**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.

**Texture Maps**

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

---

**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).
Forward mapping: (Texture scanning)

\[ u = f_u(s,t) = a_u s + b_u t + c, \]
\[ v = f_v(s,t) = a_v s + b_v t + c. \]

- Map texture pattern to the object space.

- 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.

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 (glUnproject)
  - Computing \( M_T^{-1} \) is not easy.
Inverse Mapping: An Example

\[ u = \theta, v = y \quad 0 \leq \theta \leq \pi/2, 0 \leq y \leq 1 \]
\[ x = \sin \theta, z = \cos \theta, y = v. \]
\[ M_{uv}^{-1} : u = \sin^{-1} x, \quad v = y. \]
Map texture origin to left bottom corner of the surface
\[ u = -\pi/2, \quad v = 1. \]
Projected pixels are mapped to texture with \( M_{uv}^{-1} \):
\[ M_{uv}^{-1} : s = \frac{2u}{\pi} \quad t = v. \]
\[ s = \frac{2}{\pi} \sin^{-1} x \quad t = z. \]

Object to Texture Mapping

How does one define a reasonable \( M_{uv}^{-1} \)?
(Bier & Sloan, 1986): Two-step process:

**S-mapping**: Mapping from a 2D texture space to a simple 3D surface, e.g., cylinder.
\[ T(u, v) \rightarrow T(x_1, y_1, z_1). \]

**O-mapping**: Mapping from the 3D texture pattern onto the object surface.
\[ T'(x_1, y_1, z_1) \rightarrow O(x_w, y_w, z_w). \]
3D Textures


- 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.
Anti-aliasing & Texture Mapping

Aliasing is particularly visible in periodic and coherent textures.

With Antialiasing
Without Antialiasing

Two-step process:

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

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.

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

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

```c
GLuint my_tex[512][512];
gTexImage2D(GL_TEXTURE_2D, 0, 3, 512, 512, GL_RGB, GL_LUMINANCE, my_tex);
gEnable(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.

---

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

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

$$S_u' = \frac{\partial}{\partial u}(S(u,v) + b(u,v) \cdot n)$$
$$\approx S_u + b_u \cdot n + b_{nu}$$

$$S_v' \approx S_v + b_v \cdot n$$

$$N' \approx S_u \times S_v + b_u (S_u \times n) + b_v (n \times S_v) + b_{nu} (n \times n)$$
**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.