Texture Mapping

Michael Kazhdan

(601.457/657)

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

We know how to go from this…

[J. Birn]
Textures

But what about this… to this?

[J. Birn]
Textures

• How do we draw surfaces with complex detail?
Textures

• How do we draw surfaces with complex detail?

**Direct:**
• Tessellate uniformly (finely) and then associate the appropriate material properties to each vertex.

Target Model

Complex Surface
Textures

- How do we draw surfaces with complex detail?

**Direct:**
- Or, tessellate adaptively and then associate the appropriate material properties to each vertex

![Target Model](image1)

![Complex Surface](image2)
Textures

• How do we draw surfaces with complex detail?

Indirect:
• Use a simple tessellation with an auxiliary *texture image*. Use the location of surface points to look up color values from the texture.
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 (2 dimensions)

Implementation:

• Associate a *texture coordinate* to each vertex \( v \): 
  \((s_v, t_v)\) with \((0 \leq s_v, t_v \leq 1)\)

• When rasterizing, *interpolate* to get the texture coordinate to at a pixel: 
  \((s_p, t_p)\)

• *Sample* the texture at \((s_p, t_p)\) to get the color at \( p \).
Example: Brick Wall
Example: Brick Wall

\[(s_v, t_v) = (0,1) \quad (s_v, t_v) = (1,1)\]

\[(s_v, t_v) = (0,0) \quad (s_v, t_v) = (1,0)\]
Textures (2 dimensions)

- Coordinates described by variables \( s \) and \( t \) and range over interval \((0,1)\).
- Texture elements are called *texels*.
- Often 4 bytes (rgba) per texel.
3D Rendering Pipeline

- **3D Primitives**
  - 3D Modeling Coordinates
- **Modeling Transformation**
- **Viewing Transformation**
  - 3D World Coordinates
- **Lighting**
  - 3D World Coordinates
- **Projection Transformation**
  - 3D Camera Coordinates
- **Clipping**
  - 2D Window Coordinates
- **Viewport Transformation**
  - 2D Window Coordinates
- **Scan Conversion**
  - 2D Viewport Coordinates
  - Image

- **Texture Mapping**
  - Perform the texture interpolation and look-up while rasterizing the pixels.

---

\[(s_1, t_1), (s_{12}, t_{12}), (s_{123}, t_{123}), (s_{23}, t_{23}), (s_3, t_3), (s_2, t_2)\]
Texture Mapping

Recall (Perspective Divide):

When performing scan-line rasterization and interpolating data from vertices, we need to compute the weights in 3D space.

\[
R = \frac{p_2}{p_1}
\]

\[
y = 0, y = 1
\]
Texture Mapping

Linear interpolation of texture coordinates in screen space

Correct interpolation of texture coordinates with perspective divide

Hill Figure 8.42
Texture Mapping

Linear interpolation of texture coordinates in screen space

Correct interpolation of texture coordinates with perspective divide

Hill Figure 8.42
Overview

• Texture mapping methods
  ◦ Parameterization
  ◦ Sampling

• Texture mapping applications
  ◦ Modulation textures
  ◦ Illumination mapping
  ◦ Bump mapping
  ◦ Environment mapping
  ◦ Shadow maps
Map to a 2D Domain (w/ Added Cuts)

- Introduce cuts to give the surface a disk topology
- Map the cut surface to the 2D plane
- Assign texture coordinates in the plane

✓ Good cut placement can reduce distortion
✗ Need to ensure cross-seam continuity
✗ Have to contend with distortion

[Piponi2000]
Texture Atlases

- Decompose the surface into multiple charts
- Map each chart to the 2D plane
- Assign texture coordinates in the plane

✓ Less distortion in the mapping
✗ Harder to ensure cross-seam continuity
✗ Need to pack the atlases into 2D

[Sander2001]
Overview

• Texture mapping methods
  ◦ Parameterization
  ◦ **Sampling**

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

Given pixel on a screen:

1. Determine the corresponding surface patch
2. Determine the corresponding texture patch

Angel Figure 9.4
Texture Filtering

Given pixel on a screen:

1. Determine the corresponding surface patch
2. Determine the corresponding texture patch

While the true shape of the texture patch may be hard to compute, we can obtain a linear approximation by looking at the Jacobian of the map from pixel-space to texture-spaces.
Texture Filtering

