Chapter 9

Texture Mapping

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

Texture Mapping - OpenGL

- Texture Coordinates
  - generation at vertices
    - specified by programmer or artist
      - \texttt{glTexCoord2f}(s,t)
      - \texttt{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.

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

\[
\begin{align*}
\alpha & = \frac{s_0}{w_0} + \frac{\beta}{w_1} + \frac{\gamma}{w_2} \\
\beta & = \frac{s_1}{w_1} + \frac{\gamma}{w_2} \\
\gamma & = \frac{s_2}{w_2} 
\end{align*}
\]

Perspective - Reminder

- Matrix formulation

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & d \\
0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
s_x \\
s_y \\
s_z \\
w
\end{bmatrix}
= 
\begin{bmatrix}
x \\
y \\
z \\
w
\end{bmatrix}
\]

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

\[
\begin{align*}
\alpha & = \frac{s_0}{w_0} + \frac{\beta}{w_1} + \frac{\gamma}{w_2} \\
\beta & = \frac{s_1}{w_1} + \frac{\gamma}{w_2} \\
\gamma & = \frac{s_2}{w_2} 
\end{align*}
\]

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

---

**Environmental Mapping**

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