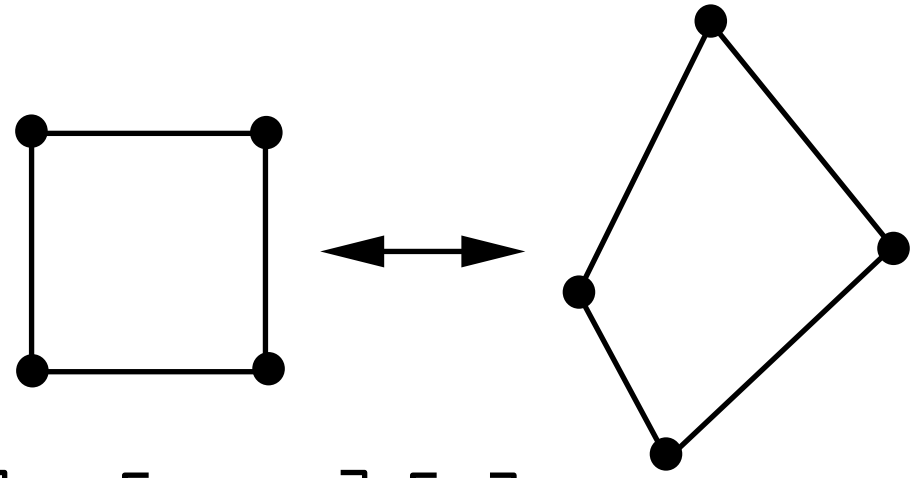
Mapping to Polygon Mesh Objects

Mapping a square to a quadrilateral



$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} s' \\ t' \\ q \end{bmatrix}$$

$(x, y) = (x'/w, y'/w)$

$(s, t) = (s'/q, t'/q)$

Affine (linear) mapping:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s \\ t \\ 1 \end{bmatrix}$$

$x = as + bt + c$
$y = ds + et + f$

Forward mapping:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s \\ t \\ 1 \end{bmatrix}$$

Inverse mapping:

$$\begin{bmatrix} s \\ t \\ 1 \end{bmatrix} = \begin{bmatrix} A & B & C \\ D & E & F \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$= \frac{\text{cofactor}(T_f)}{\det(T_f)} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$= \frac{1}{ae-bd} \begin{bmatrix} e & -b & bf-ec \\ -d & a & dc-af \\ 0 & 0 & ae-bd \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Perspective Mapping:

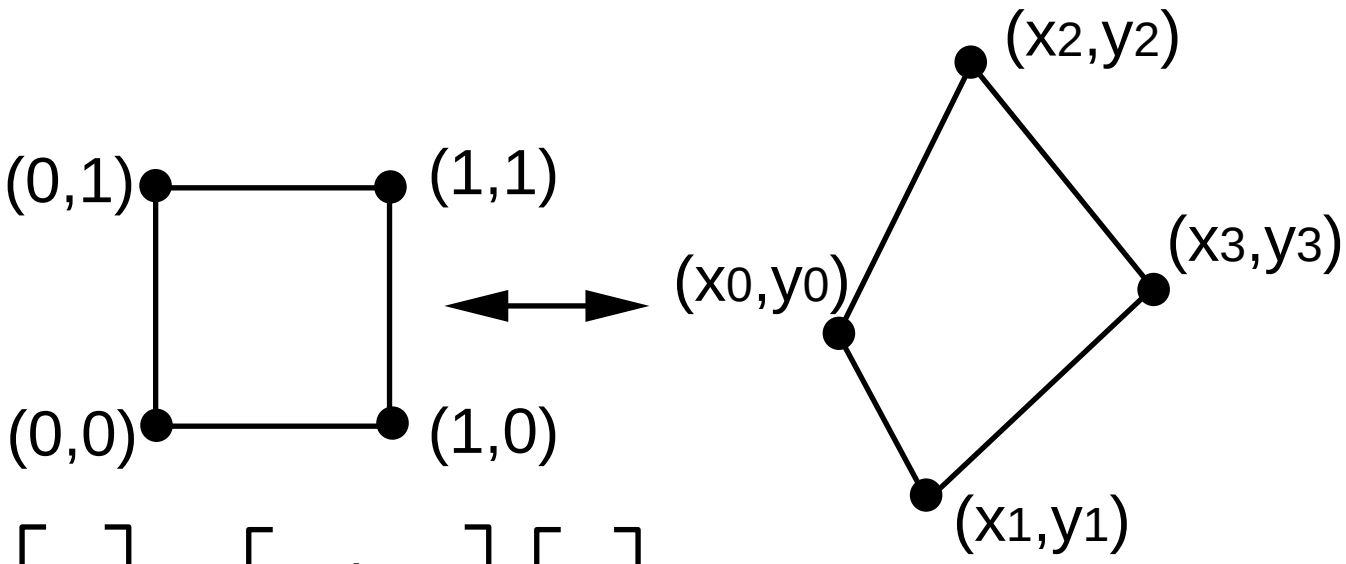
$$\begin{bmatrix} s' \\ t' \\ q \end{bmatrix} = \begin{bmatrix} A & B & C \\ D & E & F \\ G & H & I \end{bmatrix} \begin{bmatrix} x' \\ y' \\ w \end{bmatrix}$$

