Texture Mapping

Wolfgang Heldrich

Course News

Assignment 2
• Due Monday!

Quiz 2 MOVED!
• Friday, March 13 (instead of Wed, March 11)
• Office hours on Wednesday, Thursday (Mar 11/12)
• Out of town Mon, Mar 9

Reading (this week)
• No new reading this week

Reading (next week)
• Chapter 11 (wo 11.8)

The Rendering Pipeline

Alpha Blending (OpenGL)

Parameters:
• s = source color
• d = destination color
• b = source blend factor
• c = dest blend factor
• d' = bs + cd

Where
• “Source” means “color/alpha of currently rendered primitive”
• “Destination” means framebuffer value

Over operator

• \( d'' = a_s s + (1-a_s)d \)
• Examples: \( a_s=0.4 \) \( a_s=1.0 \)

A over B: \( d''=0.4\alpha C_s + (0.4)\alpha C_B \)

Comparisons from previous

A B

A over B: \( d''=1\alpha C_B + (0)\alpha C_A \)

B over A: \( d''=1\alpha C_B + (0)\alpha C_A \)

Double Buffering

Framebuffer:
• Piece of memory where the final image is written
• Problem:
  – The display needs to read the contents, cyclically, while the GPU is already working on the next frame
  – Could result in display of partially rendered images on screen
• Solution:
  – Have TWO buffers
    ▪ Currently displayed (front buffer)
    ▪ Render target for the next frame (back buffer)
The Rendering Pipeline

Texture Mapping
- Real life objects have nonuniform colors, normals
- To generate realistic objects, reproduce coloring & normal variations = texture
- Can often replace complex geometric details

Texture Mapping
- Introduced to increase realism
  - Lighting/shading models not enough
- Hide geometric simplicity
  - Images convey illusion of geometry
  - Map a brick wall texture on a flat polygon
  - Create bumpy effect on surface
- Associate 2D information with 3D surface
  - Point on surface corresponds to a point in texture
  - "Paint" image onto polygon

Color Texture Mapping
- Define color (RGB) for each point on object surface
- Two approaches
  - Surface texture map (2D)
  - Volumetric texture (3D)

Surface (2D) Textures: Texture Coordinates
- Texture image: 2D array of color values (texels)
- Assigning texture coordinates (s, t) at vertex with object coordinates (x, y, z, w)
  - Use interpolated (s, t) for texel lookup at each pixel
  - Use value to modify a polygon’s color
    - Or other surface property
  - Specified by programmer or artist
    $\text{glTexCoord2f}(s, t)$
    $\text{glVertex}(x, y, z, w)$

Texture Mapping Example

Texture Lookup: Tiling and Clamping

**What if s or t is outside the interval [0...1]?**

**Multiple choices**

- Use fractional part of texture coordinates
  - Cyclic repetition of texture to tile whole surface
    
- Clamp every component to range [0...1]
  - Re-use color values from texture image border

Texture Coordinate Transformation

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

**Approach**
- Texture matrix stack
- Transforms specified (or generated) tex coords
  
- More flexible than changing (s,t) coordinates

Texture Functions

Once you have value from the texture map, can:
- Directly use as surface color: GL_REPLACE
  - Throw away old color, lose lighting effects
- Modulate surface color: GL_MODULATE
  - Multiply old color by new value, keep lighting info
  - Texturing happens after lighting, not before
- Use as surface color, modulate alpha: GL_BLEND
  - Like replace, but support texture transparency
- Blend surface color with another: GL_BLEND
  - New value controls which of 2 colors to use

Specify desired behavior with glTexParameter (GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, <mode>)
**Texture Pipeline**

- Object position: (-2.3, 7.1, 17.7)
- Parameter space: (0.32, 0.29)
- Transformed parameter space: (0.52, 0.49)

- Texel space: (81, 74)
- Texel color: (0.9, 0.8, 0.7)
- Final color: (0.45, 0.4, 0.35)
- Object color: (0.5, 0.5, 0.5)

**Texture Objects and Binding**

- **Texture object**
  - An OpenGL data type that keeps textures resident in memory and provides identifiers to easily access them
  - Provides efficiency gains over having to repeatedly load and reload a texture
  - You can prioritize textures to keep in memory
  - OpenGL uses least recently used (LRU) if no priority is assigned

- **Texture binding**
  - Which texture to use right now
  - Switch between preloaded textures

**Basic OpenGL Texturing**

Create a texture object and fill it with texture data:

- `glGenTextures` (num, &indices) to get identifiers for the objects
- `glBindTexture(GL_TEXTURE_2D, identifier)` to bind
  - Following texture commands refer to the bound texture
- `glTexParameteri(GL_TEXTURE_2D, ..., ...)` to specify parameters for use when applying the texture
- `glTexImage2D(GL_TEXTURE_2D, ..., ...)` to specify the texture data (the image itself)

