Surface Details

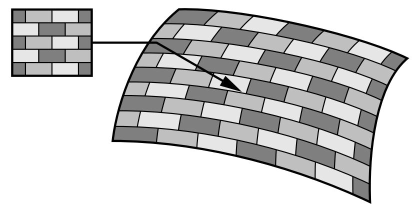
- ☆ Incorporate fine details in the scene.
- ☆ Modeling with polygons is impractical.
- ☆ Map an image (texture/pattern) on the surface (Catmull, 1974); (Blin & Newell, 1976).

- ☆ *Texture map*
  - Models patterns, rough surfaces, 3D effects.
- ☆ *Solid textures*
  - (3D textures) to model wood grain, stains, marble, etc.
- ☆ *Bump mapping*
  - Displace normals to create shading effects.
- ☆ *Environment mapping*
  - Reflections of environment on shiny surfaces.
- ☆ *Displacement mapping*
  - Perturb the position of some pixels.

Slide 1

Texture Maps

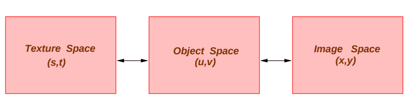
- ☆ Maps an image on a surface.
- ☆ Each element is called *texel*.
- ☆ Textures are fixed patterns, procedurally generated, or digitized images.



Slide 2

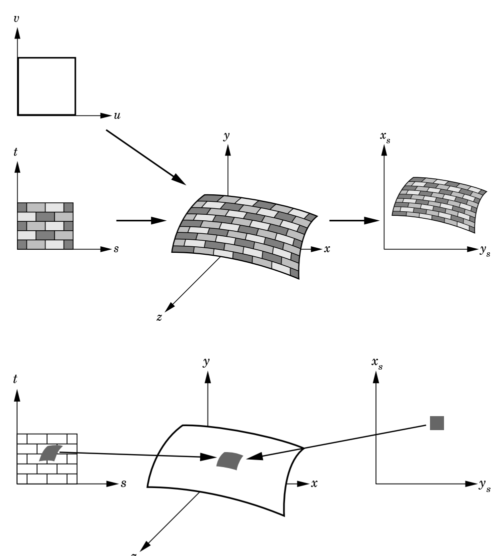
Texture Maps

- ☆ Texture map has its own coordinate system; st-coordinate system.
- ☆ Surface has its own coordinate system; uv-coordinates.
- ☆ Pixels are referenced in the window coordinate system (Cartesian coordinates).



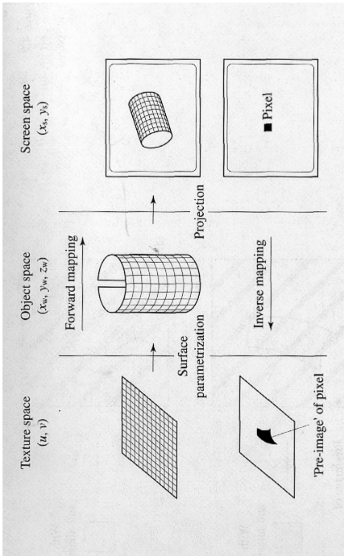
Slide 3

Texture Maps



Slide 4

Texture Mapping



Slide 5

Texture Mapping

Forward mapping: (Texture scanning)

☆ Map texture pattern to the object space.

$$\begin{aligned} u &= f_u(s,t) = a_u s + b_u t + c, \\ v &= f_v(s,t) = a_v s + b_v t + c. \end{aligned}$$

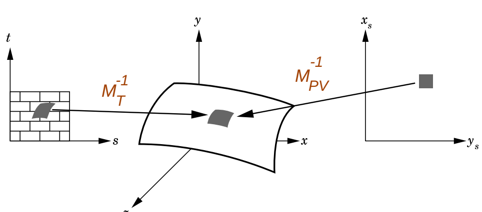
☆ Map object space to window coordinate system.
Use modelview/projection transformations.

Drawback: Selected texture patch usual does not match with pixel boundaries.

☆ Requires fractional pixel calculations.

Slide 6

Inverse Mapping



☆ Map screen coordinate system to object space.

