Chapter 9

Texture Mapping

The Rendering Pipeline

Geometry Database → Model/View Transform. → Lighting → Perspective Transform. → Clipping

Geometry Processing

Scan Conversion → Texturing → Depth Test → Blending → Frame-buffer

Rasterization → Fragment Processing
Texture Mapping

- Real life objects non-uniform in terms of color & normal
- To generate realistic objects - reproduce coloring & normal variations = **Texture**
- Can often replace complex geometric details

Color Texture Mapping

- Define color (RGB) for each point on object surface
- Two approaches
  - Surface texture map
  - Volumetric texture
Surface texture

- Define texture pattern over \((u,v)\) domain (Image)
  - Image – 2D array of “texels”
- Assign \((u,v)\) coordinates to each point on object surface
- For free-form – use inverse of surface function
- For polygons (triangle)
  - Inside – use barycentric coordinates
  - For vertices need mapping function

![Image of a baboon head with texture mapping applied]

Texture Mapping - OpenGL

- Texture Coordinates
  - generation at vertices
    - specified by programmer or artist
      - `glTexCoord2f(s,t)`
      - `glVertexf(x,y,z)`
  - generate as a function of vertex coords
  - interpolated across triangle (like R,G,B,Z)
    (well, not quite...)
Texture Mapping

- Texture coordinate interpolation
  - Perspective foreshortening problem
  - Also problematic for color interpolation, etc.

\[
\begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & \frac{d}{d-\alpha} & 1 \\
0 & 0 & \frac{-\alpha d}{d-\alpha} & 0 \\
\end{pmatrix}
\begin{pmatrix}
x \\
y \\
z \\
1 \\
\end{pmatrix} = \begin{pmatrix}
x, y, \frac{(z-\alpha)d}{d-\alpha}, \frac{z}{d} \\
\end{pmatrix}
\]

Perspective - Reminder

- Matrix formulation

\[
(x', y', z', w') = \begin{pmatrix}
x, y, \frac{(z-\alpha)d}{d-\alpha}, \frac{z}{d} \\
\end{pmatrix}
\]

\[
(x_p, y_p, z_p) = \frac{d^2}{z/d' - z'/d-\alpha} \begin{pmatrix}
x \\
y \\
(z/d' - z'/d-\alpha) \\
1 \\
\end{pmatrix}
\]
Texture Coordinate Interpolation

- Perspective Correct Interpolation
  - $\alpha$, $\beta$, $\gamma$ :
    Barycentric coordinates of point $P$
  - $s_0$, $s_1$, $s_2$ : texture coordinates of vertices
  - $w_0$, $w_1$, $w_2$ : homogenous coordinate of vertices

$$s = \frac{\alpha \cdot s_0 / w_0 + \beta \cdot s_1 / w_1 + \gamma \cdot s_2 / w_2}{\alpha / w_0 + \beta / w_1 + \gamma / w_2}$$

Reconstruction

(image courtesy of Kiriakos Kutulakos, U Rochester)
Reconstruction

- How to deal with:
  - pixels that are much larger than texels? (apply filtering, “averaging”)
  - pixels that are much smaller than texels? (interpolate)

MIP-mapping

Use “image pyramid” to precompute averaged versions of the texture

Without MIP-mapping

With MIP-mapping
Volumetric Texture

- Define texture pattern over 3D domain - 3D space containing the object
  - Texture function can be digitized or procedural
  - For each point on object compute texture from point location in space
- Common for natural material/irregular textures (stone, wood, etc...)

Texture Parameters

- In addition to color can control other material/object properties
  - Reflectance (either diffuse or specular)
  - Surface normal (bump mapping)
  - Transparency
- Reflected color (environment mapping)
Normal – Bump Mapping

- Object surface often not smooth – to recreate correctly need complex geometry model
- Can control shape “effect” by locally perturbing surface normal
  - Random perturbation
  - Directional change over region

Normal map

- 4M faces
- 8K faces
- 8K faces, normal-mapped
Environmental Mapping

- Cheap way to achieve reflective effect
  - Generate image of surrounding
  - Map to object as texture