University of British Columbia
CPSC 314 Computer Graphics
Jan-Apr 2005

Tamara Munzner

Textures II

Week 8, Wed Mar 2

http://www.ugrad.cs.ubc.ca/~cs314/Vjan2005
Correction & Review: Surface Texture

- define texture pattern over \((s,t)\) domain
  - image – 2D array of “texels”
- assign \((s,t)\) coordinates to each point on object surface
Correction & Review: Example Texture Map

\( \text{glTexCoord2d}(0,0); \)
\( \text{glVertex3d}(-x, -y, -z); \)

\( \text{glTexCoord2d}(1,1); \)
\( \text{glVertex3d}(-x, y, z); \)
Correction & Review: Example Texture Map

```c
glTexCoord2d(4, 4);
glVertex3d (x, y, z);
```

```c
glTexCoord2d(1, 1);
glVertex3d (x, y, z);
```

Correction
& Review: Example Texture Map

```c
eglTexCoord2d(1, 1);
glVertex3d (x, y, z);
```

```c
eglTexCoord2d(4, 4);
glVertex3d (x, y, z);
```

Texture
+
Object
=
Mapped Texture

Texture
+
Object
=
Mapped Texture

(4,0)
(4,4)
(0,0)
(0,4)
(1,0)
(0,0)
(1,1)
(0,1)
Review: Texture

- action when s or t is outside [0…1] interval
  - tiling
  - clamping

- texture matrix stack
  
```c
glmMatrixMode( GL_TEXTURE );
```
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
Texture Pipeline

Compute object space location

Use projector function to find (s, t)

Use corresponder function to find texels

Apply value transform function (scale, trans, rot)

Modify value (color, normal,...)
Texture Pipeline

Object position → Parameter space → Image space

(x, y, z) → (s, t) → Texel color
(-2.3, 7.1, 17.7) → (0.32, 0.29) → (0.9, 0.8, 0.7)
Texture Mapping

(s_0, t_0)

(s_1, t_1)

(s_2, t_2)
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)
- 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

- there are a large range of functions for controlling the 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,…
  - you will be given texture template sample code for project 3
- 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
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
  - also problematic for color interpolation, etc.
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

with
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

$O(u)$

$B(u)$
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 mapped normals are inconsistent with actual geometry
  - silhouettes wrong
  - shadows wrong
- displacement mapping actually affects the surface geometry
Environment Mapping

- cheap way to achieve reflective effect
  - generate image of surrounding
  - map to object as texture
Environment Mapping

- used to model object that reflects surrounding textures to the eye
  - movie example: cyborg in Terminator 2
- different approaches
  - sphere, cube most popular
    - OpenGL support
      - GL_SPHERE_MAP, GL_CUBE_MAP
- others possible too
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
Cube Mapping

- direction of reflection vector $r$ 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
Blinn/Newell Latitude Mapping