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


Procedural Texture
**Procedural Texture**

- $U_{\text{wave}} = u \mod 2 = u \& 0x01$
- $V_{\text{wave}} = v \mod 2 = v \& 0x01$
- $\text{Pix}(u, v) = U_{\text{wave}} \oplus V_{\text{wave}}$

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

---

**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:

\[
\begin{align*}
    x &= r \cos \theta, \\
    y &= r \sin \theta, \\
    z &= h, \quad 0 \leq h \leq H, \quad 0 \leq \theta \leq 2\pi
\end{align*}
\]

Let $u = \theta / 2\pi, \quad v = h/H, \quad 0 \leq u, v \leq 1$

\[
\begin{align*}
    x &= r \cos 2\pi u, \\
    y &= r \sin 2\pi u, \\
    z &= Hv
\end{align*}
\]

---

**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)
- **$S$-Map:** Map from 2D texture space to an intermediate 3D surface (cylinder, sphere, cube, …)
  \[ T(u, v) \rightarrow S(r, \theta, \phi) \]
- **$O$-Map:** Map from 3D intermediate space to 3D object:
  \[ S(r, \theta, \phi) \rightarrow O(x, y, z) \]
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

Bump & Texture Mapping
**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: $Q(u, v) = P(u, v) + B(u, v) N$
- 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:
  
  $R = 2(N \cdot V)N - V$
- 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

**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**

\[
\text{t 3 // # triangles = 3}
\]

\[
0, 0, 0, 1, 1, 0, 2, 0, 0 // abc
\]

\[
1, 1, 0, 2, 0, 3, 1, 0 // bcd
\]

\[
2, 0, 0, 3, 1, 0, 4, 0, 0 // cde
\]

Indexed Mesh Representation

\[
v 5 // #vertices = 5
\]

\[
0 0 0
\]

\[
1 1 0
\]

\[
2 0 0
\]

\[
3 1 0
\]

\[
4 0 0
\]

\[
\text{t 3 // # triangles = 3}
\]

\[
0 1 2 // abc
\]

\[
1 2 3 // bcd
\]

\[
2 3 4 // cde
\]

Indexed Mesh Representation

\[
v 5 // #vertices = 5
\]

\[
0 0 0
\]

\[
1 1 0
\]

\[
2 0 0
\]

\[
3 1 0
\]

\[
4 0 0
\]

\[
c 2 // # colors = 2
\]

\[
255 0 0 // red in byte rep
\]

\[
0 255 0 // green in byte rep
\]

\[
\text{t 3 // # triangles = 3}
\]

\[
0 0 1 2 // abc
\]

\[
1 0 2 0 // bcd
\]

\[
2 3 4 // cde
\]

Indexed Mesh Representation

\[
v 5 // #vertices = 5
\]

\[
0 0 0
\]

\[
1 1 0
\]

\[
2 0 0
\]

\[
3 1 0
\]

\[
4 0 0
\]

\[
c 2 // # colors = 2
\]

\[
255 0 0 // red in byte rep
\]

\[
0 255 0 // green in byte rep
\]

\[
\text{t 3 // # triangles = 3}
\]

\[
0 0 1 2 // abc
\]

\[
1 0 2 0 // bcd
\]

\[
2 3 4 // cde
\]