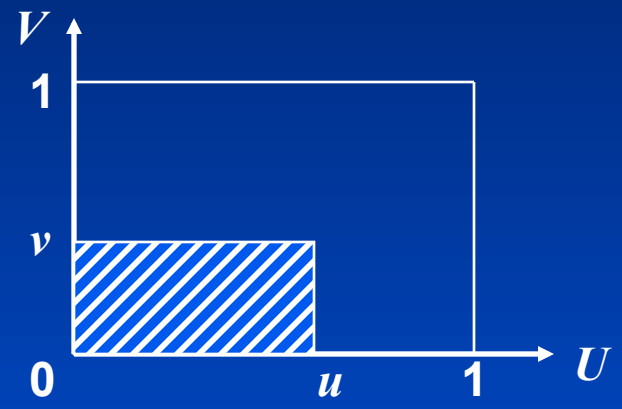
Pre-filtering: Summed Area table

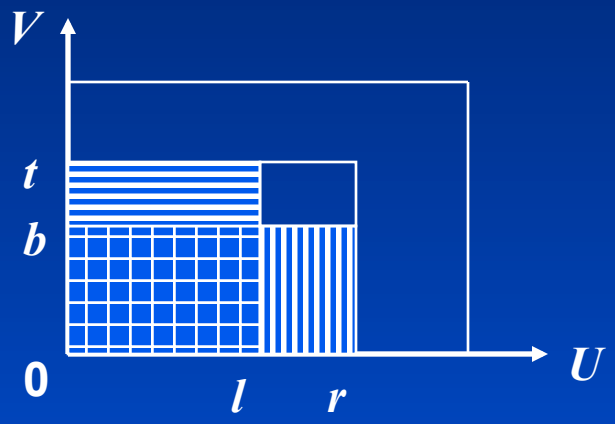
- Any rectangular region in texture map (constant weight)

$$T'(u,v) = \int_0^u \int_0^v T(u,v)\,dv\,du$$

Where T' is summed area table

T is original texture map



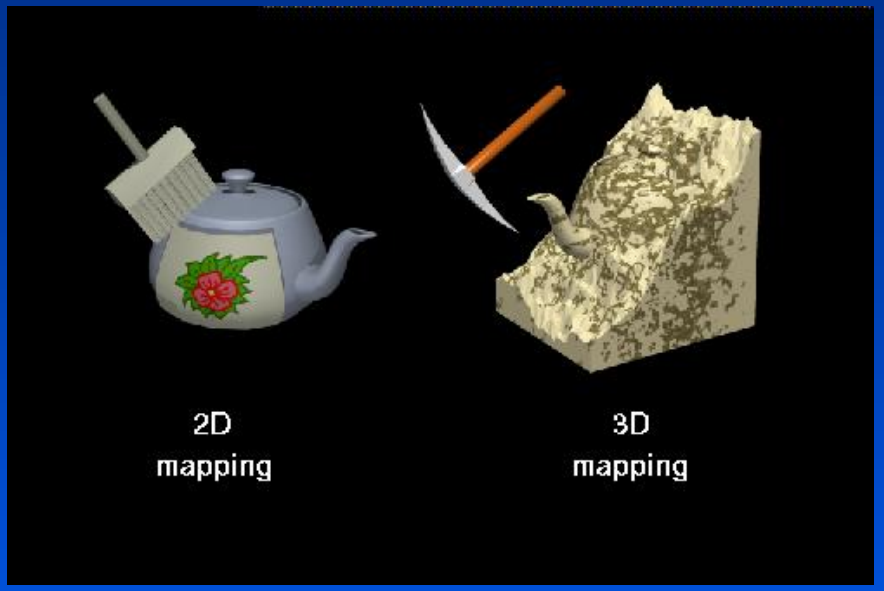


- Integral in arbitrary rectangle :♪

$$T'(u_r, v_t) - T'(u_l, v_t) - T'(u_r, v_b) + T'(u_l, v_b)$$

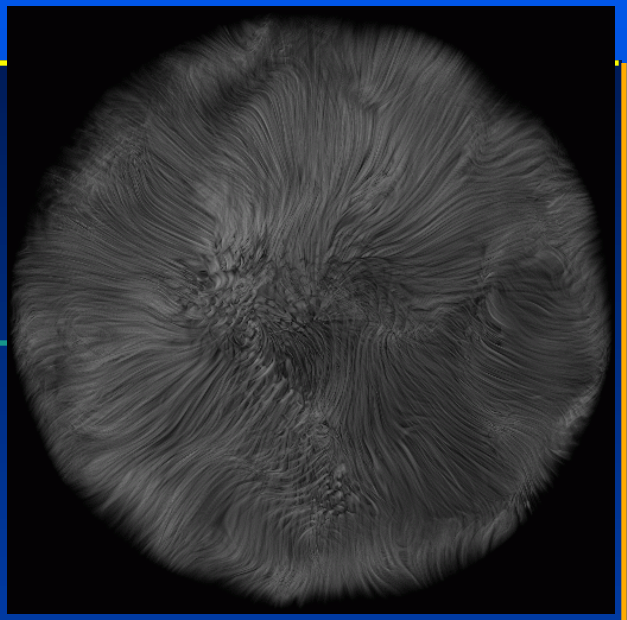
3-D Texture

- $x,y,z \rightarrow u,v,w$
- Two spaces coincident
  - Object immersed in texture space
- Texture space needs to be tied to local space of object (not world space) as before
- Maybe scale
- Procedurally generated or scanned (e.g. CT)



Hyper-Texture

- No underlying object shape
- Texture itself determine object shape
- Render: e.g. using volumetric rendering techniques



Procedural texture mapping

- Texture generated algorithmically
- Anti-aliasing by making sure no high-frequency components are present when generating them
  - Usually done by additive synthesis so just not add components that can cause aliasing
- We will discuss this topic in detail later

Next:
Procedural textures and models