**Basic OpenGL Texturing (cont.)**

- **Enable texturing:**
  - `glEnable(GL_TEXTURE_2D)`

- **State how the texture will be used:**
  - `glTexParameteri(...)`

- **Specify texture coordinates for the polygon:**
  - Use `glTexCoord2f(s, t)` before each vertex:
  - `glVertex3f(x, y, z);`

**Low-Level Details**

- **Enable texturing:**
  - `glEnable(GL_TEXTURE_2D)`

- **State how the texture will be used:**
  - `glTexParameteri(...)`

- **Specify texture coordinates for the polygon:**
  - Use `glTexCoord2f(s, t)` before each vertex:
  - `glVertex3f(x, y, z);`

**Texture Mapping**

- **Texture coordinate interpolation**
  - Perspective foreshortening problem
Interpolation: Screen vs. World Space

Screen space interpolation incorrect
- Problem ignored with shading, but artifacts more visible with texturing

Texture Coordinate Interpolation

Perspective correct interpolation
- \( \alpha, \beta, \gamma \):
  - Barycentric coordinates of a point \( P \) in a triangle
- \( x_0, x_1, x_2 \):
  - Texture coordinates of vertices
- \( w_0, w_1, w_2 \):
  - Homogeneous coordinates of vertices

\[
\begin{align*}
\alpha &= x_0 / w_0, \\
\beta &= x_1 / w_1, \\
\gamma &= x_2 / w_2
\end{align*}
\]

Texture Parameters

In addition to color can control other material/object properties
- Surface normal (bump mapping)
- Reflected color (environment mapping)

Bump Mapping: Normals As Texture

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

Bump Mapping

Original surface

A bump map

Lengthening or shortening \( O(u) \) using \( B(u) \)

The vectors to the ‘new’ surface
Displacement Mapping

Bump mapping gets silhouettes wrong
  - Shadows wrong too

Change surface geometry instead
  - Need to subdivide surface

GPU support
  - Bump and displacement mapping not directly supported: require per-pixel lighting
  - However: modern GPUs allow for programming both yourself

Environment Mapping

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

Sphere Mapping

Texture is distorted fish-eye view
  - Point camera at mirrored sphere
  - Spherical texture mapping creates texture coordinates that correctly index into this texture map

Cube Mapping

6 planar textures, sides of cube
  - Point camera in 6 different directions, facing out from origin

Cube Mapping

Direction of reflection vector \( v \) selects the face of the cube to be indexed
  - Co-ordinate with largest magnitude
    - e.g., the vector \((-0.2, 0.5, -0.84)\) selects the \(-Z\) face
  - Remaining two coordinates (normalized by the 3rd coordinate) selects the pixel from the face.
    - E.g., \((-0.2, 0.5)\) gets mapped to \((0.38, 0.80)\).

Difficulty in interpolating across faces
**Volumetric (3D) Texture**

*Define texture pattern over 3D domain - 3D space containing the object*
- Texture function can be **sampled**
  - 3D table of texels
- Or **procedural**
  - A function describes the color at each point
  - Implemented in special shading language

*Common for natural material/irregular textures (stone, wood, etc...)*

**Procedural Textures**

*Generate “image” on the fly, instead of loading from disk*
- Also called **shader**
- Often saves space
- Allows arbitrary level of detail
  - “magnification” not an issue
  - “minification” less so than for sampled representation
- But can be quite slow for complicated shaders

**Volumetric Bump Mapping**

Marble

Bump

**Volumetric Texture Mapping**

*In Hardware:*
- Sampled 3D textures supported very much analogously to 2D textures:
  - glTexCoord3f, glTexImage3f...
- Procedural textures supported with modern GPUs
  - More in upcoming lectures

**Texture Lookup – Sampling & Reconstruction**

- How to deal with:
  - **Pixels** that are much larger than texels?
    - Apply filtering, “averaging”
    - “Minification”
  - **Pixels** that are much smaller than texels?
    - Interpolate
    - “Magnification”
Magnification: Interpolating Textures
- Nearest neighbor
- Bilinear
- Hermite (cubic)

Minification: MIPmapping
use "image pyramid" to precompute averaged versions of the texture
store whole pyramid in single block of memory

MIPmaps
Multum in parvo -- many things in a small place
- Prespecify a series of prefiltred texture maps of decreasing resolutions
- Requires more texture storage
- Avoid shimmering and flashing as objects move

gluBuild3DMIPmaps
- Automatically constructs a family of textures from original texture size down to 1x1
with

MIPmap storage
only 1/3 more space required

Coming Up:
Next week
- More texture mapping
- Sampling & reconstruction