Texture Mapping

Motivation: Add interesting and/or realistic detail to surfaces of objects.
Problem: Fine geometric detail is difficult to model and expensive to render.
Idea: Modify various shading parameters of the surface by mapping a function (such as a 2D image) onto the surface.

Texture Mapping Example

- Given an image, think of it as a 2D function from $[0,1]^2$ (texture coordinates) to the RGB color space: $T(u,v) \rightarrow (r,g,b)$
- For each geometric primitive, define a mapping $M$ that maps points on the surface to texture coordinates: $M(x,y,z) = (u,v)$
- To shade a pixel corresponding to a point $(x,y,z)$ on the surface, use the color: $(r,g,b) = T(M(x,y,z))$

Texture Mapping an Image

- Texture:
- Result:
### Affected Parameters

- Final color
- Reflectance (either diffuse or specular)
- Surface normal (bump mapping)
- Transparency
- Reflected color (environment mapping)
- Any combination of the above

### Bump Mapping (Blinn 78)

**Smooth surface:**

**Bumpy surface:**

**Bump-mapped surface:**

### Bump Mapping

- Let \( P = P(u,v) \) be a smooth parametric surface, with normals \( N = N(u,v) \).
- Apply a bump map \( b = b(u,v) \):

\[
P' = P + bN
\]

\[
N' = P'_u \times P'_v
\]

\[
P'_u = \frac{\partial}{\partial u} (P + bN) = P'_u + b_u N + b N_u \approx P'_u + b_u N
\]

\[
P'_v = \frac{\partial}{\partial v} (P + bN) = P'_v + b_v N + b N_v \approx P'_v + b_v N
\]

### Bump Mapping (continued)

\[
N' \approx (P'_u + b_u N) \times (P'_v + b_v N)
\]

\[
= P'_u \times P'_v + b_u (N \times P'_v) + b_v (P'_u \times N) + b_u b_v (N \times N)
\]

\[
= N + b_u (N \times P'_v) + b_v (P'_u \times N)
\]
**Bump Mapping - Results**

- Certain objects have a natural parameterization (e.g., Bezier patches).
- Polygons (triangles): each vertex is assigned a pair of texture coordinates \((u,v)\). Inside, linear interpolation is used.
- How do we handle a more complex object?

---

**Object Parameterization**

- Step I: define a mapping between the texture and some intermediate surface:
  - plane
  - cylinder
  - sphere
  - cube
- Step II: Project intermediate surface onto object surface

---

**Two-Step Texture Mapping**  
*(Bier and Sloan 1986)*

- Step I: define a mapping between the texture and some intermediate surface:
  - plane
  - cylinder
  - sphere
  - cube
- Step II: Project intermediate surface onto object surface

---

**Intermediate Surface Projections**

![Diagram](image)

Figure 6.4. The four possible mappings that map the intermediate surface texture onto the object.
Minimization of Distortion
(Maillot, Yahia, Verroust 93)

- Given a parametric mapping \( \phi: U \rightarrow E^3 \)
  define the elastic deformation energy by the Green-Lagrange deformation tensor:
  \[
  E(U) = \int \int_U \left( \frac{\partial \phi}{\partial u} - 1 \right)^2 + 2 \left( \frac{\partial \phi}{\partial u} \cdot \frac{\partial \phi}{\partial v} \right)^2 + \left( \frac{\partial \phi}{\partial v} - 1 \right)^2 \ du \ dv
  \]

- We would like to find a mapping that minimizes the energy \( E \).

Triangulated Surfaces

- The mapping from/to texture space is affine over each triangle: \( M_i M_j M_k \rightarrow m_i m_j m_k \)
- Substituting into the Green-Lagrange formula gives a high-degree rational polynomial: solvable, but complicated.

Distance based energy

- A simpler discrete form of energy:
  \[
  E_l = \sum_{(i,j) \in \text{Edges}} \frac{\left( ||m_i - m_j||^2 - ||M_i - M_j||^2 \right)^2}{||M_i - M_j||^2}
  \]

- This is similar to the energy of a spring net.
- A normalizing term is used to accommodate triangles of different sizes.
**Surface based energy**

- An energy term defined by measuring the difference in signed areas of the triangles:

\[
E_s = \sum_{M_i, M_j, M_k} \left[ \det (\mathbf{m}_i, \mathbf{m}_j, \mathbf{m}_k) - \left\| \mathbf{M}_i \mathbf{M}_j \wedge \mathbf{M}_k \right\| \right]^2
\]

- The total distortion energy is a linear combination \( E = \alpha E_i + (1-\alpha) E_s \)

**Atlases**

- An atlas is a set of charts \( \{\phi_1, \ldots, \phi_n\} \)
- Each chart is a continuous mapping of a region on the surface onto a planar region (in texture space).
- The domains of these mappings cover the surface without overlap, except at boundaries.
- Discontinuities are allowed along boundaries.

**Atlas construction**

- Sort faces into buckets according to normals
- Build a connectivity graph to represent adjacencies between buckets
- Merge adjacent buckets, if sufficiently similar
- Flatten resulting regions, by orthogonal projection onto a plane.
Texture Aliasing

A single screen space pixel might correspond to many texels (texture elements):
Filtered Texture:

Resampling a signal

- Given a sampled (discrete) signal:
  - Reconstruct a continuous signal
  - Warp (apply mapping)
  - Prefilter warped signal
  - Sample
- In practice, some of the above stages are collapsed into a single convolution

Two special cases:

- Magnification: No real need in prefiltering. The main decision is what kind of reconstruction (interpolation) to use.
- Minification: No real need in reconstruction. The main problem is proper prefiltering.

Texture Pre-Filtering

- Problem: filtering the texture during rendering is too slow for interactive performance.
- Solution: pre-filter the texture in advance
  - Summed area tables - gives the average value of each axis-aligned rectangle in texture space
  - Mip-maps (tri-linear interpolation) - supported by most of today's texture mapping hardware
**Summed Area Tables (Crow 84)**

- A 2D table the size of the texture. At each entry (i,j), store the sum of all texels in the rectangle defined by (0,0) and (i,j).
- Calculate a bounding rectangle for the pre-image.
- Given any axis aligned rectangle, the sum of all texels is obtained from the summed area table:

\[
\text{area} = A - B - C + D
\]

**MIP-Maps (Williams 83)**

- Precompute a set of prefiltered textures (essentially an image pyramid).
- Based on the area of the pre-image of the pixel:
  - Select two "best" resolution levels
  - Use bilinear interpolation inside each level
  - Linearly interpolate the results
- Referred to as trilinear interpolation