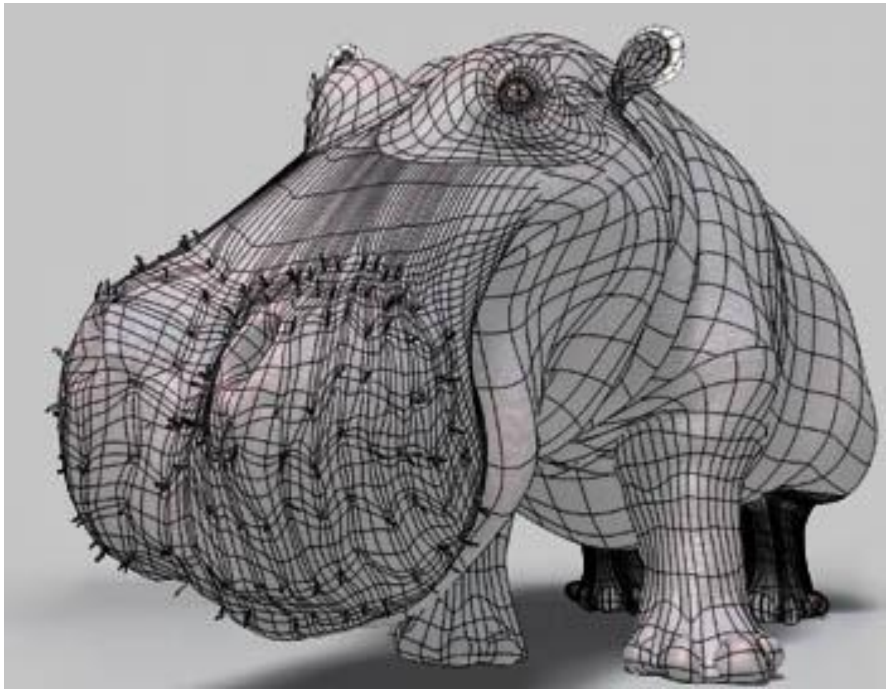
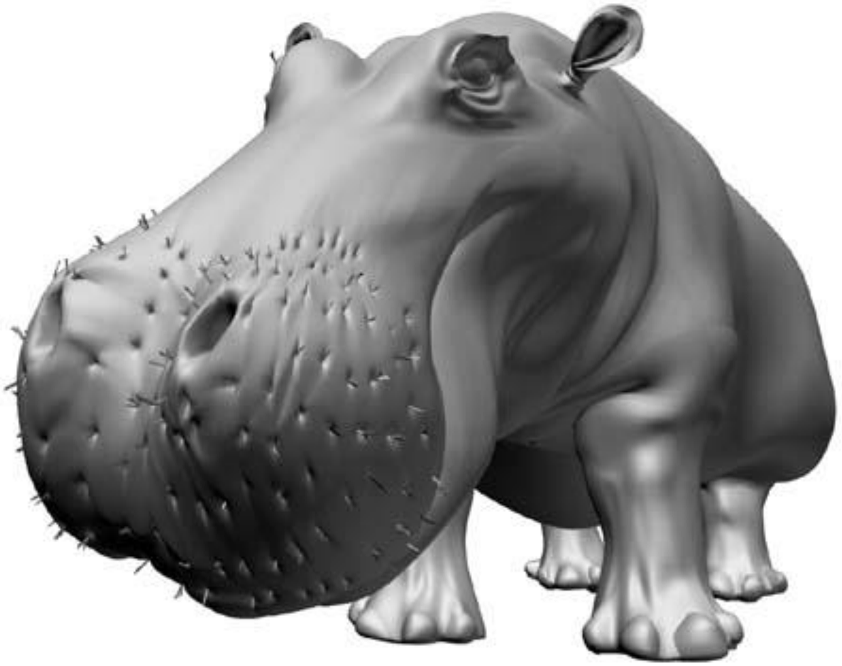
Texture Mapping

Michael Kazhdan

(601.457/657)

