Lecture 5:

Texture Mapping

Computer Graphics and Imaging
UC Berkeley CS184/284A
Texture Mapping Has Many Uses

Pattern on ball

Wood grain on floor
Describe Surface Material Properties

Proudfoot et al.
Describe Surface Material Properties

- Add details without raising geometric complexity
- Paste image onto geometry or define procedurally

Chan et al.
2D Texture Mapping of Images

Map 2D image onto object.
2D Texture Mapping of Images

Surface Color
2D Texture Mapping of Images

Surface Color

Surface Roughness
2D Texture Mapping of Images

Surface Color

Surface Roughness

Surface Geometry
Texture Coordinate Mappings
Think Chocolate Wrappers

Texture image
Three Spaces

Surface lives in 3D world space

Every 3D surface point also has a place where it goes in the 2D image and in the 2D texture.
Image Texture Applied to Surface

Rendering without texture

Rendering with texture

Texture image

Each triangle “copies” a piece of the texture image back to the surface.
Visualization of Texture Coordinates

Each surface point is assigned a texture coordinate \((u,v)\)

Visualization of texture coordinates

Triangle vertices in texture space

CS184/284A

Ren Ng, James O’Brien
Image Texture Applied to Surface

Each surface point is assigned a texture coordinate \((u,v)\)
Creating Good Surface Coordinates is Hard

Finding cuts

Texture atlases


CS184/284A  Ren Ng, James O’Brien
Sponza Palace Model

Textures applied to surfaces
Sponza Palace Model

Visualization of texture coordinates
Sponza Palace Model

Example textures used
Repeating Textures

Image Tiles allow repeating textures

• Images must be manipulated to allow tiling
• Often result in visible artifacts
  • There are methods to get around artifacts....
Interpolation Across Triangles: Barycentric Coordinates
Interpolation Across Triangles

Why do we want to interpolate?

- Specify values (e.g. texture coordinates) at vertices, and obtain smoothly varying values across surface

What do we want to interpolate?

- Texture coordinates, colors, normal vectors, ...

How do we interpolate?

- Barycentric coordinates
Barycentric Coordinates

A coordinate system for triangles \((\alpha, \beta, \gamma)\)

\[
(x, y) = \alpha A + \beta B + \gamma C
\]

\[
\alpha + \beta + \gamma = 1
\]

Inside the triangle if all three coordinates are non-negative
Barycentric Coordinates - Examples

\[(\alpha, \beta, \gamma) = (1, 0, 0)\]
\[(x, y) = \alpha A + \beta B + \gamma C\]
\[= A\]
Barycentric Coordinates - Examples

\[(\alpha, \beta, \gamma) = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})\]

\[(x, y) = \frac{1}{3} A + \frac{1}{3} B + \frac{1}{3} C\]
Linear Interpolation Across Triangle

Barycentric coords linearly interpolate values at vertices

\[ V = \alpha V_A + \beta V_B + \gamma V_C \]

\( V_A, V_B, V_C \) can be positions, texture coordinates, color, normal vectors, material attributes...
Barycentric Coordinates

Geometric viewpoint — proportional distances

Similar construction for other coordinates
Computing Barycentric Coordinates

Recall the line equation we derived in Lecture 2. \( L_{PQ}(x,y) \) is proportional to the distance from line \( PQ \).

\[
L_{PQ}(x, y) = -(x - x_P)(y_Q - y_P) + (y - y_P)(x_Q - x_P)
\]

\[
= (Q - P) \times \begin{pmatrix} x \\ y \end{pmatrix} - P
\]
Computing Barycentric Coordinates

Geometric viewpoint — proportional distances

\[ \alpha = \frac{L_{BC}(x, y)}{L_{BC}(x_A, y_A)} \]

