Texture Mapping

Michael Kazhdan

(600.357 / 600.457)

HB Ch. 14.8, 14.9
FvDFH Ch. 16.3, 16.4.5, 16.6
Textures

We know how to go from this…

to this

J. Birn
Textures

But what about this... to this?

J. Birn
Textures

• How can we go about drawing surfaces with complex detail?
Textures

- How can we go about drawing surfaces with complex detail?

- We could tessellate in a complex manner and then associate the appropriate material properties to each vertex
Textures

• How can we go about drawing surfaces with complex detail?

• We could use a simple tessellation and use the location of surface points to look up the appropriate color values.
Textures

• Advantages:
  ◦ The 3D model remains simple
  ◦ It is easier to design/modify a texture image than it is to design/modify a surface in 3D.
Textures

Properties:

• Implemented as part of shading process
• Rely on maps being stored as 1D, 2D, or 3D images
• Subject to aliasing errors
Textures

General Implementation:

• To each vertex $v$, associate a set of coordinates $s^v = \{s_1^v, \ldots, s_n^v\}$ ($0 \leq s_i^v \leq 1, n \in \{1,2,3\}$)

• For each pixel $p$, compute the interpolated coordinates $s^p = \{s_1^p, \ldots, s_n^p\}$

• Use the color of the image at $s^p$ to define the color at $p$. 
Example: Brick Wall
Another Example: Brick Wall

\[ s^v = (0,1) \quad s^v = (1,1) \]

\[ s^v = (0,0) \quad s^v = (1,0) \]
2D Texture

- Coordinates described by variables $s$ and $t$ and range over interval $(0,1)$
- Texture elements are called texels
- Often 4 bytes (rgba) per texel
2D Texture

```
glBegin( GL_QUADS );

glTexCoord2f(0.0, 0.0);
glVertex3f(0.0, 0.0, 0.0);

glTexCoord2f(1.0, 0.0);
glVertex3f(1.0, 0.0, 0.0);

glTexCoord2f(1.0, 1.0);
glVertex3f(1.0, 1.0, 0.0);

glTexCoord2f(0.0, 1.0);
glVertex3f(0.0, 1.0, 0.0);

glEnd();
```
Texture Mapping

- Scan conversion:
  - Interpolate texture coordinates down/across scan lines
  - **Do perspective divide** at each pixel based on mapping from screen space to 3-space

\[
(s, t) = \alpha (s_1, t_1) + \beta (s_2, t_2) + \gamma (s_3, t_3)
\]
Texture Mapping

Linear interpolation of texture coordinates in screen space

Correct interpolation with perspective divide

Hill Figure 8.42
3D Rendering Pipeline (for direct illumination)

3D Primitives
  ▸ Modeling Transformation
    ▹ 3D Modeling Coordinates
  ▸ Camera Transformation
    ▹ 3D World Coordinates
  ▸ Lighting
    ▹ 3D World Coordinates
  ▸ Projection Transformation
    ▹ 3D Camera Coordinates
  ▸ Clipping
    ▹ 2D Screen Coordinates
  ▸ Viewport Transformation
    ▹ 2D Screen Coordinates
  ▸ Scan Conversion
    ▹ 2D Image Coordinates

Texture mapping

Image

3D Image Coordinates
Overview

• Texture mapping methods
  ◦ Parameterization
  ◦ Filtering

• Texture mapping applications
  ◦ Modulation textures
  ◦ Illumination mapping
  ◦ Bump mapping
  ◦ Environment mapping
  ◦ Shadow maps
Option: Unfold/Map Entire Surface

[Image of a 3D model of a frog with a grid overlay]

[Image of a 3D model of a flat, grid-like structure]

[Image of a more detailed, texture-rich flat structure]

[Image of a realistic 3D model of a frog]

[Piponi2000]
Option: Unfold/Map Entire Surface

- Tricky, because mapped surface may have severe distortions
- However, because texture is continuous, may be easier to think about

