Questions from Last Time?

- Hall Shading Model
- Shadows
- Reflections
- Refractions
Slide Credits

- Jonathan Cohen - Johns Hopkins University
- Shayan Sarkar - Carnegie Mellon University
- Leonard McMillan, Jovan Popovic, Fredo Durand - MIT
- Steve Marschner - Cornell University
- David Luebke - University of Virginia
- Pat Hanrahan - Stanford University
2D Texture Mapping
The Quest for Visual Realism

At what point do things start looking real?

For more info on the computer artwork of Jeremy Birn see [http://www.3drender.com/jbirm/productions.html](http://www.3drender.com/jbirm/productions.html)
Photo-textures

For each triangle in the model establish a corresponding region in the phototexture.

The concept is very simple!

During rasterization interpolate the coordinate indices into the texture map.
Texturing

Allows higher-frequency color variation

- Not just interpolated from vertex colors

May be 2D (surface-based) or 3D (volume-based)

- (or even 4D for light fields -- that’s a different lecture)

May be strictly image-based or procedural

- Today we’ll talk about simple image-based
2D Texture Mapping

Requires surface parameterization

- Mapping from 3D surface to 2D parametric domain

Colors defined in 2D parameter space

Parameterization (texture coordinates) used to determine material color at point on surface
2D Texture Diagram

\[ F(x, y, z) = (s, t) \]
Texture Coordinate sample
2D Texture Example

+ =
Adding Texture Mapping to Illumination

Texture mapping can be used to alter some or all of the constants in the illumination equation. We can simply use the texture as the final color for the pixel, or we can just use it as diffuse color, or we can use the texture to alter the normal, or... the possibilities are endless!

\[ I_{total} = k_a I_{ambient} + \sum_{i=1}^{\text{lights}} I_i \left( k_d (\hat{N} \cdot \hat{L}) + k_s (\hat{V} \cdot \hat{R})^{n_{shiny}} \right) \]

Phong's Illumination Model

Constant Diffuse Color

Diffuse Texture Color

Texture used as Label

Texture used as Diffuse Color
Image-based Texture Mapping (2D)

2D texel array (image) determines colors in texture domain

Given texture coordinates on surface, look up color in image

Lookup may be return nearest texel (point sampled) or bilinear interpolation of 4 surrounding texels
Acquiring Texture Images

Photograph

- flat surface
- even lighting (no specularity)

3D Rendering

Procedural synthesis

- Sample a procedural texture
Canonical Parameterizations

Three common primitives:

- Plane
- Cylinder
- Sphere
Plane Parameterization

Suppose we have a plane with origin O and non-collinear axes, i and j

• \((x,y,z) = (O_x + s_i + t_j, O_y + s_i + t_j, O_z + s_i + t_j)\)
• \((u,v) = (s,t)\)
Suppose we have a circular cylinder of height \( h \) about z-axis (with base at \( z=0 \))

- \( (x,y,z) = (rcos\theta, rsin\theta, z) \)
- \( (u,v) = (\theta/2\pi, z/h) \)

Or we can choose to cover only a portion of the cylinder:

- \( (u,v) = (a(\theta-\theta_0)/2\pi, b(z-z_0)/h) \)
Sphere Parameterization

We can similarly parameterize the sphere:

- \((x,y,z) = (r\cos\theta\sin\phi, r\sin\theta\sin\phi, r\cos\phi)\)
- \((u,v) = (\theta/2\pi, \phi/\pi)\)

Note: parameterization degenerate at poles

- “you can’t comb the hair on a sphere”

Cover portion of sphere with texture:

- \((u,v) = (a^*(\theta-\theta_0)/2\pi, b^*(\phi-\phi_0)/\pi)\)
Examples of coordinate functions

- A parametric surface (e.g. spline patch)
  - surface parameterization gives mapping function directly
    (well, the inverse of the parameterization)
Examples of coordinate functions

• For a sphere: latitude-longitude coordinates
  – φ maps point to its latitude and longitude
Difficulties with texture mapping

• Another difficulty is how to map a 2D image onto an arbitrary 3D model
  – Ex: If you wanted to map onto a sphere, you can’t do it without distorting the texture
Atlas Approaches

Break complex surface into patches

Parameterize / texture each patch

- Parameterizations optimized to minimize distortions

Atlas describes mapping between texture domains and surface domain
Atlas Example


Johns Hopkins Department of Computer Science
Course 600.456: Rendering Techniques, Professor: Jonathan Cohen
Two-stage Mapping

1. Map texture onto canonical primitive (the intermediate surface)

2. Map intermediate surface to arbitrary object
   - Position objects with respect to each other
   - Project along normal direction (of either one)
Examples of coordinate functions

- For non-parametric surfaces it is trickier
  - directly use world coordinates
  - need to project one out
Examples of coordinate functions

- For non-parametric surfaces it is trickier
  - directly use world coordinates
  - need to project one out
Examples of coordinate functions

- For non-parametric surfaces it is trickier
  - directly use world coordinates
  - use intermediate parametric object
Examples of coordinate functions