Given pixel on a screen:

1. Determine the corresponding surface patch
2. Determine the corresponding texture patch
3. Average texel values over the texture patch
Texture Filtering

- Size of texture patch depends on the deformation
  - Computation is linear in the size of the pixel footprint

- Can pre-filter images for better performance
  - MIP (Multum In Parvo) maps
  - Summed area tables
MIP Maps

• In a **pre-processing** step, compute a hierarchy of successively down-sampled texture images
MIP Maps

- **Pre Processing**: Compute a hierarchy of successively down-sampled texture images.
- **Run-time**: Sample the closest MIP map level(s)
  - Easy for hardware
  - Computation is constant in the size of the pixel footprint

Average over a few pixels
MIP Maps

- **Pre Processing**: Compute a hierarchy of successively down-sampled texture images
  - Storage is only $\frac{4}{3}$ the size of the input image
MIP Maps

- **Pre Processing**: Compute a hierarchy of successively down-sampled texture images
  - Storage is only 4/3 the size of the input image
- **Run-time**: Sample the closest MIP map level(s)
  - This type of filtering is isotropic:
    - Assumes identical compression along the vertical and horizontal directions

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:

\[
\text{Sum}([a, b] \times [c, d]) = \int_a^b \int_c^d f(x, y) \, dy \, dx
\]
Recall

Integration:

Given a function $f(x)$ and an interval $[a, c]$, the integral of $f$ over the interval is denoted:

$$\int_{a}^{c} f(x)\,dx$$
Recall

Integration:

Given a function $f(x)$ and an interval $[a, c]$, the integral of $f$ over the interval is denoted:

$$\int_{a}^{c} f(x) \, dx$$

For any point $b \in [a, c]$ in the interval, we have:

$$\int_{a}^{c} f(x) \, dx = \int_{a}^{b} f(x) \, dx + \int_{b}^{c} f(x) \, dx$$
Recall

**Integration:**

Given a function $f(x)$ and an interval $[a, c]$, the integral of $f$ over the interval is denoted:

$$\int_{a}^{c} f(x) \, dx$$

For any point $b \in [a, c]$ in the interval, we have:

$$\int_{a}^{c} f(x) \, dx = \int_{a}^{b} f(x) \, dx + \int_{b}^{c} f(x) \, dx$$

This is true even if $c$ is outside the interval since:

$$\int_{a}^{b} f(x) \, dx = -\int_{b}^{a} f(x) \, dx$$
Recall

In Particular:

We can replace the integral over an interval, with two variable end-points with the difference in integrals with one variable end-point:

$$\int_a^c f(x) \, dx = \int_0^c f(x) \, dx - \int_0^a f(x) \, dx$$

Replacing the 1D function $f(x)$ with the 1D function:

$$F(x) = \int_0^x f(s) \, ds$$

lets us evaluate integrals with a constant number of look-ups:

$$\int_a^b f(s) \, ds = F(b) - F(a)$$
Recall

Integration:

In 2D, we can write out the integral of the function $f(x, y)$ over the rectangle $[a, b] \times [c, d]$ as:

$$
\int_{a}^{b} \int_{c}^{d} f(x, y) \, dy \, dx
$$
Recall

Integration:

In 2D, we can write out the integral of the function $f(x, y)$ over the rectangle $[a, b] \times [c, d]$ as:

$$\int_a^b \int_c^d f(x, y) \, dy \, dx = \int_a^b \left( \int_0^d f(x, y) \, dy \right) \, dx - \int_0^c f(x, y) \, dy \, dx$$
Recall

Integration:

In 2D, we can write out the integral of the function \( f(x, y) \) over the rectangle \([a, b] \times [c, d]\) as:

\[
\int_a^b \int_c^d f(x, y) \, dy \, dx = \int_a^b \left( \int_0^d f(x, y) \, dy - \int_0^c f(x, y) \, dy \right) \, dx \\
= \int_0^b \left( \int_0^d f(x, y) \, dy - \int_0^c f(x, y) \, dy \right) \, dx - \int_0^a \left( \int_0^d f(x, y) \, dy - \int_0^c f(x, y) \, dy \right) \, dx
\]
Recall

Integration:

