Texture Mapping

Wolfgang Heldrich

Course News

Assignment 2
- Due today

Assignment 3 (project)
- Out last Friday
- Start thinking about a project soon!

Quiz 2 MOVED!
- Friday, March 13 (instead of Wed, March 11)

Reading
- Chapter 11 (w/o 11.8)

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 Example

Example Texture Map

glTexCoord2d(0,0);
glVertex3d (0, -2, -2);
glTexCoord2d(1,1);
glVertex3d (0, 2, 2);
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
    `glTexParameteri(...) GL_TEXTURE_WRAP_S, GL_REPEAT, ...
    glTexParameteri(...) GL_TEXTURE_WRAP_T, GL_REPEAT, ...)`
  - Clamp every component to range [0 ... 1]
    `glTexParameteri(...) GL_TEXTURE_WRAP_S, GL_CLAMP, ...
    glTexParameteri(...) GL_TEXTURE_WRAP_T, GL_CLAMP, ...)`

- Re-use color values from texture image border
  `glTexParameteri(...) GL_TEXTURE_WRAP_S, GL_CLAMP_TO_BORDER, ...
  glTexParameteri(...) GL_TEXTURE_WRAP_T, GL_CLAMP_TO_BORDER, ...

Texel color

(0.9, 0.8, 0.7)

Object position

(-2.3, 7.1, 17.7)

Parameter space

(0.32, 0.29)

Transformed parameter space

(0.52, 0.49)

Final color

(0.45, 0.4, 0.35)

Object color

(0.5, 0.5, 0.5)

Textel space

(81, 74)

Texture Coordinate Transformation

Motivation

- Change scale, orientation of texture on an object

Approach

- Texture matrix stack
  - Transforms specified (or generated) tex coords
    `glMatrixMode(GL_TEXTURE);`
    `glLoadIdentity();`
    `glTranslatef();`
  - More flexible than changing (s,t) coordinates

Texture Pipeline

(x, y, z) Object position (-2.3, 7.1, 17.7) (s, t) Parameter space (0.32, 0.29) (s', t') 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)

Low-Level Details

Large range of functions for controlling layout of texture data

- State how the data in your image is arranged
  - e.g. glPixelStore(GL_UNPACK_ALIGNMENT, 1) tells OpenGL not to skip bytes at the end of a row
  - You must state how you want the texture to be put in memory: how many bits per "pixel", which channels...

Textures must have a size of power of 2

- Common sizes are 32x32, 64x64, 256x256
- But don’t need to be square, i.e. 32x64 is fine
- Smaller uses less memory, and there is a finite amount of texture memory on graphics cards

Texture Mapping

Text 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
- Barycentric coordinates of a point $P$ in a triangle:
- $x0, x1, x2$:
  - Texture coordinates of vertices
  - $w0, w1, w2$:
  - Homogeneous coordinates of vertices

$$V(x', y')$$

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

- $O(u)$: Original surface
- $N(u)$: 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 \( \mathbf{r} \) selects the face of the cube to be indexed**
  - Co-ordinate with largest magnitude
    - \( \text{e.g., the vector } [-0.2, 0.5, -0.84] \) selects the \(-2\) face
  - Remaining two coordinates (normalized by the 3rd coordinate) selects the pixel from the face
    - \( \text{E.g., } [-0.2, 0.5] \) 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:
  - `gTexCoord3f, gTexImage3f`, ...
- 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 prefiltered texture maps of decreasing resolutions
  - Requires more texture storage
  - Avoid shimmering and flashing as objects move
  - `gluBuild1dSMIPmaps`
    - Automatically constructs a family of textures from original texture size down to 1x1 with

**MIPmap storage**
- only 1/3 more space required

**Sampling & Reconstruction**
- Most things in the real world are continuous
- Everything in a computer is discrete
- The process of mapping a continuous function to a discrete one is called sampling
- The process of mapping a discrete function to a continuous one is called reconstruction
- Rendering an image requires sampling and quantization
- Displaying an image involves reconstruction
Line Segments
- We tried to sample a line segment so it would map to a 2D raster display
- We quantized the pixel values to 0 or 1
- We saw stair steps, or jaggies

Unweighted Area Sampling
Shade pixels wrt area covered by thickened line
Equal areas cause equal intensity, regardless of distance from pixel center to area
- Rough approximation formulated by dividing each pixel into a finer grid of pixels
  Primitive cannot affect intensity of pixel if it does not intersect the pixel

Weighted Area Sampling
Intuitively, pixel cut through the center should be more heavily weighted than one cut along corner
Weighting function, \( W(x, y) \)
- Specifies the contribution of primitive passing through the point \((x, y)\) from pixel center

Images
An image is a 2D function \( I(x, y) \)
- Specifies intensity for each point \((x, y)\)
- (we consider each color channel independently)

Image Sampling and Reconstruction
- Convert \( I(x, y) \) to discrete set of samples
- Display hardware reconstructs samples into continuous image
  - Finite sized source of light for each pixel
**Point Sampling an Image**

- Simplest sampling is on a grid
- Sample depends solely on value at grid points

**Point Sampling**

Multiply sample grid by image intensity to obtain a discrete set of points, or samples.

**Sampling Errors**

Some objects missed entirely, others poorly sampled
- Could try unweighted or weighted area sampling
- But how can we be sure we show everything?

Need to think about entire class of solutions!

**Image As Signal**

Image as spatial signal
- 2D raster image
  - Discrete sampling of 2D spatial signal
- 1D slice of raster image
  - Discrete sampling of 1D spatial signal

**Sampling Theory**

How would we generate a signal like this out of simple building blocks?

Theorem
- Any signal can be represented as an (infinite) sum of sine waves at different frequencies

**Coming Up:**

Wednesday / Friday
- More sampling & reconstruction