- For non-parametric surfaces it is trickier
  - directly use world coordinates
  - use intermediate parametric object
Examples of coordinate functions

- For non-parametric surfaces it is trickier
  - directly use world coordinates
  - use intermediate parametric object
Examples of coordinate functions

- For non-parametric surfaces it is trickier
  - directly use world coordinates
  - use intermediate parametric object
Examples of coordinate functions

- For non-parametric surfaces it is trickier
  - directly use world coordinates
  - use intermediate parametric object
Examples of coordinate functions

- For non-parametric surfaces it is trickier
  - directly use world coordinates
  - use intermediate parametric object
Difficulties with texture mapping

- Interested in mapping from screen to texture coordinates
Difficulties with texture mapping

- Aliasing issues, it is especially apparent with regular textures
Texture Sampling

Sampling Approaches

Point Sampling

- Pick closest texel
- (Replication/pixel zoom for upsampling)

Interpolation

- Blend closest texels

Area Sampling

- Blend all covered texels
Bilinear Interpolation

\[ p = (p_s, p_t) \]

\[ p' = \left( \frac{p_s - a_s}{b_s - a_s}, \frac{p_t - a_t}{c_t - a_t} \right) \]

\[ p_{\text{color}} = \text{lerp}(\text{lerp}(a_{\text{color}}, b_{\text{color}}, p_s'), \text{lerp}(c_{\text{color}}, d_{\text{color}}, p_s'), p_t') \]

\[ \text{lerp}(k_1, k_2, t) = (1-t)k_1 + t k_2 \]
Texture Area Sampling

If frequency of texture content is higher than sampling rate, may want better filtering

Pixel-sized area on surface covers some area in texture domain

- Curvilinear quadrilateral or ellipse

Perform weighted average of texels covered by pixel-sized piece of surface
Antialiasing and Texture Mapping

- Texture mapping is uniquely harder
  - Coherent textures present pathological artifacts
  - Correct filter shape changes

- Texture mapping is uniquely easier
  - Textures are known ahead of time
  - They can thus be prefiltered
Antialiasing and Texture Mapping

More on texture problems

- Coherent texture frequencies become infinitely high with increasing distance
  - Ex: checkerboard receding to horizon
- Unfiltered textures lead to compression
  - Ex: pixel covers entire checkerboard
  - Unfiltered mapping: pixel is black or white
  - Moving checkerboard ➔ flashing pixels
Problem: a square pixel on screen becomes a curvilinear quadrilateral in texture map (see W&W, p 140)

The coverage and area of this shape change as a function of the mapping

Most texture antialiasing algorithms approximate this shape somehow
Antialiasing and Texture Mapping

- **Mip-mapping**
  - MIP = *Multim in Parvo* (many things in a small place)
  - Ignores shape change of inverse pixel
  - But allows *size* to vary

- Idea: store texture as a pyramid of progressively lower-resolution images, filtered down from original
Antialiasing: Mip Mapping

- Distant textures use higher levels of the mipmap
- Thus, the texture map is prefiltered
- Thus, reduced aliasing!
Antialiasing: Mip Mapping

- **Which** level of mip-map to use?
  - Think of mip-map as 3-D pyramid
  - Index into mip-map with 3 coordinates: $u, v, d$ (depth)

- **Q: What does $d$ correspond to in the mip-map?**
- **A: size of the filter**
Antialiasing: Mip Mapping

- The size of the filter (i.e., \( d \) in the mip-map) depends on the pixel coverage area in the texture map
  - In general, treat \( d \) as a continuous value
  - Blend between nearest mip-map level using tri-linear interpolation
Antialiasing: Mip Mapping

- Q: What’s wrong with the mip-map approach to prefiltering texture?
- A: Assumes pixel maps to square in texture space
- More sophisticated inverse pixel filters (see F&vD p 828):
  - Summed area tables
  - Elliptical weighted average filtering
Mip-map Filtering Methods

Compute d, the parameter along level space

Sample texture

Option 1: Point sample nearest level

Option 2: Point sample each adjacent level, then linearly interpolate between them

Option 3: Choose nearest level, then bilinearly interpolate within that level

Option 4: Trilinearly interpolate between the 8 samples of two adjacent mip-map levels (2 bilinear interps + 1 linear)
Aliasing in texture maps

- Aliasing is a big problem with textures
  - they look really bad if you point sample

- ...and all those artifacts will crawl around in animation
Sampling with a MIP map

(b) Trilinear interpolation on a pyramid.

[Heckbert, 86]
Limitations of Mip-Mapping

Assumes circular footprint of pixel in texture domain

- produces only isotropic filtering
- will either over-filter or under-filter in some regions (blurry or jaggy)
Efficient Anisotropic Filtering

Use multiple mip-map lookups to produce a non-symmetric filter

Video example: Feline

Figure 19: Trilinear paints blurry text.

