Texture Mapping

February 13, 2003
M. Ian Graham
Carnegie Mellon University

Administrativia

Countdown:
- About 1 week until Assignment 3 is due
- Assignment 2 handback, comments
- Questions on Assignment 3?
Itinerary

- Introduction to Texture Mapping
- Aliasing and How to Fight It
- Texture Mapping in OpenGL
- Applications of Texture Mapping

Motivation for Texture Mapping

- Phong illumination model coupled with a single color across a broad surface
  - Produces boring objects
  - Very limited
- Options to make things interesting:
  - No simple surfaces—use many tiny polygons
    - Expensive! Too much geometry.
  - Apply textures across the polygons
    - Less geometry, and the image looks almost as good!
Definitions

Texture—the appearance and feel of a surface
Texture—an image used to define the characteristics of a surface
Texture—a multidimensional image which is mapped to a multidimensional space.

Texture mapping sample

For more info on the computer artwork of Jeremy Bim see http://www.3drender.com/bim/productions.html
Basic Concept

“Slap an image on a model.”

How do we map a two-dimensional image to a surface in three dimensions?

Texture coordinates

- 2D coordinate (s,t) which maps to a location on the image (typically s and t are over [0,1])

Assign a texture coordinate to each vertex

- Coordinates are determined by some function which maps a texture location to a vertex on the model in three dimensions

Once a point on the surface of the model has been mapped to a value in the texture, change its RGB value (or something else!) accordingly

This is called *parametric texture mapping*

A single point in the texture is called a *texel*
Something else?

- The first known use of texture in graphics was the modulation of surface color values, (diffuse coefficients) by Catmull in 1974.
- A texture does not have to indicate color!
- Bump mapping was developed in 1978 by Blinn.
- Transparency maps in 1985 by Gardner.

What is a texture map?

- Practical: “A way to slap an image on a model.”
- Better: “A mapping from any function onto a surface in three dimensions.”
- Most general: “The mapping of any image into multidimensional space.”
Overview

\[(u, v)\]
Texture Space (2D)

\[(x, y)\]
Screen Space (2D)

Screen Space (3D) (z buffer)

Object Space (3D)

World Space (3D)

*parameterization*

*projection*

Visual Overview

![Visual Overview Diagram](image-url)
Hardware Notes

- Texture-mapping is supported in all modern graphics hardware since the introduction of the Voodoo 3Dfx—it’s therefore cheap and easy.
- Though the mapping is conceptualized in the order texture -> object -> screen, it is determined in reverse order in hardware, during scan conversion (“To which texel does this pixel map?”)

Linear Texture Mapping

- Do a direct mapping of a block of texture to a surface patch.
Cube Mapping

“Unwrap” cube and map texture over the cube

Cylinder Mapping

Wrap texture along outside of cylinder, not top and bottom
- This stops texture from being distorted
Two-part Mapping

- To simplify the problem of mapping from an image to an arbitrary model, use an object we already have a map for as an intermediary!
- Texture -> Intermediate object -> Final model
- Common intermediate objects:
  - Cylinder
  - Cube
  - Sphere

Intermediate Object to Model

- This step can be done in many ways
  - Normal from intermediate surface
  - Normal from object surface
  - Use center of object
Still tough!

- Mapping onto complicated objects is difficult
  - Even simple objects can be hard—spheres always distort the texture

Itinerary

- Introduction to Texture Mapping
- Aliasing and How to Fight It
- Texture Mapping in OpenGL
- Applications of Texture Mapping
What is aliasing?

- Interested in mapping from screen to texture coordinates

---

What is aliasing?

- An on-screen pixel does not always map neatly to a texel. Particularly severe problems in regular textures
The Beginnings of a Solution

Pre-calculate how the texture should look at various distances, then use the appropriate texture at each distance. This is called *mipmapping*. 
The Beginnings of a Solution

- Each mipmap (each image below) represents a level of depth (LOD).
- Powers of 2 make things much easier.

Problem: Clear divisions between different levels of depth!
Mipmapping alone is unsatisfactory.
Another Component: Filtering

- Take the average of multiple texels to obtain the final RGB value
- Typically used along with mipmapping
- **Bilinear filtering**
  - Average the four surrounding texels
  - Cheap, and eliminates some aliasing, but does not help with visible LOD divisions

(demonstration movies)

Another Component: Filtering

- **Trilinear filtering**
  - Interpolate between two LODs
  - Final RGB value is between the result of a bilinear filter at one LOD and a second bilinear filter at the next LOD
  - Eliminates “seams” between LODs
  - At least twice as expensive as bilinear filtering
Another Component: Filtering

**Anisotropic filtering**

- Basic filtering methods assume that a pixel on-screen maps to a square (isotropic) region of the texture.
- For surfaces tilted away from the viewer, this is not the case!