$$= \frac{\text{cofactor}(T_f)}{\det(T_f)} \begin{bmatrix} x' \\ y' \\ w \end{bmatrix}$$

$$= \frac{1}{ae-bd} \begin{bmatrix} ei-fh & ch-bi & bf-ec \\ fg-di & ai-cg & dc-af \\ dg-eg & bg-ah & ae-bd \end{bmatrix} \begin{bmatrix} x' \\ y' \\ w \end{bmatrix}$$

$s = (Ax + By + C) / (Gx + Hy + I)$

$t = (Dx + Ey + F) / (Gx + Hy + I)$



$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} s' \\ t' \\ q \end{bmatrix}$$

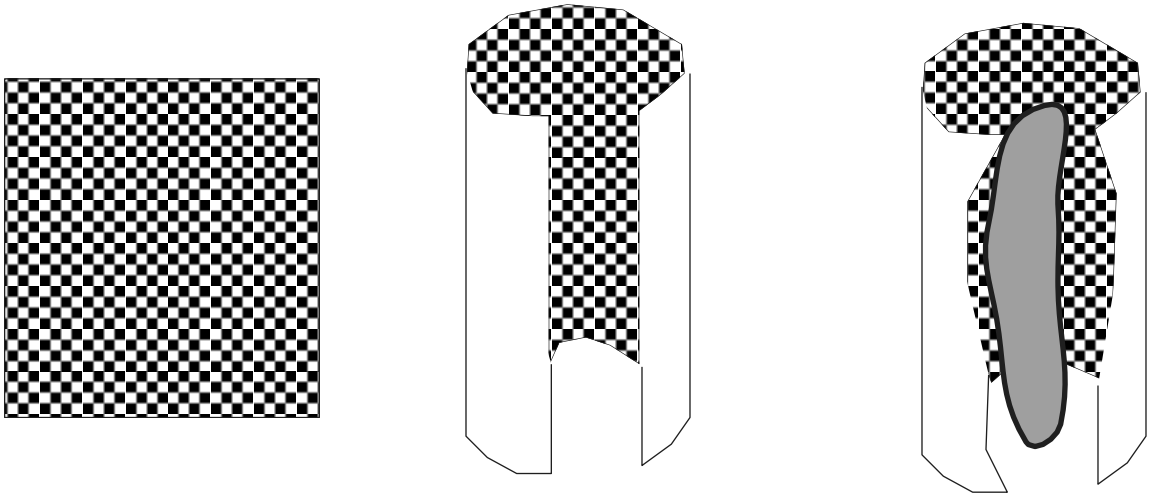
$x = x'/w = (as + bt + c)/(gs + ht + i)$
$y = y'/w = (ds + et + f)/(gs + ht + i)$

assume $i = 1$

$x = as + bt + c - gsx - htx$
$y = ds + et + f - gsy - hty$

$x_0 = c$
$y_0 = f$
$x_1 = a + c - gx_1$
$y_1 = d + f - gy_1$
$x_2 = b + c - hx_2$
$y_2 = e + f - hy_2$
$x_3 = a + b + c - gx_3 - htx_3$
$y_3 = d + e + f - gy_3 - hy_3$

