Textures

• Introduced by Catmull in 1974
• Simulate Surface Detail / Micro Geometry
• Initially, map images to geometry
• Now generalization to bumps, normals, displacements, environments, volumes, …
Texture Mapping

Texture Space (u, v) 

Object Space (x, y, z) 

Screen Space (x_s, y_s, z_s)

Texture Image

Copyright © Amitabh Varshney
Texture Image

![Texture Image](http://titan.spaceports.com/~seymor)

Procedural Texture
Procedural Texture

Uwave = u % 2
= u & 0x01
Vwave = v % 2
= v & 0x01
Pix(u, v) = Uwave ⊕ Vwave

If u and v are n bits each,
Instead of $2^{2n}$ bits we only
need $2n$ bits + code

Capability Gap

Graph courtesy of Bill Dally
Texture Mapping

- Back to the problem of mapping label to a Can
- For now, let the Can be represented by a cylinder

Parametric Equation of a Cylinder:
\[ x = r \cos \theta, \ y = r \sin \theta, \ z = h, \ 0 \leq h \leq H, \ 0 \leq \theta \leq 2\pi \]

Let \( u = \theta / 2\pi, \ v = h/H, \ 0 \leq u, v \leq 1 \)
\[ x = r \cos 2\pi u, \ y = r \sin 2\pi u, \ z = Hv \]

Texture Mapping

From the previous equations:
\[ u = \tan^{-1}(y/x) \]
\[ v = z/H \]

So given object-space \((x, y, z)\), we can find \((u, v)\).
We then look up color at \((u, v)\) and place it at \((x_s, y_s, z_s)\)
General Texture Mapping

• In general, need a 2D parameterization of a 3D surface
  – Trivial for some objects: sphere, cylinder, cone, cube, …
  – Difficult for most objects
• Current schemes involve :
  – Two-stage mapping
  – Converting a 3D surface into an atlas of different parameterizations

Two-Stage Texture Mapping

• Introduced by Bier and Sloan (1986)
• \textit{S-Map}: Map from 2D texture space to an intermediate 3D surface (cylinder, sphere, cube, …)
  \[ T (u, v) \rightarrow S (r, \theta, \phi) \]
• \textit{O-Map}: Map from 3D intermediate space to 3D object:
  \[ S(r, \theta, \phi) \rightarrow O (x, y, z) \]
Spherical Mapping

Images courtesy, David Ebert, Purdue

Cylindrical Mapping

Images courtesy, David Ebert, Purdue
Pre-Filtering

• What if one screen-space pixel corresponds to multiple texture image pixels (texels)?
• Do successive averaging to build up a powers of 2 hierarchy (MIP-map) and index at the right scale factor into this hierarchy.

Bump Mapping

• Introduced by Blinn (1978)
Procedural Bump Mapping

Images courtesy, Alan Watt, 3D Computer Graphics

Slide 17  Lecture 4

Bump & Texture Mapping

Images courtesy, Alan Watt, 3D Computer Graphics

Slide 18  Lecture 4
**Bump Mapping**

- Consider a surface $P(u, v)$
- Its normal is given by: $N = P_u \times P_v$
- Perturb the surface by a bump map function $B(u, v)$ along its normals: $Q(u, v) = P(u, v) + B(u, v) \cdot N$
- Now compute and use the perturbed normal for illumination:
  \[ N' = Q_u \times Q_v = N + B_u N \times P_v - B_v N \times P_u \]

**Normal Mapping**

- Similar to bump mapping, except normals are directly specified instead of the bump map function
- Similar generalizations exist for Phong shading using texture mapping
Environment Mapping

- Introduced by Blinn & Newell (1976) as reflection mapping
- Compute the reflection vector at each mesh vertex:
  \[ \mathbf{R} = 2(\mathbf{N} \cdot \mathbf{V})\mathbf{N} - \mathbf{V} \]
- Interpolate the reflection vector across the triangle
- Modern GPUs provide hardware cube-maps for this
- Use the reflection vectors to index into the environment (texture) map

Environment Mapping

- Index into texture map not by (u, v) but by reflection rays

Image by Jim Blinn via debevec.org

Image by Aravind Kalaiah & Amitabh Varshney
Displacement Mapping

• In contrast to Bump Mapping, the actual geometry is displaced here
• The surface as well as silhouettes appear non-smooth
• Presents a challenge for deferred shading systems (when shading is performed after visibility)
Direct Mesh Representation