Similar construction for other coordinates
Barycentric Coordinate Formulas

\[(x, y) = \alpha A + \beta B + \gamma C\]
\[\alpha + \beta + \gamma = 1\]

\[\alpha = \frac{-(x - x_B)(y_C - y_B) + (y - y_B)(x_C - x_B)}{-(x_A - x_B)(y_C - y_B) + (y_A - y_B)(x_C - x_B)}\]

\[\beta = \frac{-(x - x_C)(y_A - y_C) + (y - y_C)(x_A - x_C)}{-(x_B - x_C)(y_A - y_C) + (y_B - y_C)(x_A - x_C)}\]

\[\gamma = 1 - \alpha - \beta\]
Barycentric Coordinates

Alternative geometric viewpoint — proportional areas

\[
\alpha = \frac{A_A}{A_A + A_B + A_C}
\]
\[
\beta = \frac{A_B}{A_A + A_B + A_C}
\]
\[
\gamma = \frac{A_C}{A_A + A_B + A_C}
\]
Barycentric Coordinates

Linear Algebra View

\[(x, y) = \alpha A + \beta B + \gamma C\]

\[
\begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix} = \alpha \begin{bmatrix}
  A_x \\
  A_y \\
  1
\end{bmatrix} + \beta \begin{bmatrix}
  B_x \\
  B_y \\
  1
\end{bmatrix} + \gamma \begin{bmatrix}
  C_x \\
  C_y \\
  1
\end{bmatrix}
\]

\[
\begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix} = \begin{bmatrix}
  A_x & B_x & C_x \\
  A_y & B_y & C_y \\
  1 & 1 & 1
\end{bmatrix} \begin{bmatrix}
  \alpha \\
  \beta \\
  \gamma
\end{bmatrix}
\]

\[
\begin{bmatrix}
  \alpha \\
  \beta \\
  \gamma
\end{bmatrix} = M^{-1} \begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix}
\]
Barycentric Coordinates

Linear Algebra View

Consider SVD of

\[
\begin{bmatrix}
\alpha \\
\beta \\
\gamma
\end{bmatrix} = M_{\text{screen}}^{-1} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}
\]

\[
\begin{bmatrix}
\alpha \\
\beta \\
\gamma
\end{bmatrix} = M_{\text{texture}}^{-1} \begin{bmatrix} s \\ t \\ 1 \end{bmatrix}
\]

\[
\begin{bmatrix} s \\ t \\ 1 \end{bmatrix} = M_{\text{texture}} M_{\text{screen}}^{-1} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}
\]
Perspective Projection and Interpolation
Perspective Projection and Interpolation

Texture

Plane tilted down with perspective projection — What’s wrong?

Correct image
Perspective Projection and Interpolation

Texture

Barycentric interpolation of texture coordinates in screen-space

Correct image

Ren Ng, James O’Brien
Perspective Projection Creates Non Linearity

Linear interpolation in world coordinates yields nonlinear interpolation in screen coordinates!

Perspective interpolation supported in GPU
Perspective-Correct Interpolation

Texture

Affine screen-space interpolation

Perspective world-space interpolation
Depth Distortion

- Recall depth distortion from perspective
  - Interpolating in screen space different than in world
  - Ok, for shading (mostly)
  - Bad for texture
Depth Distortion

\[ S_1 = P_1 / h_1 \]

\[ S_2 = P_2 / h_2 \]

\[ S_3 = P_3 / h_3 \]

\[ S_4 = P_4 / h_4 \]
Depth Distortion

\[ S_1 = \frac{P_1}{h_1} \]
\[ S_2 = \frac{P_2}{h_2} \]
\[ S_3 = \frac{P_3}{h_3} \]
\[ S_4 = \frac{P_4}{h_4} \]

\[ X = \sum_i S_i b_i \]
\[ Q = \sum_i P_i a_i \]