Inverse Mapping using an Intermediate Surface



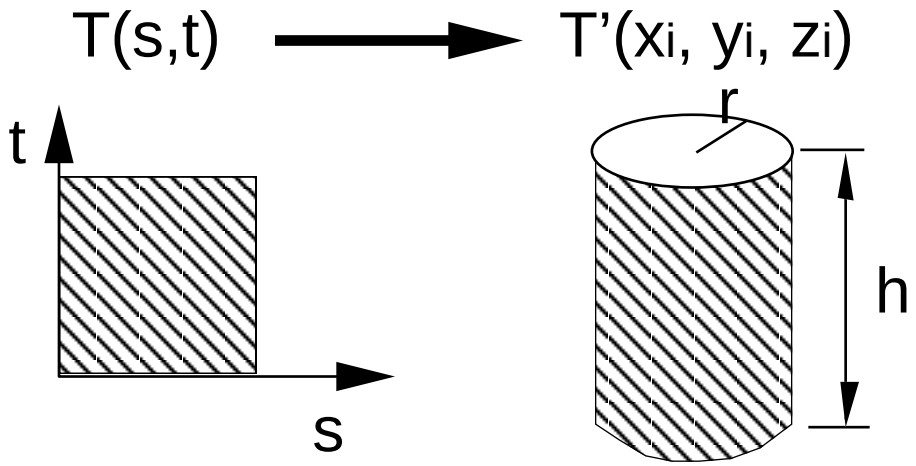
- used when there is no texture coordinate−vertex coordinate correspondence
- two pass method
- the intermediate surface is generally non−planar but it possesses an analytic mapping function

e.g.

Using a cylinder, cube or sphere results in a 3d to 3d mapping

Procedure:

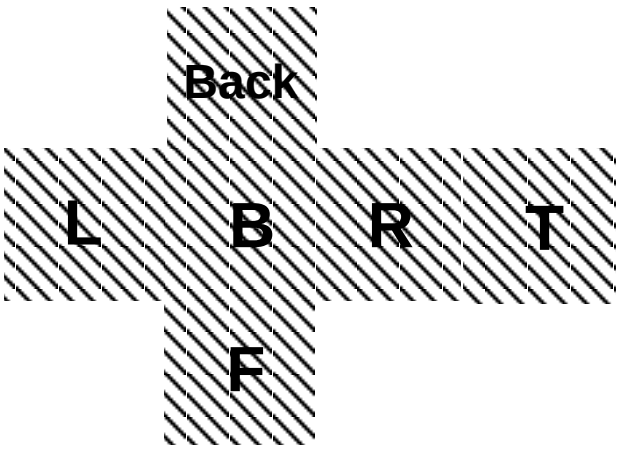
1. mapping from 2d texture space to a simple 3d intermediate surface

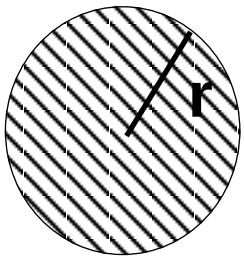


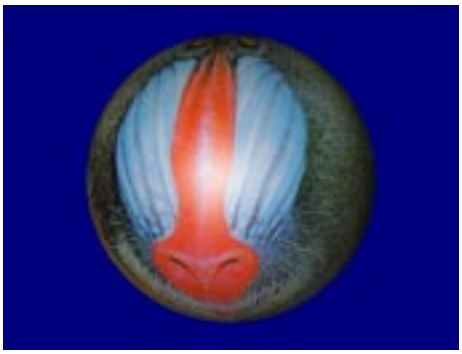
points on the cylinder are given by parametric equations:

$$xi = r \cos(2\pi\alpha) \qquad 0 \le \alpha, \beta \le 1$$
$$yi = r \sin(2\pi\alpha)$$
$$zi = \beta/h$$