Gu et al. 2003
Option: Unfold/Map Entire Surface

• Tricky, because mapped surface may have severe distortions

• However, because texture is continuous, may be easier to think about

In general, it is impossible to parameterize a complex shape to a simple base domain so that both angles and areas are preserved
Option: Make an Atlas

charts  atlas  surface

[Sander2001]
Option: Make an Atlas

• Less distortion on each little piece of atlas

• Need to pack to patches to reduce wasted space in texture image

• May be more difficult to think about the relationships between the different pieces
Overview

• Texture mapping methods
  ◦ Parameterization
  ◦ Filtering

• Texture mapping applications
  ◦ Modulation textures
  ◦ Illumination mapping
  ◦ Bump mapping
  ◦ Environment mapping
  ◦ Shadow maps
Texture Filtering

Must sample texture to determine color at each pixel in image
Texture Filtering

Must sample texture to determine color at each pixel in image

- In general, the transformation from screen space to texture space does not preserve area
- Need to compute the average of the pixels in texture space to get the color for screen space
Texture Filtering

Must sample texture to determine color at each pixel in image

• In general, the transformation from screen space to texture space does not preserve area

• Need to compute the average of the pixels in texture space to get the color for screen space

If the distortion is very large, this will require a lot of texture look-ups/adds.
Texture Filtering

Size of filter depends on the projective deformation

• Can prefilter images for better performance
  ◦ Mip maps
  ◦ Summed area tables
Mip Maps

• Keep textures prefiltered at multiple resolutions
  ◦ For each pixel, use the mip-map closest level(s)
  ◦ Fast, easy for hardware
Mip Maps

• Keep textures prefiltered at multiple resolutions
  ○ For each pixel, use the mip-map closest level(s)
  ○ Fast, easy for hardware

Average over many pixels
Mip Maps

• Keep textures prefiltered at multiple resolutions
  ◦ For each pixel, use the mip-map closest level(s)
  ◦ Fast, easy for hardware
Mip Maps

- Keep textures prefiltered at multiple resolutions
  - For each pixel, use the mip-map closest level(s)
  - Fast, easy for hardware

Again: we’re trading aliasing for blurring!
Mip Maps

• Keep textures prefiltered at multiple resolutions
  ◦ For each pixel, use the mip-map closest level(s)
  ◦ Fast, easy for hardware

• This type of filtering is isotropic:
  ◦ It doesn’t take into account that there is more compression in the vertical direction than in the horizontal one

Again: we’re trading aliasing for blurring!
Summed-Area Tables

Key Idea:

• Approximate the summation/integration over an arbitrary region by a summation/integration over an axis-aligned rectangle:

\[
\text{Sum}([a, b] \times [c, d]) = \int_{a}^{b} \int_{c}^{d} f(x, y) \, dy \, dx
\]
Summed-Area Tables

Key Idea:

• Approximate the summation/integration over an arbitrary region by a summation/integration over an axis-aligned rectangle.

• Perform the integration quickly by pre-computing integrals and leveraging the formula:

\[ \int_{a}^{b} \int_{c}^{d} f(x, y) \, dy \, dx = \int_{0}^{b} \int_{0}^{d} f(x, y) \, dy \, dx - \int_{0}^{b} \int_{0}^{c} f(x, y) \, dy \, dx - \int_{0}^{a} \int_{0}^{d} f(x, y) \, dy \, dx + \int_{0}^{a} \int_{0}^{c} f(x, y) \, dy \, dx \]
Summed-Area Tables

Key Idea:

- Approximate the summation/integration over an arbitrary region by a summation/integration over an axis-aligned rectangle.

- Perform the integration quickly by pre-computing integrals and leveraging the formula

\[
\int_a^b \int_c^d f(x, y) \, dy \, dx = \int_0^b \int_0^d f(x, y) \, dy \, dx - \int_0^b \int_0^c f(x, y) \, dy \, dx - \int_0^a \int_0^d f(x, y) \, dy \, dx + \int_0^a \int_0^c f(x, y) \, dy \, dx
\]
Summed-Area Tables

Key Idea:

- Approximate the summation/integration over an arbitrary region by a summation/integration over an axis-aligned rectangle.