We know the \( S_i, \ P_i, \) and \( b_i, \) but not the \( a_i. \)
Depth Distortion

\[ S_1 = \frac{P_1}{h_1} \]

\[ S_2 = \frac{P_2}{h_2} \]

\[ S_3 = \frac{P_3}{h_3} \]

\[ S_4 = \frac{P_4}{h_4} \]

\[ X = \sum_i S_i b_i \]

\[ X = \frac{Q}{h} = \left( \sum_i P_i a_i \right) / \left( \sum_j h_j a_j \right) \]

\[ Q = \sum_i P_i a_i \]
Depth Distortion

\[ S_1 = \frac{P_1}{h_1} \]

\[ S_2 = \frac{P_2}{h_2} \]

\[ S_3 = \frac{P_3}{h_3} \]

\[ S_4 = \frac{P_4}{h_4} \]

\[ X = \sum_i S_i b_i \]

\[ Q = \sum_i P_i a_i \]

\[ \sum_i S_i b_i = \left( \sum_i P_i a_i \right) / \left( \sum_j h_j a_j \right) \]

Ren Ng, James O’Brien
Depth Distortion

\[ S_1 = \frac{P_1}{h_1} \]
\[ S_2 = \frac{P_2}{h_2} \]
\[ S_3 = \frac{P_3}{h_3} \]
\[ S_4 = \frac{P_4}{h_4} \]

\[ X = \sum_i S_i b_i \]

\[ Q = \sum_i P_i a_i \]

\[ \sum_i P_i b_i / h_i = \left( \sum_i P_i a_i \right) / \left( \sum_j h_j a_j \right) \]
Depth Distortion

\[ S_1 = P_1 / h_1 \]
\[ S_2 = P_2 / h_2 \]
\[ S_3 = P_3 / h_3 \]
\[ S_4 = P_4 / h_4 \]
\[ X = \sum_i S_i b_i \]
\[ Q = \sum_i P_i a_i \]

\[ \sum_i P_i b_i / h_i = \left( \sum_i P_i a_i \right) / \left( \sum_j h_j a_j \right) \]

\[ b_i / h_i = a_i / \left( \sum_j h_j a_j \right) \quad \forall i \]

Independent of given vertex locations.

Note: Material on gray slides is optional for cs184.
Depth Distortion

\[ S_1 = P_1/h_1 \]
\[ S_2 = P_2/h_2 \]
\[ S_3 = P_3/h_3 \]
\[ S_4 = P_4/h_4 \]
\[ X = \sum_i S_i b_i \]
\[ Q = \sum_i P_i a_i \]

Linear equations in the \( a_i \):

\[
\left( \sum_j h_j a_j \right) b_i/h_i - a_i = 0 \quad \forall i
\]

\[
b_i/h_i = a_i/ \left( \sum_j h_j a_j \right) \quad \forall i
\]

Ren Ng, James O’Brien
Depth Distortion

\[ S_1 = \frac{P_1}{h_1} \]

\[ S_4 = \frac{P_4}{h_4} \]

\[ S_3 = \frac{P_3}{h_3} \]

\[ X = \sum_i S_i b_i \]

\[ P_1 \]

\[ P_4 \]

\[ P_2 \]

\[ P_3 \]

\[ Q = \sum_i P_i a_i \]

Linear equations in the \( a_i \).

Not invertible so add some extra constraints.

\[ \left( \sum_j h_j a_j \right) \frac{b_i}{h_i} - a_i = 0 \quad \forall i \]

\[ \sum_i a_i = \sum_i b_i = 1 \]

Ren Ng, James O’Brien
Depth Distortion

For a line: \[ a_1 = \frac{h_2 b_i}{(b_1 h_2 + h_1 b_2)} \]

For a triangle: \[ a_1 = \frac{h_2 h_3 b_1}{(h_2 h_3 b_1 + h_1 h_3 b_2 + h_1 h_2 b_3)} \]

Obvious Permutations for other coefficients.

Ren Ng, James O’Brien
Applying Textures is Sampling!
Simple Texture Mapping Operation

for each rasterized screen sample \((x,y)\):

\[(u,v) = \text{evaluate texcoord value at } (x,y)\]

\[
\text{float3 texcolor} = \text{texture.sample}(u,v);
\]

set sample’s color to texcolor;
Applying Textures is Sampling!

Actually “re-sampling”

Mathematically, to draw a texture sample at \((u,v)\):

- Start with discrete, sampled 2D function \(f(x,y)\). This function is only non-zero at sampled locations.