no distortion without using top and bottom







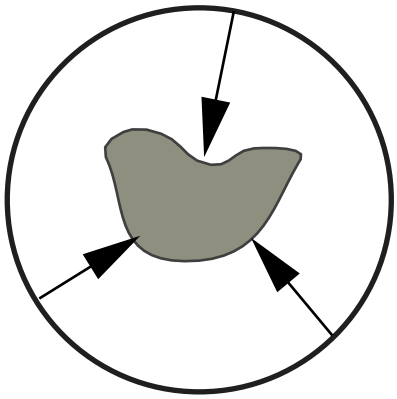
$$xi = r \cos(2\pi\alpha)$$
$$yi = r \sin(2\pi\alpha) \cos(2\pi\beta)$$
$$zi = r \sin(2\pi\alpha) \sin(2\pi\beta)$$

introduce distortion!

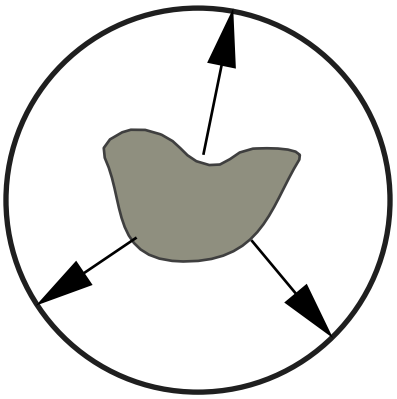
Procedure (cont'd)

2. mapping the 3d texture pattern onto the object surface

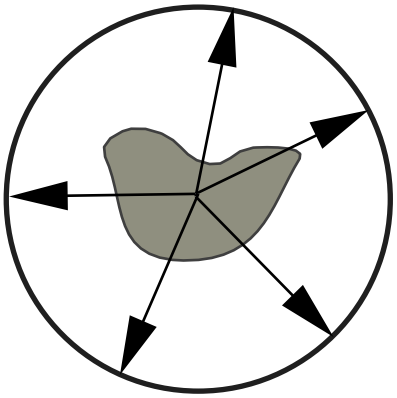
$T'(x_i, y_i, z_i) \longrightarrow O(x_w, y_w, z_w)$



using the normal of the intermediate surface



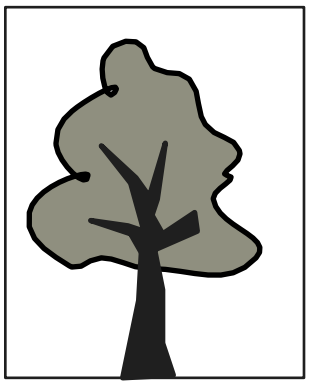
using the normal from the object surface

