Texture Mapping

COMP 575/770
Spring 2013
Textures

Goal: Add visual detail to 3D objects
Textures

- **Option 1:** Model everything with lots of little triangles
  - Way too much work (for computers and for people)

- **Option 2:** Paint an image on the model
  - The image is called a *texture*
Texture Mapping Overview

Map each point on the surface to a point in the texture image.
Outline

- Introduction
- Surface Parameterization
- Texture Mapping
- Texture Filtering
- Variations
Surface Parameterization

- How to assign \((u,v)\) coordinates to a surface?
- Trivial for rectangles
- Easy for spheres too (just use longitude, latitude)
Surface Parameterization

- Easy for other parametric surfaces
  - Surfaces defined by functions of two variables
Surface Parameterization

- Not so easy for bunnies
  - Or other arbitrary triangle meshes

- Typically performed by 3D modeling software
Surface Parameterization

- Option 1: Project to parametric surfaces
Surface Parameterization

- Option 2: Unwrap the model

[Image: Diagram showing the process of unwrapping a model.]

[Reference: Piponi 2000]
Surface Parameterization

- Option 3: Make an atlas

charts  atlas  surface

[Sander 2001]
Outline

- Introduction
- Surface Parameterization
- Texture Mapping
- Texture Filtering
- Variations
Texture Mapping

- After parameterization, we get \((u,v)\) coordinates for each vertex

- Use interpolation for \((u,v)\) values in the interior of triangles

- Essentially, \((u,v)\) texture coordinates are another vertex attribute
Texture Mapping

- Let’s use this method to apply a checkerboard pattern to a square, viewed at an angle

- This is what we want:
Texture Mapping

- This is what we get:
Perspective–Correct Mapping

- Fundamental problem with interpolation in screen-space
  - Uniform screen-space steps ≠ uniform world-space steps
Perspective–Correct Mapping

- Perspective foreshortening doesn’t get applied to texture coordinates with linear interpolation

- Instead of interpolating \((u,v)\), interpolate \((u/z,v/z)\)
  - Or, since \(w \propto 1/z\), interpolate \((uw, vw)\)
  - Perform divisions: \(u = uw/w, v = vw/w\) for each fragment

- This gives perspective–correct texture mapping
Outline

- Introduction
- Surface Parameterization
- Texture Mapping
- Texture Filtering
- Variations
Texture Filtering

- Texture image is, after all, a set of discrete samples (called **texels**)

- \((u,v)\) coordinates at a surface point rarely land exactly on a sample

- Need to use reconstruction filters to figure out the texture values between samples
Texture Filtering

Nearest-neighbor filtering
Texture Filtering

Bilinear filtering
Texture Filtering

- What if the area “covered” by a pixel contains multiple texels?

- Need a way to combine multiple texels to a single value

- Can pre-filter images using
  - MIP mapping
  - Summed area tables
MIP Mapping

- MIP = *multim im parvo* ("many things in one place")

- Keep a bunch of downsampled images at different sizes (called **levels**).
MIP Mapping

- Choose the level based on depth

- Alternately, interpolate between two levels (trilinear filtering)

Without MIP mapping

With MIP mapping
Summed Area Tables

- For each texel, store the sum of texel values below it and to its right

- Sum of values within any rectangle = difference of two texel values
Summed Area Tables

- Can compute more accurate averages
- MIP mapping is simpler, and supported in GPUs
Outline

- Introduction
- Surface Parameterization
- Texture Mapping
- Texture Filtering
- Variations
Texture Mapping Variations

- So far, we’ve used textures to specify $k_a, k_d, k_s$

- But there are many more uses!
  - For example, can store $k_s, p$ in a separate specular map
Bump Mapping

- Intensity = offset from surface
  - Used to derive varying surface normals
Normal Mapping

- Explicitly specify surface normals

original mesh
4M triangles

simplified mesh
500 triangles

simplified mesh
and normal mapping
500 triangles

[Paolo Cignoni]
Displacement Mapping

- Like bump mapping, but geometry shader used to actually displace the surface
Light Mapping

- Texture stores intensity/color of light incident at the surface
  - Precomputed using *global illumination* techniques

![Without light map](image1.png)  ![With light map](image2.png)
Environment Mapping

- Texture is an image of a complicated environment, sampled in all directions around a point.

Images from *Illumination and Reflection Maps: Simulated Objects in Simulated and Real Environments*

Gene Miller and C. Robert Hoffman

SIGGRAPH 1984 “Advanced Computer Graphics Animation” Course Notes