☆ Map object coordinate system to texture space.

☆ Avoids fractional pixel calculations.

☆ Allows anti-aliasing.

☆ Requires calculating inverse transformations; M_{PV}^{-1}, M_T^{-1}.

- M_{PV}^{-1} can be computed from projection and modelview matrices (`gluUnproject`)
- Computing M_T^{-1} is not easy.

Slide 7

Parametric Representation

Curves: Coordinates are represented as functions of one parameter.

$$\gamma(t) = (x(t), y(t)), \quad t \in [a,b]$$

Line: $\ell : (a_1 + b_1 t, a_2 + b_2 t),\ t \in \mathbb{R}$

Surfaces: Coordinates of each point is represented as a function of u and v.

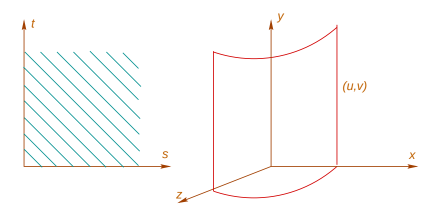
$$S(u,v) = (x(u,v), y(u,v), z(u,v)).$$

☆ *Sphere:*
$S(u,v) = (r\cos u \cos v, r\cos u \sin v, r\sin u)$
$-\pi/2 \le u \le \pi/2, 0 \le v \le 2\pi.$

☆ *Cylinder:*
$S(u,v) = (r\cos u, r\sin u, v)$
$0 \le u \le 2\pi, 0 \le v \le h.$

Slide 8

Inverse Mapping: An Example



$$u = \theta, v = y \qquad 0 \le \theta \le \pi/2, 0 \le y \le 1$$

$x = \sin\theta, z = \cos\theta, y = v.$
$M_{PV}^{-1} : u = \sin^{-1} x, \quad v = y.$
Map texture origin to left bottom corner of the surface

$$u = s\pi/2, \quad v = t.$$

Projected pixels are mapped to texture with M_T^{-1}:

$$M_T^{-1} : s = \frac{2u}{\pi} \qquad t = v.$$

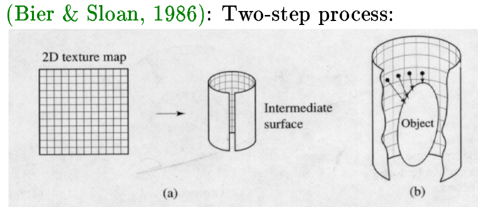
$$s = \frac{2}{\pi}\sin^{-1} x \qquad t = z.$$

Slide 9

Object to Texture Mapping

How does one define a reasonable M_T^{-1}?

(Bier & Sloan, 1986): Two-step process:



***S*-mapping**: Mapping from a 2D texture space to a simple 3D surface, e.g., cylinder.

$$T(u, v) \to T'(x_i, y_i, z_i).$$

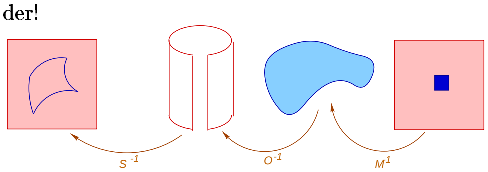
***O*-mapping**: Mapping from the 3D texture pattern onto the object surface.

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

Slide 10

Two-Step Inverse Mapping

Inverse mapping: Apply them in the reverse order!



Slide 11

S-Mapping

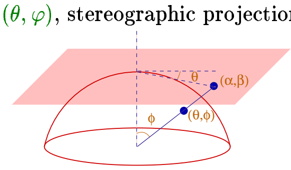
Four possible surfaces: Plane, cylinder, sphere, box.

Cylinder: (θ, z)

$$S : (\theta, z) \to \left[\tfrac{r}{c}(\theta - \theta_0), \tfrac{1}{d}(z - z_0)\right]$$

- ☆ c, d: Scaling factors.
- ☆ Texture origin is mapped to the point (θ_0, z_0) on the cylinder.

Sphere: (θ, φ), stereographic projection.