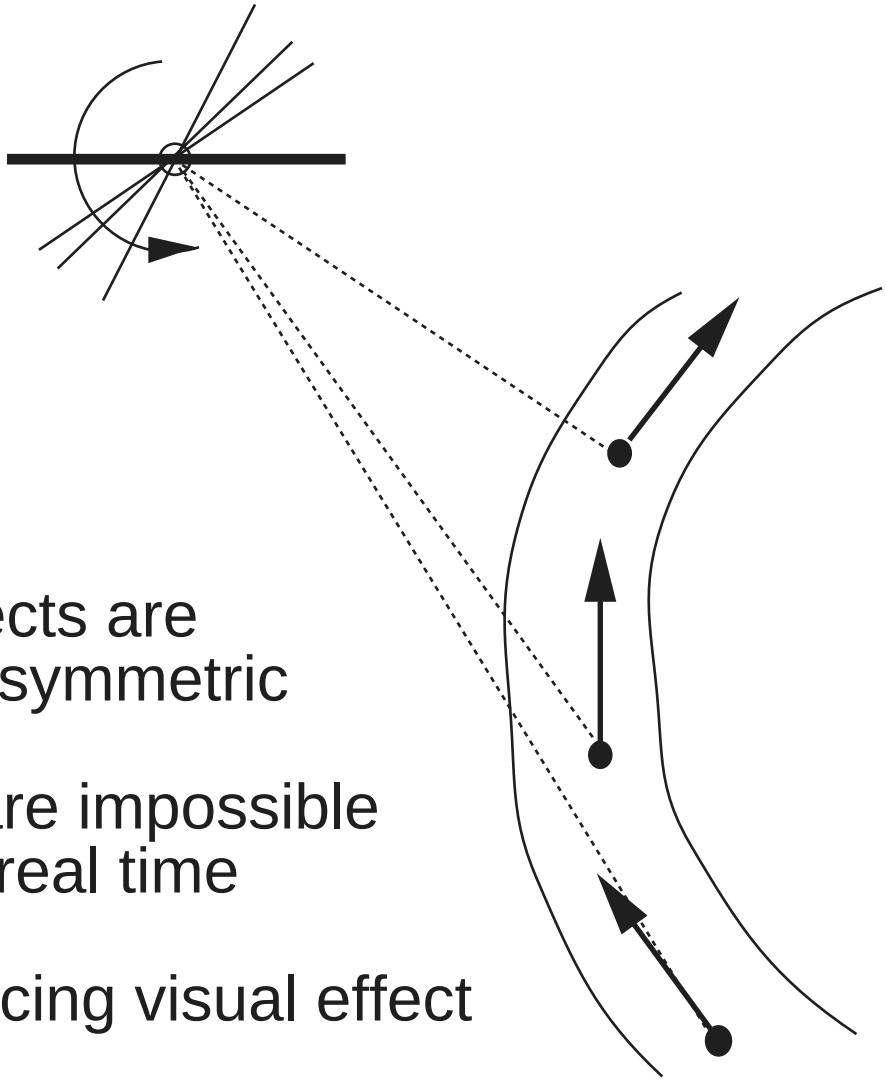


using the center of the object

Billboards

- Use 2d image as a 3d entity
- Rotate the plane of the image so it is always normal to its viewing direction





- best for objects are cylindrically symmetric
- for objects are impossible to render in real time
- Offer convincing visual effect
- View point only changes slightly
- View vector is parallel to the horizontal plane in the scene space

