Surface Details

- Incorporate fine details in the scene.
- Modeling with polygons is impractical.
- Map an image (texture/pattern) on the surface (Catmull1974); (Blin & Newell1976).

- **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.
Texture Maps

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

![Texture Map Diagram]
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).
Texture Maps

CPS124, 296: Computer Graphics  Surface Details  Page 4
Texture Mapping

Screen space \((x_s, y_s)\)

Object space \((x_o, y_o, z_o)\)

Forward mapping

Projection

Surface parameterization

Texture space \((u, v)\)

Pre-image of pixel

Inverse mapping
Texture Mapping

Forward mapping: (Texture scanning)

☆ Map texture pattern to the object space.

\[
\begin{align*}
  u &= f_u(s, t) = a_u s + b_u t + c, \\
  v &= f_v(s, t) = a_v s + b_v t + c.
\end{align*}
\]

☆ 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.
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.
Curves: Coordinates are represented as functions of one parameter.
\[
\gamma(t) = (x(t), y(t)), \quad t \in [a, b]
\]

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

Surfaces: Coordinates of each point is represented as a function of \( u \) and \( v \).
\[
S(u, v) = (x(u, v), y(u, v), z(u, v)).
\]

\( \star \) **Sphere:**
\[
S(u, v) = (r \cos u \cos v, r \cos u \sin v, r \sin u)
\]
\[
-\pi/2 \leq u \leq \pi/2, 0 \leq v \leq 2\pi.
\]

\( \star \) **Cylinder:**
\[
S(u, v) = (r \cos u, r \sin u, v)
\]
\[
0 \leq u \leq 2\pi, 0 \leq v \leq h.
\]
Inverse Mapping: An Example

\[ u = \theta, \ v = y \quad 0 \leq \theta \leq \pi/2, \ 0 \leq y \leq 1 \]

\[ x = \sin \theta, \ z = \cos \theta, \ y = v. \]

\[ M_{PV}^{-1} : u = \sin^{-1} x, \quad v = y. \]

Map texture origin to left bottom corner of the surface

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

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

\[ M_T^{-1} : s = \frac{2u}{\pi} \quad t = v. \]

\[ s = \frac{2}{\pi} \sin^{-1} x \quad t = z. \]
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., a cylinder.

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

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

$$ T''(x_i, y_i, z_i) \rightarrow O(x_w, y_w, z_w). $$
Two-Step Inverse Mapping

**Inverse mapping**: Apply them in the reverse order!
Four possible surfaces: Plane, cylinder, sphere, box.

**Cylinder:** \((\theta, z)\)

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

\(\star\) \(c, d\): Scaling factors.

\(\star\) Texture origin is mapped to the point \((\theta_0, z_0)\) on the cylinder.

**Sphere:** \((\theta, \varphi)\) stereographic projection.

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

\[
\alpha = \tan \varphi \cos \theta \Gamma \beta = \tan \varphi \sin \theta.
\]
$S$-Mapping

Diagram showing the mapping of 2D to 3D surfaces with arrows indicating the transformation of the $s$ and $t$ axes.
\( \Gamma \): Intermediate surface
\( P' \): Intersection of \( \Gamma \) with a ray \( \rho \) emanating from \( P \).

\( \rho \): Direction of:
(i) Reflection direction
(ii) Normal of \( O \) at \( P \)
(iii) \( CP \); \( C \): centroid of \( O \)
(iv) \(-\rho\): Normal of \( \Gamma \) at \( P' \)

\( \star \) (ii)(iii) are bad if \( \Gamma \) is cylinder.
3D Textures


STAR Define texture to be a 3D image.
  • Carving out an object from a 3D solid material.

STAR Ignoring scaling $M_T$ is identity.

STAR Distortion is minimized.

STAR Three dimensional vector fields can be mapped coherently.

STAR Texture is generated by a procedure.

STAR Example: Wood grain can be mapped as a set of cylinders with respect to a prespecified axis.
3D Textures

- Basic texture
- 3D noise
- Textured object
- 3D texture
- Textures object
- Object
Aliasing is particularly visible in periodic and coherent textures.
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 \textit{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.
**Anti-aliasing: Weight Functions**

**Blinn & Newell** (1976): Pyramid weight function

- Preimage of pixel
- Approximation
- Preimage of the center

**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.
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.
Texture Mapping in OpenGL

.vertexes → Geometric processing → Rasterization → Display

Pixels → Pixel operations

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

\texttt{glTexImage2D (GL\_TEXTURE\_2D level comp w h border format type tarray)}

\texttt{GLubyte my\_texel[512][512];}
\texttt{glTexImage2D (GL\_TEXTURE\_2D 0 3 512 512 0 GL\_RGB GL\_GLuint tarray);}  
\texttt{glEnable(GL\_TEXTURE\_2D);}

✫ \textit{level}: Multiple levels of texture maps; 0 for one level.
✫ \textit{comp}: integer between 1 and 4; specifies how many of R\text{G}B\text{A} components specified.
✫ \textit{format}: Format of the texture map.
Assigns the two dimensional texture coordinates to a vertex.

Texture Mapping in OpenGL
Texture objects

\[ \text{glGenTextures}(n\Gamma*\text{names}); \quad \text{glBindTexture}(\text{target}\Gamma\text{name}); \]

- Repeating the texture pattern

\[ \text{glTexParameteri}(\text{GL_TEXTURE_WRAP_S}, \text{GL_REPEAT}) \]
\[ \text{glTexParameteri}(\text{GL_TEXTURE_WRAP_S}, \text{GL_CLAMP}) \]

Use \text{GL_TEXTURE_WRAP_T} for \( t \)-coordinates

- Assign a pixel color to the nearest texel

\[ \text{glTexParameterf}(\text{GL_TEXTURE}\ 2D, \text{GL_TEXTURE_MAG_FILTER}, \text{GL_NEAREST}) \]
\[ \text{glTexParameterf}(\text{GL_TEXTURE}\ 2D, \text{GL_TEXTURE_MIN_FILTER}, \text{GL_NEAREST}) \]

- \text{GL_LINEAR}: Interpolates the color using a \( 2 \times 2 \) average.
Bump Mapping

Perturb normals in the illumination model calculations.

\[ S(u, v): \text{Parameterized surface} \]

\[ S_u = \frac{\partial S(u, v)}{\partial u} \quad S_v = \frac{\partial S(u, v)}{\partial v} \]

\[ N(u, v) = S_u \times S_v; \quad n = N/|N| \]

\[ b(u, v): \text{Bump function} \]

\[ S'(u, v) = S(u, v) + b(u, v) \cdot n \quad N'(u, v) = S'_u \times S_v \]

\[
S'_u = \frac{\partial}{\partial u} (S(u, v) + b(u, v) \cdot n) \\
= S_u + b_u \cdot n + bn_u \\
\approx S_u + b_u \cdot n
\]

\[
S'_v \approx S_v + b_v \cdot n
\]

\[
N' \approx S_u \times S_v + b_u (S_u \times n) + b_v (n \times S_v) + b_u b_v (n \times n) \\
= S_u \times S_v + b_u (S_u \times n) + b_v (n \times S_v)
\]
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.
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.