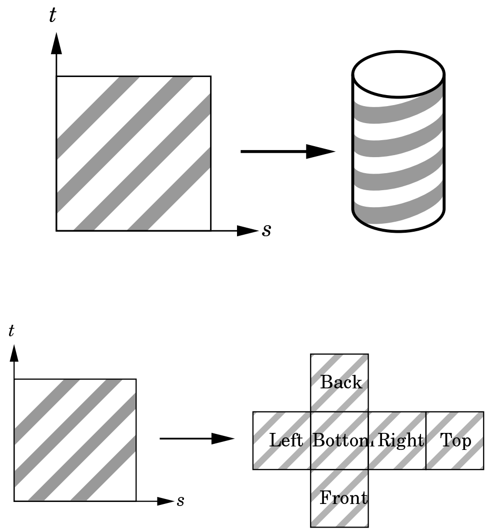


$$S : (\theta, \varphi) \quad \to \quad \left[\frac{2\alpha}{1 + (1 + \alpha^2 + \beta^2)^{1/2}}, \frac{2\beta}{1 + (1 + \alpha^2 + \beta^2)^{1/2}}\right]$$

$$\alpha = \tan\varphi\cos\theta,\ \beta = \tan\varphi\sin\theta.$$

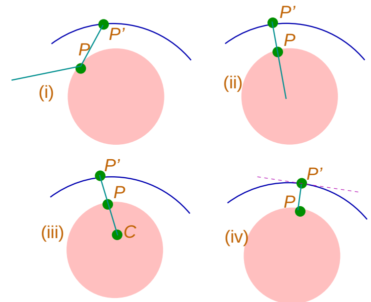
Slide 12

S-Mapping



Slide 13

O-Mapping



☆ Γ: Intermediate surface

☆ P': Intersection of Γ with a ray ρ emanating from P.

☆ Direction of ρ:

(i) Reflection direction

(ii) Normal of O at P

(iii) $\overline{CP}$; C: centroid of O

(iv) $-\rho$: Normal of Γ at P'

☆ (ii), (iii) are bad if Γ is cylinder.

Slide 14

3D Textures

Introduced by Peachey and Perlin in 1985.

☆ Define texture to be a 3D image.

- Carving out an object from a 3D solid material.

☆ Ignoring scaling, M_T is identity.

☆ Distortion is minimized.

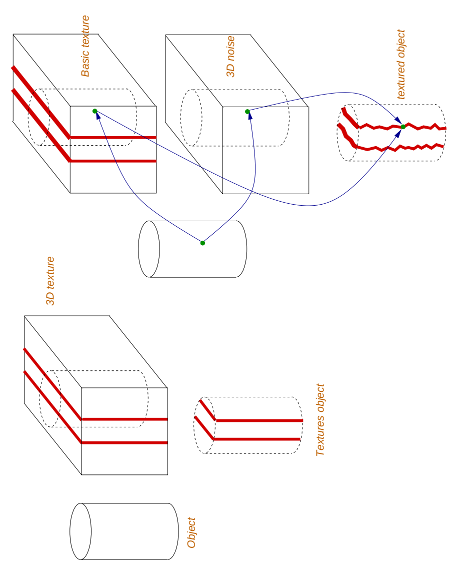
☆ Three dimensional vector fields can be mapped coherently.

☆ Texture is generated by a procedure.

☆ *Example:* Wood grain can be mapped as a set of cylinders with respect to a prespecified axis.

Slide 15

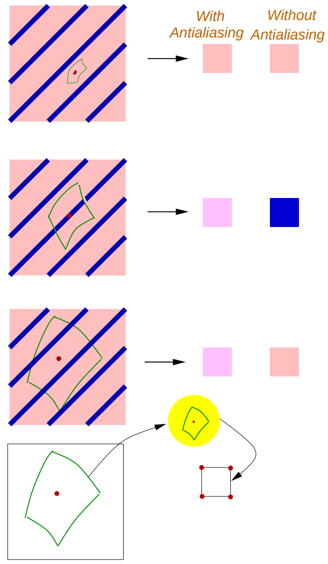
3D Textures



Slide 16

Anti-aliasing & Texture Mapping

Aliasing is particularly visible in periodic and coherent textures.