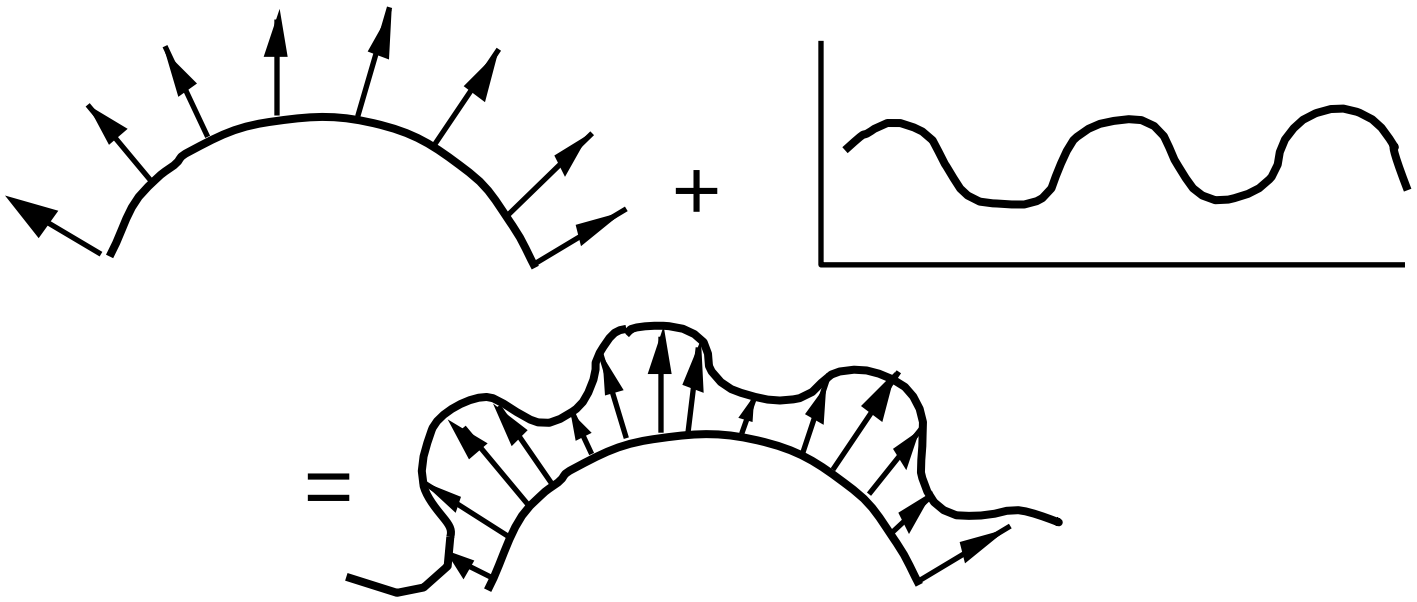
Bump Mapping

- makes a surface appear as if it were wrinkled or dimpled without changing the geometry of the surface

- angularly perturb surface normals according to a 2−d bump map

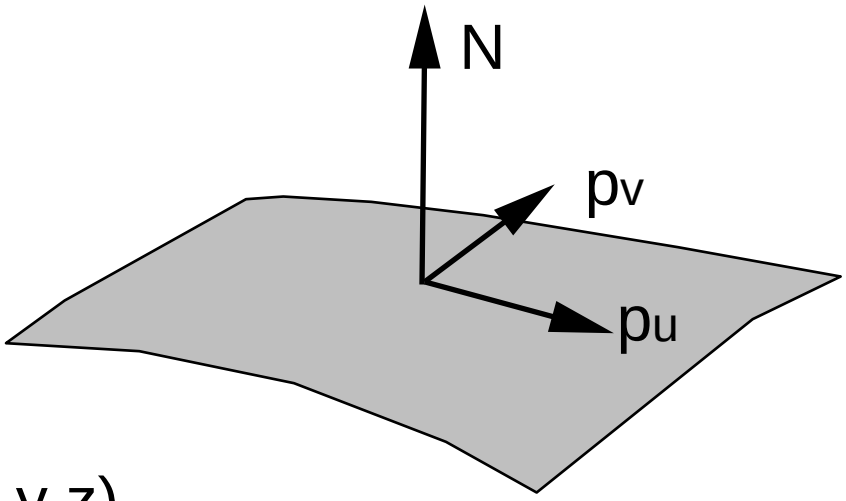
- coupled with a local illumination model, a visual illusion of surface bumps results.

$$I = I_a k_a + I_d k_d (N \bullet L) + I_s k_s (R \bullet V)^n$$



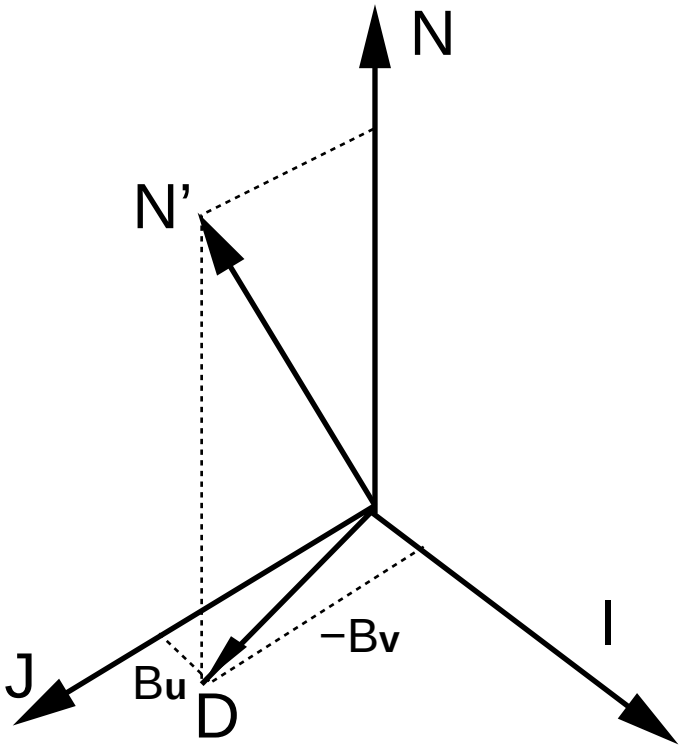
- surface detail should be small relatively to the overall size of the object

- silhouette edges follow the real geometry!!!