Image courtesy of nVidia

Another Component: Filtering

**Anisotropic filtering**

- A pixel may map to a rectangular or trapezoidal section of texels—shape filters accordingly and use either bilinear or trilinear filtering.
- Complicated, but produces very nice results.
Bilinear Filtering

Trilinear Filtering
Anisotropic Filtering

Side-by-Side Comparison

Isotropic Filter  Anisotropic Filter

nVidia
Itinerary

- Introduction to Texture Mapping
- Aliasing and How to Fight It
- Texture Mapping in OpenGL
- Applications of Texture Mapping

---

gTexImage2D

- glTexImage2D(GL_TEXTURE_2D, level, components, width, height, border, format, type, tarray)
- GL_TEXTURE_2D
  - Specify that it is a 2D texture
- Level
  - Used for specifying levels of detail for mipmaping (more on this later)
- Components
  - Generally is 0 which means GL_RGB
  - Represents components and resolution of components
- Width, Height
  - The size of the texture must be powers of 2
- Border
- Format, Type
  - Specify what the data is (GL_RGB, GL_RGBA, …)
  - Specify data type (GL_UNSIGNED_BYTE, GL_BYTE, …)
glTexCoord2f

```
GLfloat nx, ny;
GLubyte *pix;

for (i = 0; i < texture->nx; i++) {
    pix = texture->image vaiscu[i];
    for (j = 0; j < texture->ny; j++) {
        glTexCoord2f((GLfloat)i / texture->nx,
                     (GLfloat)j / texture->ny);
        glVertex3f(i, j, 0);
    }
}
```

Other Texture Parameters

- **glTexParameterf()**
  - Use this function to set how textures repeat
    - `glTexParameterf(GL_TEXTURE_WRAP_S, GL_REPEAT)`
    - `glTexParameterf(GL_TEXTURE_WRAP_S, GL_CLAMP)`
  - Which spot on texture to pick
    - `glTexParameterf(GL_TEXTURE_2D,
                      GL_TEXTURE_MAG_FILTER, GL_NEAREST)`
    - `glTexParameterf(GL_TEXTURE_2D,
                      GL_TEXTURE_MIN_FILTER, GL_NEAREST)`
Mipmapping in OpenGL

- gluBuild2DMipmaps(GL_TEXTURE_2D, components, width, height, format, type, data)
- This will generate all the mipmaps using gluScaleImage
- glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST_MIPMAP_NEAREST)

- glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, 128, 128, 0, GL_RGB, GL_UNSIGNED_BYTE, LOD1)
- glTexImage2D(GL_TEXTURE_2D, 1, GL_RGB, 64, 64, 0, GL_RGB, GL_UNSIGNED_BYTE, LOD2)

If you design the mipmaps yourself

- glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, 128, 128, 0, GL_RGB, GL_UNSIGNED_BYTE, LOD1)
- glTexImage2D(GL_TEXTURE_2D, 1, GL_RGB, 64, 64, 0, GL_RGB, GL_UNSIGNED_BYTE, LOD2)
Other Texturing Issues

- `glTexEnvi(GL_TEX_ENV, GL_TEX_ENV_MODE, GL_MODULATE)`
  - Will balance between shade color and texture color
- `glTexEnvi(GL_TEX_ENV, GL_TEX_ENV_MODE, GL_DECAL)`
  - Will replace shade color with texture color

- `glHint(GL_PERSPECTIVE_CORRECTION, GL_NICEST)`
  - OpenGL does linear interpolation of textures
  - Works fine for orthographic projections
  - Allows for OpenGL to correct textures for perspective projection
  - There is a performance hit

- Texture objects
  - Maintain texture in memory so that it will not have to be loaded constantly

OpenGL texturing code

This code assumes that it’s an RGB texture map
Itinerary

- Introduction to Texture Mapping
- Aliasing and How to Fight It
- Texture Mapping in OpenGL
- Applications of Texture Mapping

Non-2D Texture Mapping

- The domain of a texture mapping function may be any number of dimensions
  - 1D might be used to represent rock strata
  - 2D is used most often
  - 3D can be used to represent interesting physical phenomena
  - Animated textures are a cheap extra dimension—further dimensions are somewhat harder to conceptualize
3D Texture Mapping

- Almost the same as 2D texture mapping
  - Texture is a “block” which objects fit into
  - Texture coordinates are 3D coordinates which equal some value inside the texture block

RGB values *or*...

- Textures do not have to represent color values.
- Using texture information to modify other aspects of a model can yield much more realistic results
RGB values or...

- Specularity (patches of shininess)
- Transparency (patches of clearness)
- Normal vector changes (bump maps)
- Reflected light (environment maps)
- Shadows
- Changes in surface height (displacement maps)

Bump Mapping

- How do you make a surface look rough?
  - Option 1: model the surface with many small polygons
  - Option 2: perturb the normal vectors before the shading calculation
    - Fakes small displacements above or below the true surface
    - The surface doesn’t actually change, but shading makes it look like there are irregularities!
    - A texture stores information about the “fake” height of the surface
    - For the math behind it all look at Angel 7.8
Bump Mapping

- We can perturb the normal vector without having to make any actual change to the shape.
- This illusion can be seen through—how?

![Original model (5M)](image1) ![Simplified (500)](image2) ![Simple model with bump map](image3)

Environment Mapping

- Allows for world to be reflected on an object without modeling the physics
- Map the world surrounding an object onto a cube
- Project that cube onto the object
- During the shading calculation:
  - Bounce a ray from the viewer off the object (at point $P$)
  - Intersect the ray with the environment map (the cube), at point $E$
  - Get the environment map's color at $E$ and illuminate $P$ as if there were a light source at position $E$
  - Produces an image of the environment reflected on shiny surfaces
Light Mapping

*Quake* uses light maps in addition to texture maps. Texture maps are used to add detail to surfaces, and light maps are used to store pre-computed illumination. The two are multiplied together at run-time, and cached for efficiency.

![Radiance Texture Map Only](image1.png)

![Radiance Texture + Light Map](image2.png)

Summary

- Introduction to Texture Mapping
- Aliasing and How to Fight It
- Texture Mapping in OpenGL
- Applications of Texture Mapping
Acknowledgements/Resources

Frank Pfenning and Shayan Sharkar  
(last year’s instance of this course)

Paul Heckbert  
- http://www.cs.cmu.edu/~ph

UNC (filter demonstration movies)  