Slide 17

Anti-aliasing & Texture Mapping

- ☆ A pixel is mapped to a curvilinear quadrilateral.
- ☆ A single pixel may cover many texels.
- ☆ Compute a weighted sum of texel values covered by the pixel.
- ☆ Summation is called *filtering*.

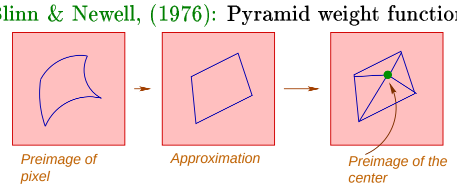
Two-step process:

- ☆ Define and approximate the texture over which filtering is performed.
- ☆ Integrate by weighing and summing the texel values within the filtering area.

Slide 18

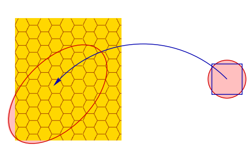
Anti-aliasing: Weight Functions

Blinn & Newell, (1976): Pyramid weight function



Greene & Heckbert, (1986): Elliptical weighted average

- ☆ Approximate a pixel by circle.
- ☆ Circle always maps to an ellipse in the texture space.
- ☆ Find the texels that lie inside the ellipse.
- ☆ Use a look-up table to determine the weighted value of each texel.

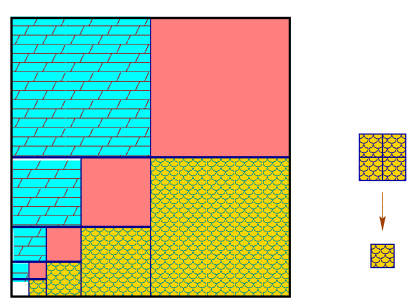


Slide 19

Anti-aliasing & Texture Mapping

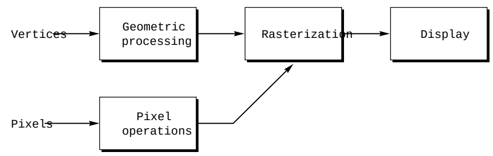
Mip-mapping: *multum in parvo* (Williams, 1983)

- ☆ Store many texture images.
- ☆ *i*-th image is obtained by scaling down the previous image by half along each axis.
- ☆ Effectively a 3D database.
- ☆ Given a pixel, search in the image with an appropriate resolution.



Slide 20

Texture Mapping in OpenGL



- ☆ Relies on the pipeline architecture
- ☆ Texture mapping is done at the rasterization stage.

glTexImage2D (GL_TEXTURE_2D, level, comp, w, h, border, format, type, tarray)

```
GLubyte my_texel[512][512];
glTexImage2D (GL_TEXTURE_2D, 0, 3, 512, 512, 0
              GL_RGB, GLuint, tarray);
glEnable(GL_TEXTURE_2D);
```

- ☆ *level*: Multiple levels of texture maps; 0 for one level.
- ☆ *comp*: integer between 1 and 4; specifies how many of R, G, B, and A components specified.
- ☆ *format*: Format of the texture map.

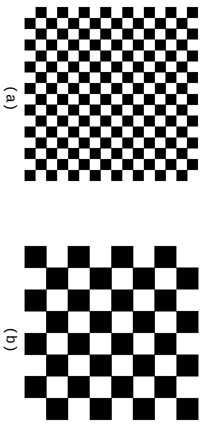
Slide 21

Texture Mapping in OpenGL

glTexCoord2f (s, t)

Assigns the two dimensional texture coordinates to a vertex.

```
glBegin(GL_QUAD);
  glTexCoord2f (0.0, 0.0);
  glVertex3f (x1, y1, z1);
  glTexCoord2f (1.0, 0.0);
  glVertex3f (x2, y2, z2);
  glTexCoord2f (1.0, 1.0);
  glVertex3f (x3, y3, z3);
  glTexCoord2f (0.0, 1.0);
  glVertex3f (x4, y4, z4);
glEnd
```



(a) (b)