Textures

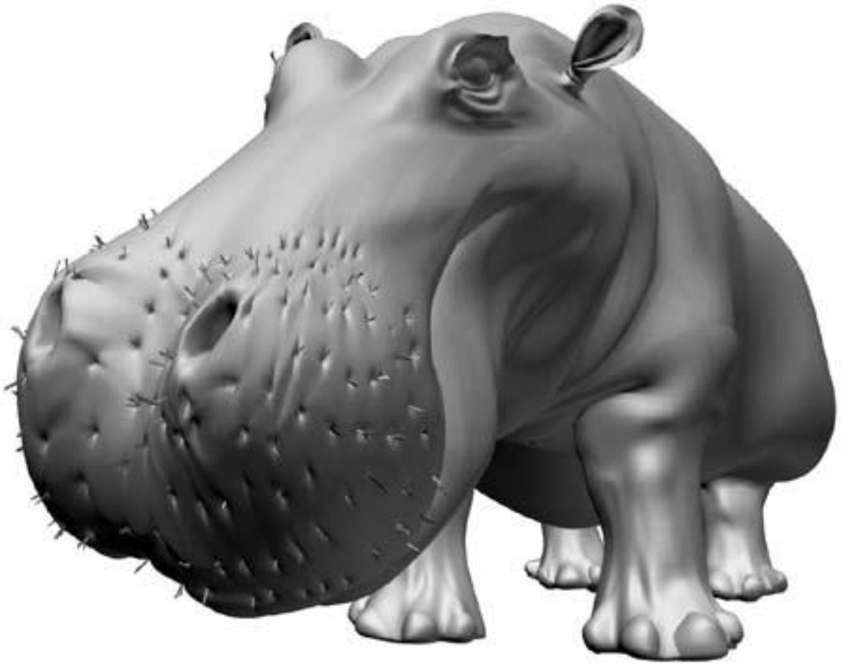




We know how to go from this… to this

[J. Birn]

Textures



But what about this…

to this?

[J. Birn]

Textures

- How do we draw surfaces with complex detail?

Target Model

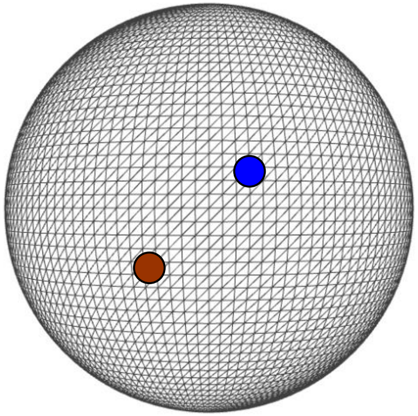
Textures

- How do we draw surfaces with complex detail?

Target Model

Direct:

- Tessellate *uniformly* (finely) and then associate the appropriate material properties to each vertex



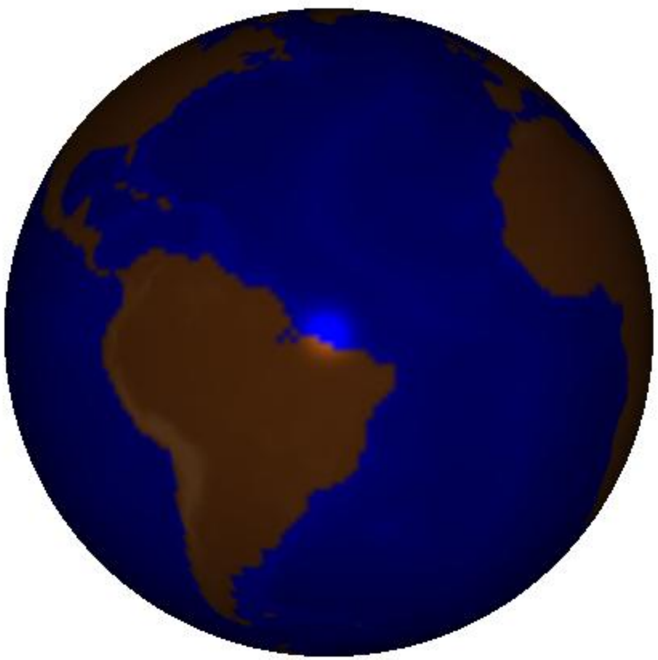
Complex Surface

Textures

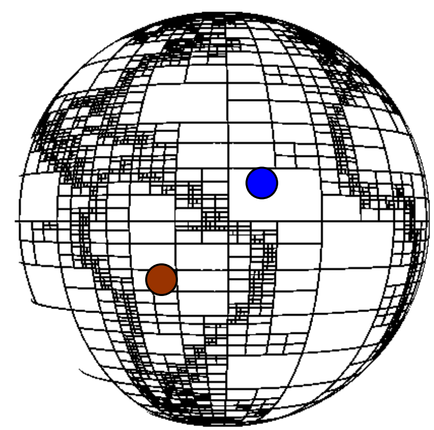
- How do we draw surfaces with complex detail?

<u>Direct</u>:

- Tessellate *adaptively* and then associate the appropriate material properties to each vertex



Target Model



Complex Surface

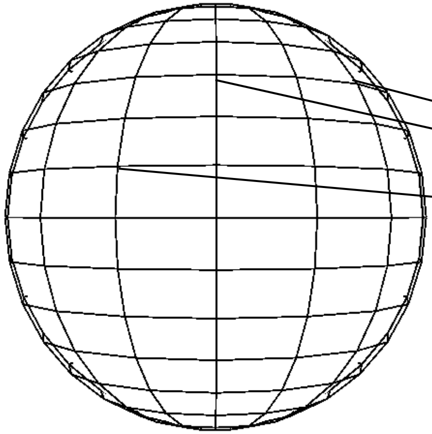
Textures

- How do we draw surfaces with complex detail?