P(x,y,z)

$$N = \frac{\delta P}{\delta u} \times \frac{\delta P}{\delta v}$$



$$P'(u,v) = P(u,v) + B(u,v)N$$

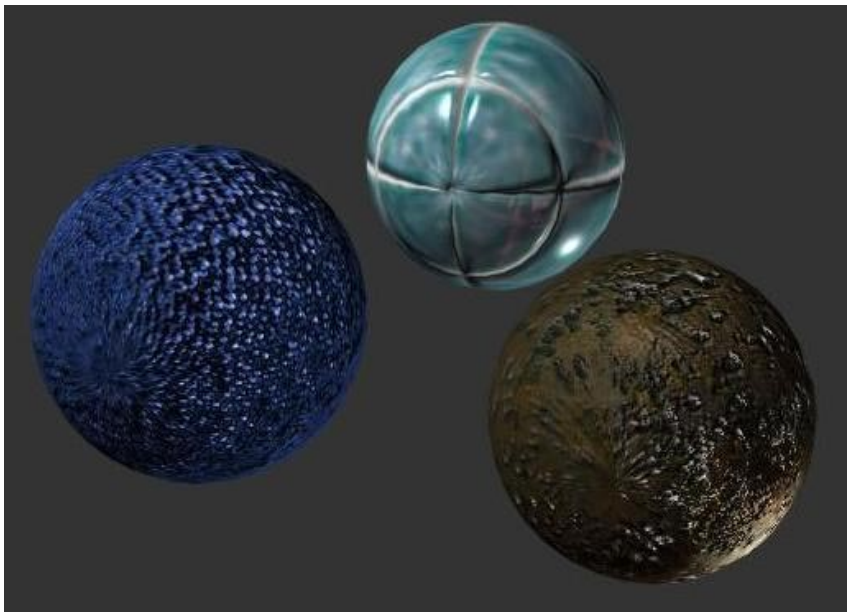
$$N' = P'_u \times P'_v$$

$$P'_u = P_u + B_u N + B(u,v)N_u$$
$$P'_v = P_v + B_v N + B(u,v)N_v$$

$$N' = N + B_u N \times P_v + B_v P_u \times N$$

$$= N + (B_u I - B_v J)$$

$$= N + D$$



Environment Mapping

- a short cut to rendering shinny objects that reflect the environment in which they are placed
- can approximate the quality of ray−tracing for specular reflections
- the environment is defined by a 2d map which is obtained with a preprocessing step

- geometrically correct only when the object becomes smal w.r.t. the env that contains it
- doesn’t work for concave objects
- a separate map is needed for each object in the scene
- in some cases, it’s view dependent

A two−step procedure:

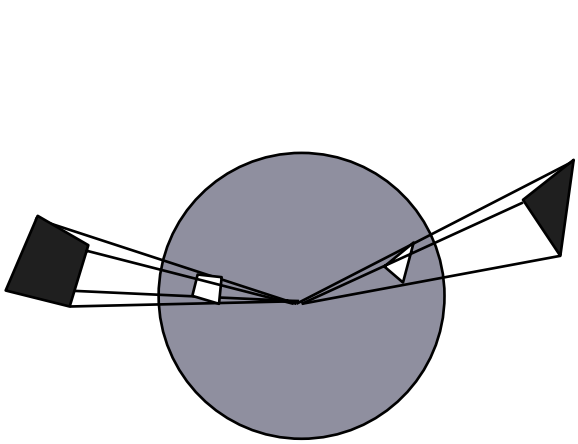
1) compute an image of the environment on an intermediate projection surface

