Textures I

Week 9, Fri Mar 19

http://www.ugrad.cs.ubc.ca/~cs314/Vjan2010
News

• extra TA office hours in lab for Q&A
  • Mon 10-1, Tue 12:30-3:30 (Garrett)
  • Tue 3:30-5, Wed 2-5 (Kai)
  • Thu 12-3:30 (Shailen)
  • Fri 2-4 (Kai)
Reading for Texture Mapping

- FCG Chap 11 Texture Mapping
  - except 11.7 (except 11.8, 2nd ed)
- RB Chap Texture Mapping
Review: Back-face Culling

VCS

NDCS

works to cull if $N_Z > 0$
Review: Invisible Primitives

• why might a polygon be invisible?
  • polygon outside the field of view / frustum
    • solved by clipping
  • polygon is back-facing
    • solved by backface culling
  • polygon is occluded by object(s) nearer the viewpoint
    • solved by hidden surface removal
Review: Rendering Pipeline

Geometry Database → Model/View Transform. → Lighting → Perspective Transform. → Clipping

Scan Conversion → Texturing → Depth Test → Blending → Frame-buffer
Review: Blending/Compositing

- how might you combine multiple elements?
- foreground color A, background color B
Premultiplying Colors

- specify opacity with alpha channel: (r,g,b,α)
  - α=1: opaque, α=.5: translucent, α=0: transparent

- A over B
  - \( \mathbf{C} = \alpha \mathbf{A} + (1-\alpha) \mathbf{B} \)

- but what if \( \mathbf{B} \) is also partially transparent?
  - \( \mathbf{C} = \alpha \mathbf{A} + (1-\alpha) \beta \mathbf{B} = \beta \mathbf{B} + \alpha \mathbf{A} + \beta \mathbf{B} - \alpha \beta \mathbf{B} \)
  - \( \gamma = \beta + (1-\beta)\alpha = \beta + \alpha - \alpha\beta \)
    - 3 multiplies, different equations for alpha vs. RGB

- premultiplying by alpha
  - \( \mathbf{C'} = \gamma \mathbf{C}, \mathbf{B'} = \beta \mathbf{B}, \mathbf{A'} = \alpha \mathbf{A} \)

  - \( \mathbf{C'} = \mathbf{B'} + \mathbf{A'} - \alpha \mathbf{B'} \)
    - \( \gamma = \beta + \alpha - \alpha\beta \)
      - 1 multiply to find \( \mathbf{C} \), same equations for alpha and RGB
Texturing
Rendering Pipeline

Geometry Processing

Geometry Database → Model/View Transform. → Lighting → Perspective Transform. → Clipping

Scan Conversion → Texturing → Depth Test → Blending → Frame-buffer

Rasterization → Fragment Processing
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
  - volumetric texture
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
    
    ```
    glTexCoord2f(s,t)
    glVertexf(x,y,z,w)
    ```

![Image of a baboon face with a 3D model image]
Texture Mapping Example
Example Texture Map

\[ \text{glTexCoord2d}(0,0); \]
\[ \text{glVertex3d} (0, -2, -2); \]
\[ \text{glTexCoord2d}(1,1); \]
\[ \text{glVertex3d} (0, 2, 2); \]

\[ \text{glTexCoord2d}(0,0); \]
\[ \text{glVertex3d} (0, -2, -2); \]
Fractional Texture Coordinates

texture image

(0,0) (0,.5) (0,0)
(0,1) (1,1) (.25,.5) (.25,0)
(1,0)
Texture Lookup: Tiling and Clamping