- Perform the integration quickly by pre-computing integrals and leveraging the formula

\[
\int_{a}^{b} \int_{c}^{d} f(x, y) \, dy \, dx = \int_{0}^{b} \int_{0}^{d} f(x, y) \, dy \, dx - \int_{0}^{b} \int_{0}^{c} f(x, y) \, dy \, dx - \int_{0}^{a} \int_{0}^{d} f(x, y) \, dy \, dx + \int_{0}^{a} \int_{0}^{c} f(x, y) \, dy \, dx
\]
Summed-Area Tables

Key Idea:

• Approximate the summation/integration over an arbitrary region by a summation/integration over an axis-aligned rectangle.

• Perform the integration quickly by pre-computing integrals and leveraging the formula

\[
\int_{a}^{b} \int_{c}^{d} f(x, y) \, dy \, dx = \int_{0}^{b} \int_{0}^{d} f(x, y) \, dy \, dx - \int_{0}^{b} \int_{0}^{c} f(x, y) \, dy \, dx - \int_{0}^{a} \int_{0}^{d} f(x, y) \, dy \, dx + \int_{0}^{a} \int_{0}^{c} f(x, y) \, dy \, dx
\]
Summed-Area Tables

Key Idea:

• Approximate the summation/integration over an arbitrary region by a summation/integration over an axis-aligned rectangle.

• Perform the integration quickly by pre-computing integrals and leveraging the formula

\[
\int_a^b \int_c^d f(x, y) \, dy \, dx = \int_0^b \int_0^d f(x, y) \, dy \, dx - \int_0^b \int_0^c f(x, y) \, dy \, dx + \int_0^a \int_0^d f(x, y) \, dy \, dx - \int_0^a \int_0^c f(x, y) \, dy \, dx
\]
Summed-Area Tables

Key Idea:

• Approximate the summation/integration over an arbitrary region by a summation/integration over an axis-aligned rectangle.

• Perform the integration quickly by pre-computing integrals and leveraging the formula

\[
\int_a^b \int_c^d f(x, y) \, dy \, dx = \int_a^b \int_0^d f(x, y) \, dy \, dx - \int_b^a \int_0^d f(x, y) \, dy \, dx - \int_b^a \int_0^c f(x, y) \, dy \, dx + \int_0^c \int_0^d f(x, y) \, dy \, dx
\]
Summed-Area Tables (Pre-Process)

- Precompute the values of the integral:
  \[ S(a, b) = \int_0^a \int_0^b f(x, y) \, dy \, dx \]

- Each texel is the sum of all texels below and to the left of it

<table>
<thead>
<tr>
<th>Original image</th>
<th>Summed area table</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 4 0</td>
<td></td>
</tr>
<tr>
<td>0 3 1 1</td>
<td></td>
</tr>
<tr>
<td>4 2 0 1</td>
<td></td>
</tr>
<tr>
<td>1 2 1 3</td>
<td></td>
</tr>
</tbody>
</table>
Summed-Area Tables (Pre-Process)

• Precompute the values of the integral:

\[ S(a, b) = \int_0^a \int_0^b f(x, y) \, dy \, dx \]

• Each texel is the sum of all texels below and to the left of it

<table>
<thead>
<tr>
<th>Original image</th>
<th>Summed area table</th>
</tr>
</thead>
<tbody>
<tr>
<td>1  2  4  0</td>
<td></td>
</tr>
<tr>
<td>0  3  1  1</td>
<td></td>
</tr>
<tr>
<td>4  2  0  1</td>
<td></td>
</tr>
<tr>
<td>1  2  1  3</td>
<td>1</td>
</tr>
</tbody>
</table>
Summed-Area Tables (Pre-Process)