Slide 22

Texture Mapping in OpenGL

- ☆ Texture objects
  glGenTextures(n, *names); **glBindTexture**(target, name);
- ☆ Repeating the texture pattern
  glTexParameteri(GL_TEXTURE_WRAP_S,GL_REPEAT)
  glTexParameteri (GL_TEXTURE_WRAP_S,GL_CLAMP)
  Use GL_TEXTURE_WRAP_T for t-coordinates
- ☆ Assign a pixel color to the nearest texel
  glTexParameterf (GL_TEXTURE2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST)
  glTexParameterf (GL_TEXTURE2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST)
- ☆ GL_LINEAR: Interpolates the color using a 2×2 average.

Slide 23

Bump Mapping

Perturb normals in the illumination model calculations.

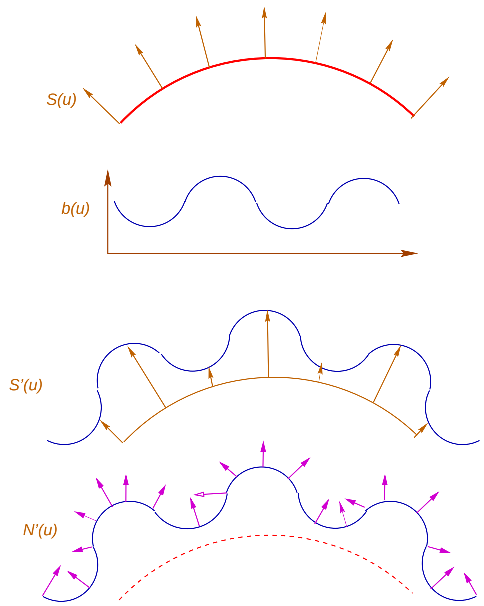
- ☆ $S(u, v)$: Parameterized surface
- ☆ $S_u = \partial S(u,v)/\partial u$, $S_v = \partial S(u,v)/\partial v$.
- ☆ $N(u,v) = S_u \times S_v$; $n = N/|N|$
- ☆ $b(u,v)$: Bump function

$$S'(u,v) = S(u,v) + b(u,v)\cdot n \qquad N'(u,v) = S'_u \times S_v$$

$$\begin{aligned}
S'_u &= \frac{\partial}{\partial u}(S(u,v) + b(u,v)\cdot n) \\
&= S_u + b_u \cdot n + b n_u \\
&\approx S_u + b_u \cdot n \\
S'_v &\approx S_v + b_v \cdot n \\
N' &\approx S_u \times S_v + b_u(S_u \times n) + b_v(n \times S_v) + \\
& \quad b_u b_v (n \times n) \\
&= S_u \times S_v + b_u(S_u \times n) + b_v(n \times S_v)
\end{aligned}$$

Slide 24

Bump Mapping



Slide 25

Bump Mapping

- ☆ Define bump functions analytically.
- ☆ Use look-up tables for bump functions.
- ☆ Approximate b_u, b_v with finite differences.
- ☆ Random pattern vs regular patterns.

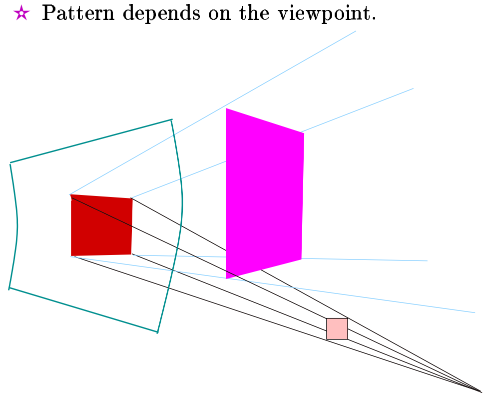
Displacement Mapping:

- ☆ Perturb normals as well as local coordinate system
- ☆ Used to render anisotropic objects.

Slide 26

Environment Mapping

- ☆ Reflects the surrounding environment on the surface of shiny objects.
- ☆ Similar to texture mapping.
- ☆ Pattern depends on the viewpoint.



- ☆ Store environment maps as 2D images.

Slide 27