Figure 21: “High-efficiency” Simple Feline paints smooth text.
Other Types of 2D Maps

Bump/normal maps

- Modify or define surface normals

Displacement maps

- Modify surface itself

Environment/reflection maps

- Define environment seen in specular reflections
History

Catmull/Williams 1974 - basic idea
Blinn and Newell 1976 - basic idea, reflection maps
Blinn 1978 - bump mapping
Williams 1978, Reeves et al. 1987 - shadow maps
Smith 1980, Heckbert 1983 - texture mapped polygons
Williams 1983 - mipmaps
Miller and Hoffman 1984 - illumination and reflectance
Perlin 1985, Peachey 1985 - solid textures
Greene 1986 - environment maps/world projections
Akeley 1993 - Reality Engine
Surface Color and Transparency

Tom Porter’s Bowling Pin

Source: RenderMan Companion, Pls. 12 & 13

CS348B Lecture 12

Pat Hanrahan, Spring 2002
Reflection Maps

Blinn and Newell, 1976

CS348B Lecture 12

Pat Hanrahan, Spring 2002

Slide 6 of 28
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 virtual light source at position E
  - You see an image of the environment reflected on shiny surfaces
Environment Maps

If, instead of using the ray from the surface point to the projected texture's center, we used the direction of the reflected ray to index a texture map. We can simulate reflections. This approach is not completely accurate. It assumes that all reflected rays begin from the same point.
Environment Map Approximation

Ray Traced
Environment Map

Self reflections are missing in the environment map

CS348B Lecture 12
Pat Hanrahan, Spring 2002
What's the Best Chart?

Box Map

Latitude Map

GL Map
Fisheye Lens

Pair of 180 degree fisheye
Photo by K. Turkowski

CS348B Lecture 12
Pat Hanrahan, Spring 2002
Cylindrical Panoramas

QuickTime VR

Mars Pathfinder

Memorial Church (Ken Turkowski)

CS348B Lecture 12

Pat Hanrahan, Spring 2002
Cubical Environment Map

- Easy to produce with rendering system
- Possible to produce from photographs
- "Uniform" resolution
- Simple texture coordinates calculation

CS348B Lecture 12

Pat Hanrahan, Spring 2002
Reflection Mapping
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
    • the surface doesn’t actually change, but shading makes it look that way
    • bump map fakes small displacements above or below the true surface
    • can use texture-mapping for this
    • For the math behind it all look at Angel 9.4
Bump Mapping

Textures can be used to alter the surface normal of an object. This does not change the actual shape of the surface -- we are only shading it as if it were a different shape! This technique is called *bump mapping*. The texture map is treated as a single-valued height function. The value of the function is not actually used, just its partial derivatives. The partial derivatives tell how to alter the true surface normal at each point on the surface to make the object appear as if it were deformed by the height function.

Since the actual shape of the object does not change, the silhouette edge of the object will not change. Bump Mapping also assumes that the Illumination model is applied at every pixel (as in Phong Shading or ray tracing).
Bump mapping

If $P_u$ and $P_v$ are orthogonal and $N$ is normalized:

$$\vec{P} = [x(u, v), y(u, v), z(u, v)]^T$$

Initial point

$$\vec{N} = \vec{P}_u \times \vec{P}_v$$

Normal

$$\vec{P} + B(u, v)\vec{N}$$

Simulated elevated point after bump

$$\vec{N} \times \vec{N} + B_u \vec{P}_u + B_v \vec{P}_v$$

Variation of normal in $u$ direction

$$B_u = \frac{B(s, t + \square) - B(s, t)}{2\square}$$

$$B_v = \frac{B(s + \square, t) - B(s, t + \square)}{2\square}$$

Variation of normal in $v$ direction

Compute bump map partials by numerical differentiation
More Bump Map Examples

Cylinder w/Diffuse Texture Map

Bump Map

Cylinder w/Texture Map & Bump Map
One More Bump Map Example

Notice that the shadow boundaries remain unchanged.
Bump Mapping

• In CG, we can perturb the normal vector without having to make any actual change to the shape.
• What would be the problems with bump mapping?
Displacement Mapping

We use the texture map to actually move the surface point. This is called displacement mapping. How is this fundamentally different than bump mapping?

The geometry must be displaced before visibility is determined. Is this easily done in the graphics pipeline? In a ray-tracer?
Stochastic Microgeometry for Displacement Mapping

- Reconstruct a stochastic distribution over a mesh
**Quake Light Maps**

Lower resolution

CS348B Lecture 12
Pat Hanrahan, Spring 2002

Slide 21 of 28
Shadow Maps

Shadow maps = depth maps from light source
Correct Shadow Maps

Step 1:
Create z-buffer of scene as seen from light source

Step 2.
Render scene as seen from the eye
For each light
Transform point into light coordinates

return (zl < zbuffer[xl][yl] ) ? 1 : 0
The Best of All Worlds

All these texture mapping modes are great!
The problem is, no one of them does everything well.
Suppose we allowed several textures to be applied to each primitive during rasterization.
Wrap Up

- Discuss status/problems/issues with this week’s programming assignment