- Reconstruct a continuous 2D function, \(f_{\text{cont}}(x,y) = f(x,y) \ast k(x,y)\) by convolution with a reconstruction filter \(k(x,y)\).

- Draw the desired sample at \((u,v)\) from the continuous 2D signal by function evaluation: \(f_{\text{cont}}(u,v)\).

Signal processing concepts that should come to mind for you:

- Frequency spectrum, aliasing, Nyquist frequency, filtering, anti-aliasing...
Point Sampling Textures

High-res reference
Source image: 1280x1280 pixels

Point sampling
256x256 pixels
Texture Sampling Frequency
Sampling Rate on Screen vs Texture

Screen space (x,y)  Texture space (u,v)
1:1 mapping
Sampling Rate on Screen vs Texture

Screen space \((x,y)\)  
Texture space \((u,v)\)  
Magnified
Sampling Rate on Screen vs Texture

Screen space (x,y)  Texture space (u,v)

“Minified”
Texture Sampling Rate

The sampling frequency in screen space translates to a sampling frequency in texture space as determined by the mapping function.

In general the frequency varies across the scene depending on geometric transforms, viewing transforms, and the texture coordinate function.
Screen Pixel Area vs Texel Area

At optimal viewing size:

- 1:1 mapping between pixel sampling rate and texel sampling rate
- Dependent on texture resolution! e.g. 512x512

When larger (magnification)

- Multiple pixel samples per texel sample

When smaller (minification)

- One pixel sample per multiple texel samples
Screen Pixel Footprint in Texture

- Upsampling (Magnification)
- Downsampling (Minification)
Screen Pixel Footprint in Texture

Screen space

Texture space

NB: texture sampling pattern not rectilinear or isotropic
Estimating Footprint Area With Jacobian

\[ \psi(x) = \left( \frac{\partial u}{\partial x}, \frac{\partial v}{\partial x} \right) \]
\[ \left( \frac{\partial u}{\partial y}, \frac{\partial v}{\partial y} \right) \]

Screen space

Texture space
Texture Antialiasing
Will Supersampling Antialias?

High-res reference

512x supersampling

High quality, but costly
Texture Antialiasing

Will supersampling work?

• Yes, high quality, but costly
• When highly minified, many texels in pixel footprint

Goal: efficient texture antialiasing

• Want antialiasing with one/few texels per pixel
• How? Antialiasing = filtering before sampling!
Antialiasing: Signal, Sampling Rate, Nyquist Rate?

What signal are we sampling? What is the sampling frequency? What is the Nyquist frequency?
Texture Filtering
Texture Magnification
Bilinear Filtering

Want to sample texture value $f(u,v)$ at red point

Black points indicate texture sample locations
Texture Magnification - Easy Case

(Generally don’t want this — insufficient resolution)
This is image interpolation (will see kernel function)

Nearest  Bilinear  Bicubic
Bilinear Filtering

Take 4 nearest sample locations, with texture values as labeled.
Bilinear Filtering

And fractional offsets, \((s, t)\) as shown
Bilinear Filtering

\[
\begin{align*}
\text{Linear interpolation (1D)} \\
\text{lerp}(x, v_0, v_1) &= v_0 + x(v_1 - v_0)
\end{align*}
\]
Bilinear Filtering

Linear interpolation (1D)
\[ \text{lerp}(x, v_0, v_1) = v_0 + x(v_1 - v_0) \]

Two helper lerps (horizontal)
\[ u_0 = \text{lerp}(s, u_{00}, u_{10}) \]
\[ u_1 = \text{lerp}(s, u_{01}, u_{11}) \]
Bilinear Filtering

Linear interpolation (1D)
\[ \text{lerp}(x, v_0, v_1) = v_0 + x(v_1 - v_0) \]

Two helper lerps
\[ u_0 = \text{lerp}(s, u_{00}, u_{10}) \]
\[ u_1 = \text{lerp}(s, u_{01}, u_{11}) \]

