News

• signup sheet for P3 grading
  • today/Wed/Fri signups in class
  • or send email to dingkai AT cs
    • by 48 hours after the due date or you'll lose marks

• again: 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)
Review: 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

```gl
glTexCoord2f(s, t);
glVertexf(x, y, z, w);
```
Review: Tiled Texture Map

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

```
.glTexCoord2d(4, 4);
.glVertex3d (x, y, z);
```
Review: Fractional Texture Coordinates

texture image

(0,0)  (1,0)

(0,1)  (1,1)

(0,.5)  (.25,.5)

(0,0)  (.25,0)
Review: Texture

- action when s or t is outside [0…1] interval
  - tiling
  - clamping
- functions
  - replace/decal
  - modulate
  - blend
- texture matrix stack
  ```c
  glMatrixMode( GL_TEXTURE );
  ```
Textures II
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
  - `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

- 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
Texture Mapping

- texture coordinates
  - specified at vertices
    
    ```
    glTexCoord2f(s,t);
    glVertexf(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
  - $s0$, $s1$, $s2$:
    - texture coordinates of vertices
  - $w0$, $w1$, $w2$:
    - 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
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
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

Original surface

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
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

Grace Cathedral Light Probe
©1999 Paul Debevec
http://www.debevec.org/Probes
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 3\textsuperscript{rd} 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 Texture

- define texture pattern over 3D domain - 3D space containing the object
  - texture function can be digitized or procedural
  - for each point on object compute texture from point location in space
- common for natural material/irregular textures (stone, wood, etc…)

![Examples of volumetric texture application](image)
Volumetric Bump Mapping

Marble

Bump
Volumetric Texture Principles

- 3D function $\rho(x,y,z)$
- texture space – 3D space that holds the texture (discrete or continuous)
- rendering: for each rendered point $P(x,y,z)$ compute $\rho(x,y,z)$
- volumetric texture mapping function/space transformed with objects