AMCS / CS 247 – Scientific Visualization
Lecture 10: (GPU) Texture Mapping

Markus Hadwiger, KAUST
Reading Assignment #5 (until Oct. 8)

Read (required):

- Real-Time Volume Graphics, Chapter 2 (*GPU Programming*)
- Real-Time Volume Graphics, Chapter 5.3 (*Gradient-Based Illumination*)
- Real-Time Volume Graphics, Chapter 5.4.1 (*Blinn-Phong Illumination*)
Gradient Reconstruction

We need to reconstruct the derivatives of a continuous function given as discrete samples.

Central differences
- Cheap and quality often sufficient (2+2+2 neighbors in 3D)

Discrete convolution filters on grid
- Image processing filters; e.g. Sobel (3x3x3 neighbors)

Continuous convolution filters
- Derived continuous reconstruction filters
- E.g., the cubic B-spline and its derivatives (4x4x4 neighbors)
Example: Gaussian Kernel (1D)

- $G(x)$
- $G'(x)$
- $G''(x)$
2D Gaussian Kernel

\[ G(x_1, x_2) \]
2D Gaussian Derivative Kernels

\[ \frac{\partial G(x_1, x_2)}{\partial x_1} \]

\[ \frac{\partial G(x_1, x_2)}{\partial x_2} \]
Texture Mapping

Rage / id Tech 5 (id Software)
Special case of linear combination

$$\alpha_1 x_1 + \alpha_2 x_2 + \cdots + \alpha_n x_n$$

$$\alpha_i \geq 0$$

$$\alpha_1 + \alpha_2 + \cdots + \alpha_n = 1.$$  

The weights $\alpha_i$ are also the (normalized) barycentric coordinates.

They vary linearly $\rightarrow$ linear interpolation.
Bi-linear Interpolation

nearest-neighbor

Analog in 3D:
tri-linear interpolation

bi-linear interpolation
2D Texture Mapping

For each fragment:
interpolate the texture coordinates
(barycentric)

Or:
Use arbitrary, computed coordinates

Texture-Lookup:
interpolate the texture data
(bi-linear)

Or:
Nearest-neighbor for “array lookup”
3D Texture Mapping

For each fragment:
interpolate the texture coordinates *(barycentric)*

Or:
Use arbitrary, computed coordinates

**Texture-Lookup:**
interpolate the texture data *(tri-linear)*

Or:
Nearest-neighbor for “array lookup”
Shading Recap

- Flat shading
  - compute light interaction per polygon
  - the whole polygon has the same color
- Gouraud shading
  - compute light interaction per vertex
  - interpolate the colors
- Phong shading
  - interpolate normals per pixel
- Remember: difference between
  - Phong Lighting Model (specular lighting)
  - Phong Shading
Traditional OpenGL Lighting

- Phong lighting model at each vertex (glLight, …)
- Local model only (no shadows, radiosity, …)
- ambient + diffuse + specular (glMaterial!)

Fixed function: Gouraud shading
  - Note: need to interpolate specular separately!

Phong shading: evaluate Phong lighting model in fragment shader (per-fragment evaluation!)
Why Texturing?

- Idea: enhance visual appearance of surfaces by applying fine / high-resolution details.
OpenGL Texture Mapping

- Basis for most real-time rendering effects
- Look and feel of a surface
- Definition:
  - A *regularly sampled function* that is *mapped* onto every *fragment* of a surface
  - Traditionally an image, but…
- Can hold arbitrary information
  - Textures become general data structures
  - Sampled and interpreted by fragment programs
  - Can render into textures → important!
Types of Textures

- Spatial layout
  - Cartesian grids: 1D, 2D, 3D, 2D_ARRAY, …
  - Cube maps, …

- Formats (too many), e.g. OpenGL
  - GL_LUMINANCE16_ALPHA16
  - GL_RGB8, GL_RGBA8, …: integer texture formats
  - GL_RGB16F, GL_RGBA32F, …: float texture formats
  - compressed formats, high dynamic range formats, …

- External format vs. internal (GPU) format
  - OpenGL driver converts from external to internal
Texturing: General Approach

Texture space \((u,v)\)  
Object space \((x_O,y_O,z_O)\)  
Image Space \((x_I,y_I)\)

Parametrization  
Rendering (Projection etc.)
Texture Wrap Mode

- How to extend texture beyond the border?
- Border and repeat/clamp modes
- Arbitrary \((s,t,\ldots) \rightarrow [0,1] \times [0,1] \rightarrow [0,255] \times [0,255]\)

repeat  mirror/repeat  clamp  border
Texture Reconstruction: Magnification

- Bilinear reconstruction for texture magnification ($D < 0$) ("upsampling")
- Weight adjacent texels by distance to pixel position

\[ T(u+du,v+dv) = du \cdot dv \cdot T(u+1,v+1) + du \cdot (1-dv) \cdot T(u+1,v) + (1-du) \cdot dv \cdot T(u,v+1) + (1-du) \cdot (1-dv) \cdot T(u,v) \]
Magnification (Bilinear Filtering Example)

Original image

Nearest neighbor  Bilinear filtering
Solid Texturing

texture defined in 3D
every position in space has a color
coherent textures across corners
Solid Texturing Examples

examples for application of 3D textures on a scull and a face
Thank you.

Thanks for material

- Helwig Hauser
- Eduard Gröller
- Daniel Weiskopf
- Torsten Möller
- Ronny Peikert
- Philipp Muigg
- Christof Rezk-Salama