Final vertical lerp, to get result:
\[ f(x, y) = \text{lerp}(t, u_0, u_1) \]
Bilinear Interpolation

\[ e = a + \alpha(b - a) \]
\[ f = d + \alpha(c - d) \]
\[ p = e + \beta(f - e) \]

Same answer either way.

\[ e' = b + \beta(c - b) \]
\[ f' = a + \beta(d - a) \]
\[ p = f' + \alpha(e' - f') \]

\[ \alpha = \frac{||e - a||}{||b - a||} \]
\[ \beta = \frac{||f' - a||}{||d - a||} \]
Reconstruction Filter Function

Test your understanding:

• What is the reconstruction filter $k(x,y)$ for bilinear interpolation? Nearest? What is a theoretically ideal filter? What are the pros/cons of each?

Nearest  Bilinear  Bicubic
Texture Minification
Screen Pixel Footprint in Texture

Upsampling (Magnification)  

Downsampling (Minification)
Texture Minification - Hard Case

Challenging

- Many texels can contribute to pixel footprint
- Shape of pixel footprint can be complex

Idea:

- Take texture image file, then low-pass filter it (i.e. filter out high frequencies) and downsample it (i.e. sample at a lower resolution) texture file. Do this recursively, and store successively lower resolution, each with successively lower maximum signal frequency.

- For each sample, use the texture file whose resolution approximates the screen sampling rate
Level 0 - Full Resolution Texture
Level 2 - Downsampling 4x4

Aliasing

Blurring
Level 4 - Downsampling 16x16
Mipmap (L. Williams 83)

Level 0 = 128x128
Level 1 = 64x64
Level 2 = 32x32
Level 3 = 16x16
Level 4 = 8x8
Level 5 = 4x4
Level 6 = 2x2
Level 7 = 1x1

“Mip” comes from the Latin “multum in parvo", meaning a multitude in a small space
MIP Map

Pre-compute filtered versions of the texture

- A given UV rate is some level of the texture
- Tri-linear filtering UV × map level
What is the storage overhead of a mipmap?
Computing Mipmap Level D

Estimate texture footprint using texture coordinates of neighboring screen samples
Computing Mipmap Level D

\[ D = \log_2 L \]

\[ L = \max \left( \sqrt{\left( \frac{du}{dx} \right)^2 + \left( \frac{dv}{dx} \right)^2}, \sqrt{\left( \frac{du}{dy} \right)^2 + \left( \frac{dv}{dy} \right)^2} \right) \]
Computing Mipmap Level $D$

$$D = \log_2 L$$

$$L = \max \left( \sqrt{\left( \frac{du}{dx} \right)^2 + \left( \frac{dv}{dx} \right)^2}, \sqrt{\left( \frac{du}{dy} \right)^2 + \left( \frac{dv}{dy} \right)^2} \right)$$
Mipmap Level, Direct Derivative

Linear Algebra View

\[
\begin{bmatrix}
  s \\
  t \\
  1
\end{bmatrix}
= M_{\text{texture}} M_{\text{screen}}^{-1}
\begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix}
\]

This function is linear so \( \frac{\partial s}{\partial x} \) would be const if no depth distortion.

\[a_1 = \frac{h_2 h_3 b_1}{(h_2 h_3 b_1 + h_1 h_3 b_2 + h_1 h_2 b_3)}\]

Accounting for depth distortion creates non-constant derivatives. It adds a nonlinear operation between matrices.
Visualization of Mipmap Level

D rounded to nearest integer level
Trilinear Filtering

Mipmap Level D

Bilinear result

Linear interpolation based on continuous D value

Mipmap Level D+1

Bilinear result
Visualization of Mipmap Level

Trilinear filtering: visualization of continuous D
Bilinear vs Trilinear Filtering Cost

Bilinear resampling:
- 4 texel reads
- 3 lerps (3 mul + 6 add)

Trilinear resampling:
- 8 texel reads
- 7 lerps (7 mul + 14 add)
Texture Filtering in Assignment

Image resampling choices

• Nearest
• Bilinear interpolation

Mipmap level resampling choices

• Always level 0
• Nearest D
• Linear interpolation

2 x 3 = 6 choices
Mipmap Limitations

Point sampling
Mipmap Limitations

Supersampling 512x
Mipmap Limitations

Overblur

Why?