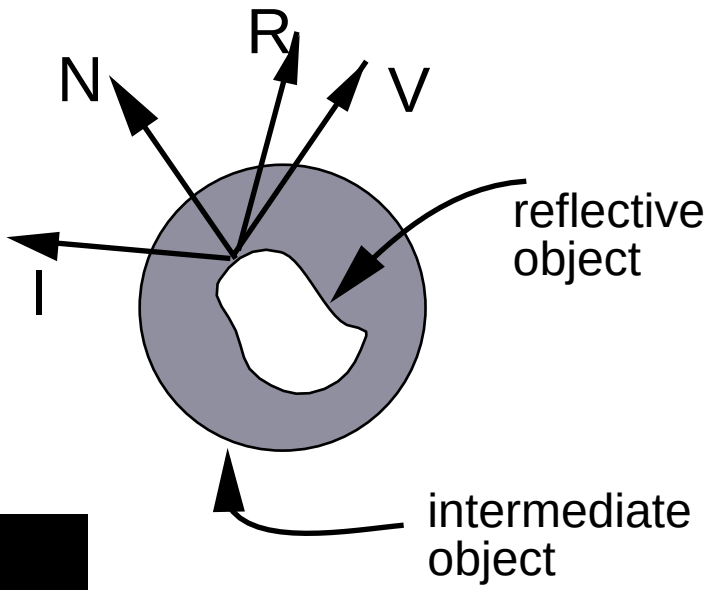
   − cop is the center of the reflective object

   − projections are computed with the object removed

2) map the resulting "texuture" to the object

   − use viewer location and surface normals

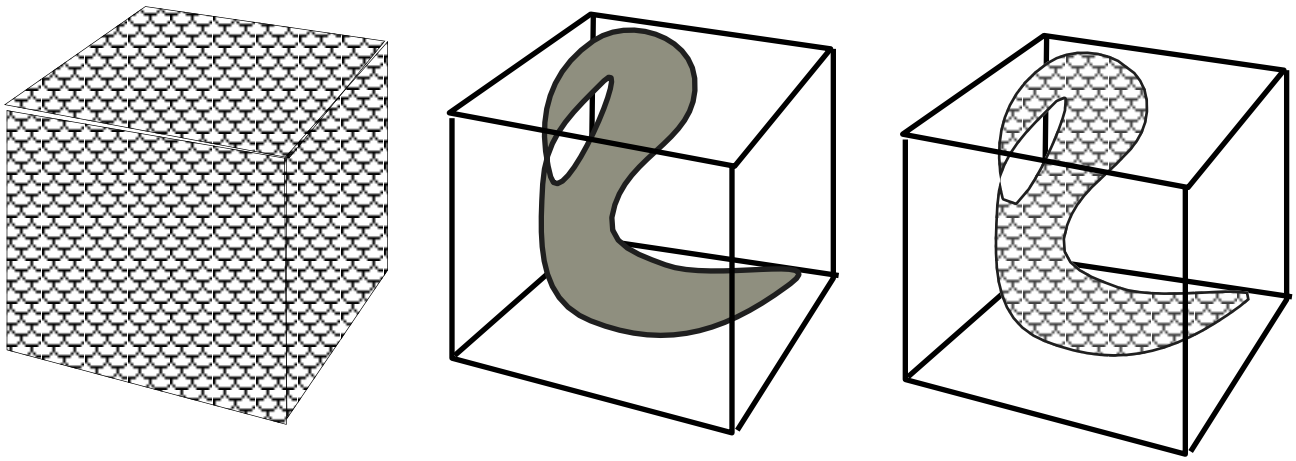




3D Textures

- map a 2D texture to the surface of a 3d object can be a non−trivial and problematic task.

- with 3d texture mapping, only need to determine the position of the point in space



P(x, y, z) on the surface of an object has color defined by T(x, y, z)

- T is procedurally generated

  −> compact, no fixed resolution, parameterized

- mapping problem is eliminated but the type of textures can be created is limited

3D Noise Textures

- noise as a texturing primitive
- noise functions should be controllable
- noise functions should have a narrow bandpass limit in frequency