- Precompute the values of the integral:
  \[ S(a, b) = \int_0^a \int_0^b f(x, y) \, dy \, dx \]

- Each texel is the sum of all texels below and to the left of it

<table>
<thead>
<tr>
<th>Original image</th>
<th>Summed area table</th>
</tr>
</thead>
<tbody>
<tr>
<td>1  2  4  0</td>
<td>1  3</td>
</tr>
<tr>
<td>0  3  1  1</td>
<td></td>
</tr>
<tr>
<td>4  2  0  1</td>
<td></td>
</tr>
<tr>
<td>1  2  1  3</td>
<td></td>
</tr>
</tbody>
</table>

Original image

Summed area table
Summed-Area Tables (Pre-Process)

- Precompute the values of the integral:
  \[ S(a, b) = \int_0^a \int_0^b f(x, y) \, dy \, dx \]

- Each texel is the sum of all texels below and to the left of it
Summed-Area Tables (Pre-Process)

- Precompute the values of the integral:
  \[ S(a, b) = \int_0^a \int_0^b f(x, y) \, dy \, dx \]

- Each texel is the sum of all texels below and to the left of it

Original image

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

Summed area table

|   | 1 | 3 | 4 | 7 |
Summed-Area Tables (Pre-Process)

- Precompute the values of the integral:
  \[ S(a, b) = \int_0^a \int_0^b f(x, y) \, dy \, dx \]

- Each texel is the sum of all texels below and to the left of it

<table>
<thead>
<tr>
<th>Original image</th>
<th>Summed area table</th>
</tr>
</thead>
<tbody>
<tr>
<td>1  2  4  0</td>
<td>5</td>
</tr>
<tr>
<td>0  3  1  1</td>
<td>1</td>
</tr>
<tr>
<td>4  2  0  1</td>
<td>3</td>
</tr>
<tr>
<td>1  2  1  3</td>
<td>7</td>
</tr>
</tbody>
</table>
Summed-Area Tables (Pre-Process)

- Precompute the values of the integral:
  \[ S(a, b) = \int_0^a \int_0^b f(x, y) \, dy \, dx \]

- Each texel is the sum of all texels below and to the left of it
Summed-Area Tables (Pre-Process)

• Precompute the values of the integral:

\[ S(a, b) = \int_0^a \int_0^b f(x, y) \, dy \, dx \]

• Each texel is the sum of all texels below and to the left of it

<table>
<thead>
<tr>
<th>Original image</th>
<th>Summed area table</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 4 0</td>
<td>6 15 21 26</td>
</tr>
<tr>
<td>0 3 1 1</td>
<td>5 12 14 19</td>
</tr>
<tr>
<td>4 2 0 1</td>
<td>5 9 10 14</td>
</tr>
<tr>
<td>1 2 1 3</td>
<td>1 3 4 7</td>
</tr>
</tbody>
</table>
Summed-Area Tables (Run-Time)

• Given a pixel on the screen that maps to a rectangle in texture space use the summed area table to compute the average:

\[
\text{Sum}([1,3] \times [2,3]) = S(3,3) - S(0,3) - S(3,1) + S(0,1) = 26 - 6 - 14 + 5 = 11
\]

\[
\text{Average}([1,3] \times [2,3]) = \frac{\text{Sum}([1,3] \times [2,3])}{\text{Area}([1,3] \times [2,3])} = \frac{11}{6}
\]
Overview

• Texture mapping methods
  ◦ Parameterization
  ◦ Filtering

• Texture mapping applications
  ◦ Modulation textures
  ◦ Illumination mapping
  ◦ Bump mapping
  ◦ Environment mapping
Modulation textures

Map texture values to scale factor

\[ I = T(s, t) \left( I_E + K_A I_{AL} + \sum_i \left( K_D \langle \vec{N}, \vec{L}_i \rangle I_i + K_S \langle \vec{V}, \vec{R}_i \rangle^n I_i \right) \right) \]
Illumination Mapping