In 2D, we can write out the integral of the function $f(x, y)$ over the rectangle $[a, b] \times [c, d]$ as:

\[
\int_{a}^{b} \int_{c}^{d} f(x, y) \, dy \, dx = \int_{a}^{b} \left( \int_{0}^{d} f(x, y) \, dy \right) \, dx - \int_{0}^{b} \left( \int_{0}^{c} f(x, y) \, dy \right) \, dx
\]

\[
= \int_{0}^{b} \left( \int_{0}^{a} \int_{0}^{d} f(x, y) \, dy \, dx - \int_{0}^{a} \int_{0}^{c} f(x, y) \, dy \, dx \right) - \int_{0}^{a} \int_{0}^{d} f(x, y) \, dy \, dx + \int_{0}^{a} \int_{0}^{c} f(x, y) \, dy \, dx
\]
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:

Replacing the 2D function \( f(x, y) \) with the 2D function:

\[
F(x, y) = \int_0^x \int_0^y f(s, t) \, dt \, ds
\]

lets us us evaluate integrals with a constant number of look-ups:

\[
\int_a^b \int_c^d f(s, t) \, dt \, ds = F(b, d) - F(b, c) - F(a, d) + F(a, c)
\]
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

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

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

- Precompute the values of the integral:

\[ S(a, b) = \int_{0}^{a} \int_{0}^{b} f(x, y) \, dy \, dx \]

- Each summed-area table texel is the sum of all input texels below and to the left

<table>
<thead>
<tr>
<th>Input 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 summed-area table texel is the sum of all input texels below and to the left

<table>
<thead>
<tr>
<th>Input image</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Summed area table</th>
</tr>
</thead>
<tbody>
<tr>
<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 summed-area table texel is the sum of all input texels below and to the left

<table>
<thead>
<tr>
<th>Input 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>
<tr>
<td></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 summed-area table texel is the sum of all input texels below and to the left
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 summed-area table texel is the sum of all input texels below and to the left

<table>
<thead>
<tr>
<th>Input image</th>
<th>Summed area table</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 4 0</td>
<td>1 3 4 7</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 summed-area table texel is the sum of all input texels below and to the left
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 summed-area table texel is the sum of all input texels below and to the left

<table>
<thead>
<tr>
<th>Input 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>
<tr>
<td></td>
<td>5 9</td>
</tr>
<tr>
<td></td>
<td>1 3 4 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 summed-area table texel is the sum of all input texels below and to the left

<table>
<thead>
<tr>
<th>Input 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)

Example:

Compute the average in the rectangle $[1,3] \times [2,3]$. 

$\Rightarrow$ Compute the sum and divide by the area

\[
\begin{array}{cccc}
1 & 2 & 4 & 0 \\
0 & 3 & 1 & 1 \\
4 & 2 & 0 & 1 \\
1 & 2 & 1 & 3 \\
\end{array}
\]

\[
\begin{array}{cccc}
6 & 15 & 21 & 26 \\
5 & 12 & 14 & 19 \\
5 & 9 & 10 & 14 \\
1 & 3 & 4 & 7 \\
\end{array}
\]

Input image

Summed-area table
Summed-Area Tables (Run-Time)

Example:
Compute the average in the rectangle $[1,3] \times [2,3]$.

$\Rightarrow$ Compute the sum and divide by the area

$$\text{Sum}([1,3] \times [2,3]) = S(3,3)$$
Summed-Area Tables (Run-Time)

Example:

Compute the average in the rectangle $[1,3] \times [2,3]$.

$\Rightarrow$ Compute the sum and divide by the area

$\text{Sum}([1,3] \times [2,3]) = S(3,3) - S(0,3)$
Summed-Area Tables (Run-Time)

Example:

Compute the average in the rectangle $[1,3] \times [2,3]$.  
$\Rightarrow$ Compute the sum and divide by the area  
$\text{Sum}([1,3] \times [2,3]) = S(3,3) - S(0,3) - S(3,1)$
Summed-Area Tables (Run-Time)

Example:

Compute the average in the rectangle $[1,3] \times [2,3]$.  
$\Rightarrow$ Compute the sum and divide by the area 

$\text{Sum}([1,3] \times [2,3]) = S(3,3) - S(0,3) - S(3,1) + S(0,1)$

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

Input image

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>15</td>
<td>21</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>14</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>9</td>
<td>10</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>4</td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

Summed-area table
Summed-Area Tables (Run-Time)

Example:

Compute the average in the rectangle $[1,3] \times [2,3]$.