Mipmap trilinear sampling
Anisotropic Filtering

Elliptical weighted average (EWA) filtering
Anisotropic Filtering

Ripmaps and summed area tables

- Can look up axis-aligned rectangular zones
- Diagonal footprints still a problem

EWA filtering

- Use multiple lookups
- Weighted average
- Mipmap hierarchy still helps

CS184/284A
Advanced Texturing Methods
Many, Many Uses for Texturing

In modern GPUs, texture = memory + filtering

• General method to bring data to fragment calculations

Many applications

• Environment lighting
• Store microgeometry
• Procedural textures
• Solid modeling
• Volume rendering
• …
Environment Map

A function from the sphere to colors, stored as a texture.

Lat / long texture map

Reflection vector indexes into texture map

[Blinn & Newell 1976]
Environment Map

With environment map

Without environment map

Reference: [Dror, Willsky, & Adelson 2004]
Spherical Environment Map

Hand with Reflecting Sphere. M. C. Escher, 1935. lithograph

Light Probes, Paul Debevec
Environmental Lighting

Environment map (left) used to render realistic lighting
Cube Map

A vector maps to cube point along that direction. The cube is textured with 6 square texture maps.
Environment Maps

Used in 1985 in movie Interface

• Lance Williams from the New York Institute of Technology

Note errors
Environment Maps

Used in 1985 in movie *Interface*

- Lance Williams from the New York Institute of Technology
Displacement Mapping

Texture stores perturbation to surface position
base surface

hand-painted displacement map (detail)

displaced surface
Bump Mapping

Texture stores perturbation to surface normal

[Blinn 1978]

CS184/284A

Ren Ng, James O'Brien
Bump Mapping

Where is the difference most noticeable?

Geometry

Bump mapping
Perturbs normals

Displacement mapping
Perturbs positions
3D Textures

Textures is a function of \((u,v,w)\). Solid modeling.
3D Procedural Noise + Solid Modeling

Perlin noise, Ken Perlin

CS184/284A

Ren Ng, James O’Brien
3D Textures and Volume Rendering
Provide Precomputed Shading

Simple shading

Ambient occlusion texture map

With ambient occlusion
Many uses of texturing

- Bring high-resolution data to fragment calculations
- Colors, normals, lighting on sphere, volumetric data, ...

How does texturing work?

- Texture coordinate parameterization
- Barycentric interpolation of coordinates
- Texture sampling pattern and frequency
- Mipmaps: texture filtering hierarchy, level calculation, trilinear interpolation
- Anisotropic sampling
Acknowledgments

This slide set contain contributions from:

- Kayvon Fatahalian
- David Forsyth
- Pat Hanrahan
- Angjoo Kanazawa
- Ren Ng
- James O’Brien
- Mark Pauly
Bonus Slides
Examples of Texture Coordinate Functions
Examples of Texture Coordinate Functions

A parametric surface (e.g. spline patch)

- Use parameter space coordinates as texture coordinates directly
Examples of Texture Coordinate Functions

Planar projection

Rosalee Wolfe
http://www.siggraph.org/education/materials/HyperGraph/mapping/r_wolfe/r_wolfe_mapping_1.htm
Examples of Texture Coordinate Functions

Spherical projection

Rosalee Wolfe
http://www.siggraph.org/education/materials/HyperGraph/mapping/r_wolfe/r_wolfe_mapping_1.htm
Examples of Texture Coordinate Functions

Cube map projection

Rosalee Wolfe
http://www.siggraph.org/education/materials/HyperGraph/mapping/r_wolfe/r_wolfe_mapping_1.htm
Examples of Texture Coordinate Functions

Function of object or world coordinates?

Rosalee Wolfe
http://www.siggraph.org/education/materials/HyperGraph/mapping/r_wolfe/r_wolfe_mapping_1.htm
Examples of Texture Coordinate Functions

Complex surfaces: project parts to parametric surfaces