• what if \( s \) or \( t \) is outside the interval \([0\ldots1]\)?
• multiple choices
  • use fractional part of texture coordinates
    • cyclic repetition of texture to tile whole surface
      \[
      \text{glTexParameteri}(\ldots, \text{GL\_TEXTURE\_WRAP\_S, GL\_REPEAT, GL\_TEXTURE\_WRAP\_T, GL\_REPEAT}, \ldots )
      \]
  • clamp every component to range \([0\ldots1]\)
    • re-use color values from texture image border
      \[
      \text{glTexParameteri}(\ldots, \text{GL\_TEXTURE\_WRAP\_S, GL\_CLAMP, GL\_TEXTURE\_WRAP\_T, GL\_CLAMP}, \ldots )
      \]
glTexCoord2d(1, 1);
glVertex3d (x, y, z);

Tiled Texture Map

(0,0) + (1,0) = (1,1)

(0,0) + (0,1) = (0,1)

Texture
Object
Mapped Texture

(0,0) + (4,0) = (4,4)

(0,0) + (0,4) = (0,4)

Mapped Texture

glTexCoord2d(4, 4);
glVertex3d (x, y, z);
Demo

- Nate Robbins tutors
  - texture
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();
    glRotate();
    ...
    ```
  • more flexible than changing (s,t) coordinates

• [demo]
Texture Functions

• once 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 relit
  • use as surface color, modulate alpha: GL_DECAL
    • like replace, but supports texture transparency
  • blend surface color with another: GL_BLEND
    • new value controls which of 2 colors to use
    • indirection, new value not used directly for coloring

• specify with glTexEnvi(GL_TEXTURE_ENV,
  GL_TEXTURE_ENV_MODE, <mode>)

• [demo]
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
\((0.9, 0.8, 0.7)\)

Object color
\((0.5, 0.5, 0.5)\)
Final color
\((0.45, 0.4, 0.35)\)
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
    • `glTexParameteriv(GL_TEXTURE_2D, …, …)` to specify parameters for use when applying the texture
    • `glTexImage2D(GL_TEXTURE_2D, …)` to specify the texture data (the image itself)
• enable texturing: `glEnable(GL_TEXTURE_2D)`
• state how the texture will be used:
  • `glTexEnvf(…)`
• specify texture coordinates for the polygon:
  • use `glTexCoord2f(s, t)` before each vertex:
    • `glTexCoord2f(0, 0); glVertex3f(x, y, z);`
Low-Level Details

• large range of functions for controlling layout of texture data
  • state how the data in your image is arranged
  • e.g.: `glPixelStorei(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 be square and size a power of 2
  • common sizes are 32x32, 64x64, 256x256
  • smaller uses less memory, and there is a finite amount of texture memory on graphics cards

• ok to use texture template sample code for project 4
  • http://nehe.gamedev.net/data/lessons/lesson.asp?lesson=09
Texture Mapping

• texture coordinates
  • specified at vertices
    ```
    glTexCoord2f(s,t);
    glVertex3f(x,y,z);
    ```
  • interpolated across triangle (like R,G,B,Z)
    • …well not quite!
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
  - $s_0, s_1, s_2$:
    - texture coordinates of vertices
  - $w_0, w_1, w_2$:
    - homogeneous coordinates of vertices

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

(image courtesy of Kiriakos Kutulakos, U Rochester)
Reconstruction

• how to deal with:
  • pixels that are much larger than texels?
    • apply filtering, “averaging”
  • pixels that are much smaller than texels?
    • interpolate
MIPmapping

use “image pyramid” to precompute averaged versions of the texture

store whole pyramid in single block of memory

Without MIP-mapping

With MIP-mapping
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
- `gluBuild2DMipmaps`
  - automatically constructs a family of textures from original texture size down to 1x1

![without](image1.png) ![with](image2.png)
MIPmap storage

• only 1/3 more space required
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

$B(u)$
A bump map
Bump Mapping

$O'(u)$
Lengthening or shortening $O(u)$ using $B(u)$

$N'(u)$
The vectors to the 'new' surface
Embossing

- at transitions
  - rotate point’s surface normal by $\theta$ or $-\theta$
Displacement Mapping

• bump mapping gets silhouettes wrong
  • shadows wrong too
• change surface geometry instead
  • only recently available with realtime graphics
  • need to subdivide surface