Texture Mapping

**CPSC 314**

- Images attached to geometry
- "Texels": texture elements
- Adds visual detail, substitute for geometric detail

Texture Coordinates

- Generation at vertices
  - Specified by programmer or artist
    - \( \text{glTexCoord2f}(s, t) \)
    - \( \text{glVertexf}(x, y, z) \)
  - Generate as a function of vertex coords
    - \( \text{glTexGeni}(), \text{glTexGenfv}() \)
    - \( s = a*x + b*y + c*z + d*h \)
- Interpolated across triangle (like R,G,B,Z)
  (well, not quite...)
Texture Mapping

Texture Coordinate Interpolation
- perspective foreshortening problem
- also problematic for colour interpolation, etc.

Perspective Correct Interpolation
- $\alpha, \beta, \gamma$:
  - Barycentric coordinates of a point $P$ in a triangle
- $s_0, s_1, s_2$ : texture coordinates
- $w_0, w_1, w_2$ : homog coordinates

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

Textures of other dimensions
- 1D: represent isovalues
  - e.g.: contour lines, temp, ...
  - $\text{glTexCoord1f}(s)$
Texture Mapping

Textures of other dimensions
- 3D: solid textures
  - e.g.: wood grain, medical data, ...
  \[ \text{glTexCoord3f}(s, t, r) \]
- 4D: 3D + time, projecting textures
  \[ \text{glTexCoord3f}(s, t, r, q) \]

Texture Coordinate Transformations

Motivation:
- Change scale, orientation of texture on an object

Approach:
- texture matrix stack
  - 4x4 matrix stack
  - transforms specified (or generated) tex coords
    \[ \text{glMatrixMode}( \text{GL_TEXTURE} ) ; \]
    \[ \text{glLoadIdentity}() ; \]
    ...

Example:

\[ \begin{matrix}
(0,0) & (1,0) \\
(0,4) & (4,4)
\end{matrix} \]

\[ \begin{matrix}
(0,1) & (1,1) \\
(0,3) & (3,3)
\end{matrix} \]

\[ \begin{matrix}
(0,0) & (4,0) \\
(0,0) & (4,0)
\end{matrix} \]

\[ \text{glScalef}(4.0, 4.0, ?); \]

Projective Transformations
- can do projective transformations
- tex coord \((s, t, r, q)\) : \(q \leftrightarrow h\)
Texture Coordinate Transformations

Example:

| Brabec and Heidrich | Brabec and Heidrich |

Texture Lookup

**Issue:**
- What happens to fragments with \( s \) or \( t \) outside the interval \([0...1]\)?

**Multiple choices:**
- Take only fractional part of texture coordinates
  - Cyclic repetition of texture to tile whole surface
    \[ \text{glTexParameter( ..., GL_TEXTURE_WRAP_S, GL_REPEAT )} \]
- Clamp every component to range \([0...1]\)
  - Re-use color values from border of texture image
    \[ \text{glTexParameter( ..., GL_TEXTURE_WRAP_S, GL_CLAMP )} \]

Reconstruction

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

(image courtesy of Kiriakos Kutulakos, U Rochester)
**MIP-mapping**

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

Without MIP-mapping

With MIP-mapping

**MIP mapping**

**Problem:**

- A MIP-map level selects the same minification factor for both the $s$ and the $t$ direction (isotropic filtering)
- In reality, perspective foreshortening (amongst other reasons) can cause different scaling factors for the two directions

**Which resolution to choose:**

- MIP-mapping: take resolution corresponding to the smaller of the sampling rates for $s$ and $t$
  - Avoids aliasing in one direction at cost of blurring in the other direction
- Better: anisotropic texture filtering
  - Also uses MIP-map hierarchy
  - Choose larger of sampling rates to select MIP-map level
  - Then use more samples for that level to avoid aliasing
  - Maximum anisotropy (ratio between $s$ and $t$ sampling rate) usually limited (e.g. 4 or 8)