Let \( a = \{0, 0, 0\}, \ b = \{1, 1, 0\}, \ c = \{2, 0, 0\}, \ d = \{3, 1, 0\}, \ e = \{4, 0, 0\} \)

Simplest File Format

\[
t 3 \quad // \text{# triangles} = 3 \\
0, 0, 0, 1, 1, 0, 2, 0, 0 \quad // abc \\
1, 1, 0, 2, 0, 0, 3, 1, 0 \quad // bcd \\
2, 0, 0, 3, 1, 0, 4, 0, 0 \quad // cde
\]

More Generally:

\[
f n \quad // \text{# faces} = n \\
i_1 x_{11} y_{11} z_{11} x_{12} y_{12} z_{12} \ ... \ x_{i_{11}} y_{i_{11}} z_{i_{11}} \ // \text{first face has} \ i_1 \ \text{vertices} \\
i_2 x_{21} y_{21} z_{21} x_{22} y_{22} z_{22} \ ... \ x_{2_{i_{21}}} y_{2_{i_{21}}} z_{2_{i_{21}}} \ // \text{second face has} \ i_2 \ \text{vertices} \\
\ ... \\
\ ... \\
\ ... \\
i_n x_{n1} y_{n1} z_{n1} x_{n2} y_{n2} z_{n2} \ ... \ x_{n_{i_n1}} y_{n_{i_n1}} z_{n_{i_n1}} \ // \text{nth face has} \ i_n \ \text{vertices}
\]

For vertices with colors, normals etc:

\[
i_1 x_{11} y_{11} z_{11} r_{11} g_{11} b_{11} n_{x_{11}} n_{y_{11}} n_{z_{11}} \\
x_{12} y_{12} z_{12} r_{12} g_{12} b_{12} n_{x_{12}} n_{y_{12}} n_{z_{12}} \\
\ ... \\
\ ... \\
\ ... \\
i_n x_{n1} y_{n1} z_{n1} r_{n1} g_{n1} b_{n1} n_{x_{n1}} n_{y_{n1}} n_{z_{n1}}
\]
Indexed Mesh Representation

\[v\ 5\ //\ #\text{vertices} = 5\]
\[
\begin{array}{c}
0\ 0\ 0 \\
1\ 1\ 0 \\
2\ 0\ 0 \\
3\ 1\ 0 \\
4\ 0\ 0 \\
\end{array}
\]
\[t\ 3\ //\ #\text{triangles} = 3\]
\[
\begin{array}{c}
0\ 1\ 2\ //\ abc \\
1\ 2\ 3\ //\ bcd \\
2\ 3\ 4\ //\ cde \\
\end{array}
\]

Indexed Mesh Representation

\[v\ 5\ //\ #\text{vertices} = 5\]
\[
\begin{array}{c}
0\ 0\ 0 \\
1\ 1\ 0 \\
2\ 0\ 0 \\
3\ 1\ 0 \\
4\ 0\ 0 \\
\end{array}
\]
\[c\ 2\ //\ #\text{colors} = 2\]
\[
\begin{array}{c}
255\ 0\ 0\ //\ \text{red in byte rep} \\
0\ 255\ 0\ //\ \text{green in byte rep} \\
\end{array}
\]
\[t\ 3\ //\ #\text{triangles} = 3\]
\[
\begin{array}{c}
0\ 1\ 0\ 2\ 0\ //\ abc \\
1\ 0\ 2\ 0\ 3\ 0\ //\ bcd \\
2\ 3\ 4\ 1\ //\ cde \\
\end{array}
\]
Indexed Mesh Representation

- File format consists of a Vertex Array, Color Array, Normal Array, … and a Face Array with indices pointing to the above arrays.

Indexed face sets used in numerous file formats including VRML/X3D, Wavefront OBJ, …

v 5 // #vertices = 5
0 0 0
1 1 0
2 0 0
3 1 0
4 0 0
c 2 // # colors = 2
255 0 0 // red in byte rep
0 255 0 // green in byte rep
t 3 // # triangles = 3
0 1 0 2 0 // abc
1 0 2 3 0 // bcd
2 1 3 4 1 // cde