Map texture values to any material parameter

Modulation

Diffuse

\[
I = I_E + K_A I_{AL} + \sum_i \left( T(s, t) \langle \hat{N}, \hat{L}_i \rangle I_i + K_S \langle \hat{V}, \hat{R}_i \rangle^n I_i \right)
\]
Illumination Mapping

Map texture values to any material parameter

Note that we need to evaluate the texture at each pixel but can still use the interpolated lighting values $\langle \vec{N}, \vec{L}_i \rangle$

$$I = I_E + K_A I_{AL} + \sum_i \left( T(s, t) \langle \vec{N}, \vec{L}_i \rangle I_i + K_S \langle \vec{V}, \vec{R}_i \rangle^n I_i \right)$$
Illumination Mapping

Map texture values to any material parameter

Modulation

Diffuse

Specular

\[ I = I_E + K_A I_{AL} + \sum_i \left( K_D \langle \hat{N}, \hat{L}_i \rangle I_i + T(s, t)(\hat{V}, \hat{R}_i)^n I_i \right) \]
Illumination Mapping

Map texture values to any material parameter

- Modulation
- Diffuse
- Specular

Again, we don’t need to re-compute most of the lighting calculation $\langle \hat{V}, \hat{R}_i \rangle^n$

\[ I = I_E + K_A I_{AL} + \sum_i \left( K_D \langle \hat{N}, \hat{L}_i \rangle I_i + T(s, t) \langle \hat{V}, \hat{R}_i \rangle^n I_i \right) \]
Bump Mapping

• Recall that many parts of our lighting calculation depend on surface normals

\[ I = I_E + K_A I_{AL} + \sum_i \left( K_D \langle N, L_i \rangle I_i + K_S \langle V, R_i \rangle^n I_i \right) \]
Bump Mapping

\[ n_0 \quad n_1 \]

P. Rheingans
Bump Mapping

Phong shading performs per-pixel lighting calculations with the interpolated normals, approximating a smoothly curved surface.
Bump Mapping

Phong shading performs per-pixel lighting calculations with the interpolated normals, approximating a smoothly curved surface.

With bump maps, we encode the normals in the texture.

P. Rheingans
Bump Mapping

Phong shading performs per-pixel lighting calculations with the interpolated normals, approximating a smoothly curved surface.

With bump maps, we encode the normals in the texture. This allows Phong shading to give the appearance of a bumpy surface.
Bump Mapping

H&B Figure 14.100
Bump Mapping

Note that bump mapping does not change object silhouette
Environment Mapping

- Generate a spherical/cubic map of the environment around the model.
- Texture values are dynamically reflected off surface patch.
Environment Mapping

• Generate a spherical/cubic map of the environment around the model.

• Texture values are dynamically reflected off surface patch

Set the texture coordinates based on the direction of the reflected view direction
Environment Mapping

- Generate a spherical/cubic map of the environment around the model.
- Texture values are dynamically reflected off surface patch

Set the texture coordinates based on the direction of the reflected view direction
Environment Mapping

• Generate a spherical/cubic map of the environment around the model.

• Texture values are dynamically reflected off surface patch

Set the texture coordinates based on the direction of the reflected view direction
Environment Mapping

• Generate a spherical/cubic map of the environment around the model.

• Texture values are dynamically reflected off surface patch

Set the texture coordinates based on the direction of the reflected view direction

Changing the position of the camera changes the texture coordinates.
Environment Mapping

• Generate a spherical/cubic map of the environment around the model.

• Texture values are dynamically reflected off surface patch

Set the texture coordinates based on the direction of the reflected view direction

Changing the position of the camera changes the texture coordinates.
Environment Mapping

• Generate a spherical/cubic map of the environment around the model.

• Texture values are dynamically reflected off surface patch

Set the texture coordinates based on the direction of the reflected view direction

Changing the position of the camera changes the texture coordinates.
Environment Mapping

Texture values are dynamically reflected off surface patch