⇒ Compute the sum and divide by the area

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

\[
\text{Average}([1,3] \times [2,3]) = \frac{\text{Sum}([1,3] \times [2,3])}{\text{Area}([1,3] \times [2,3])} = \frac{11}{6}
\]
Summed-Area Tables (Run-Time)

- Precompute the values of the integral
  ✓ Constant time averaging, regardless of rectangle size
  ✗ If the input image has values in the range $[0, 255]$ (i.e. one byte per channel), the summed area table can have values in the range $[0, 255 \cdot \text{width} \cdot \text{height}]$

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

Input image

<table>
<thead>
<tr>
<th></th>
<th>6</th>
<th>15</th>
<th>21</th>
<th>26</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>12</td>
<td>14</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>9</td>
<td>10</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>4</td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

Summed-area table
Overview

• Texture mapping methods
  ◦ Parameterization
  ◦ Sampling

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

Map texture values to scale factor

Modulation

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

Map texture values to a material parameter

Modulation

Diffuse

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

Map texture values to a material parameter

Modulation

Diffuse

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

This requires the graphics card to separately store the diffuse component of the lighting at each vertex

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

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

$$I = I_E + K_A I_{AL} + \sum_i (K_D \langle \overrightarrow{N}, \overrightarrow{L}_i \rangle + K_S \langle \overrightarrow{V}, \overrightarrow{R}_i^n \rangle) I_i$$
Bump Mapping

**Phong** shading performs per-pixel lighting calculations with the interpolated normal

\[ \downarrow \]

approximates a smoothly curved surface

Bump maps encode the normals in the texture

\[ \downarrow \]

approximates a more complex undulating surface
Bump Mapping

H&B Figure 14.100
Bump Mapping

Note that bump mapping does not change object silhouette.
Environment Mapping

Simulate reflective materials through pre-computed texture maps representing the environment around a shape.
Environment Mapping

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

• Texture coordinates are computed dynamically through reflection
Environment Mapping

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

• Texture coordinates are computed **dynamically** through reflection

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 coordinates are computed **dynamically** through reflection

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 coordinates are computed **dynamically** through reflection

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 coordinates are computed **dynamically** through reflection

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

At the same triangle, changing the position of the camera changes the texture coordinates.
Environment Mapping

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

• Texture coordinates are computed **dynamically** through reflection.

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

At the same triangle, changing the position of the camera changes the texture coordinates.
Environment Mapping

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

• Texture coordinates are computed dynamically through reflection

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

At the same triangle, changing the position of the camera changes the texture coordinates.
Environment Mapping

Texture coordinates are computed dynamically through reflection of the view direction through the surface normal.
Shadow Mapping (Williams 1978)

Test if surface is visible to the light when computing the contribution to the lighting equation.

Shadow Mapping (Williams 1978)

Q: Is the point that is seen by the camera visible (i.e. not in shadow) to the light?

A: The point $p$ is visible if:

- The light can “see” the point $p$.
  
  ⇒ Rendering the scene from the light’s perspective, $p$’s $z$-coordinate is the value stored in the $z$-buffer.
Shadow Mapping (Williams 1978)

Test if surface is visible to the light when computing contribution to the lighting equation.

- Render the scene from the light’s perspective and read back the $z$-buffer/shadow map.
- For each pixel in the camera view, compute its $z$-coordinate relative to the light.

Shadow Mapping (Williams 1978)

Test if surface is visible to the light when computing contribution to the lighting equation.

- Render the scene from the light’s perspective and read back the $z$-buffer/shadow map.
- For each pixel in the camera view, compute its $z$-coordinate relative to the light
  - If it's further back than the value in the shadow map, it's in shadow
  - Otherwise, it's illuminated

Shadow Mapping (Williams 1978)

Test if surface is visible to the light when computing contribution to the lighting equation.

- The projection used for rendering from the light-source depends on the type of light:
  - Directional → Parallel
  - Point → Perspective

- Need to use multiple shadow maps if there are multiple lights in the scene