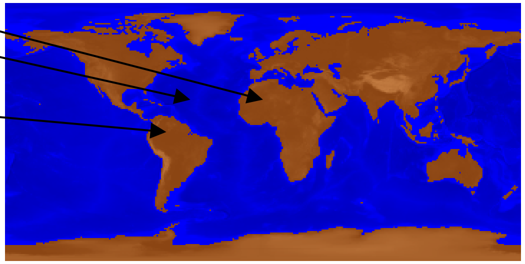
<u>Indirect</u>:

- Use a simple tessellation with an auxiliary *texture image*.
- Use *texture coordinates* stored at surface points to look up color values from the texture.

Target Model



Simple Surface

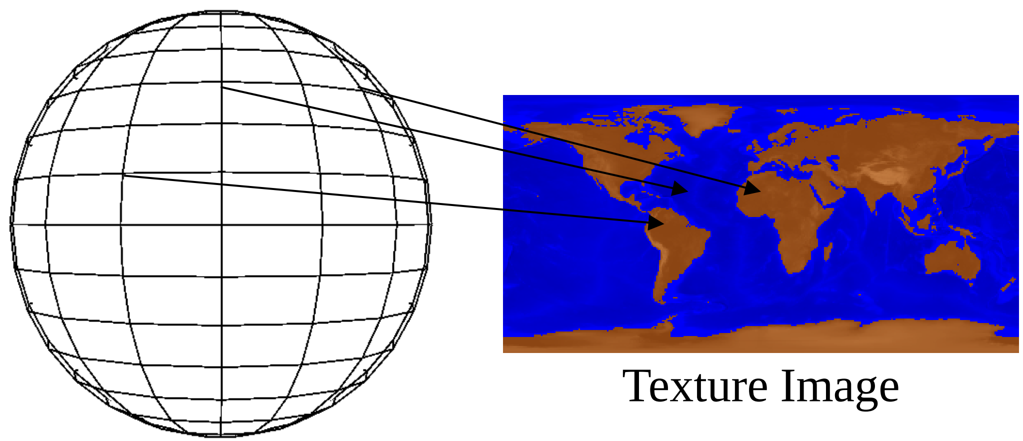


Texture Image

Textures

- Advantages:
  - The 3D model remains simple
  - It is easier to design/modify a texture image than it is to design/modify a signal on a surface.

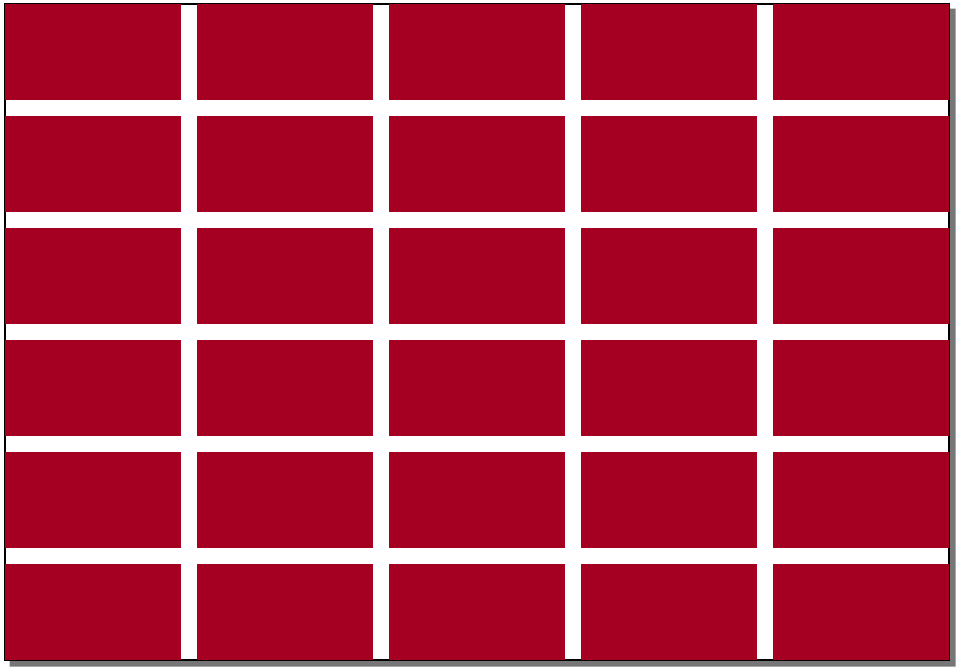
Target Model



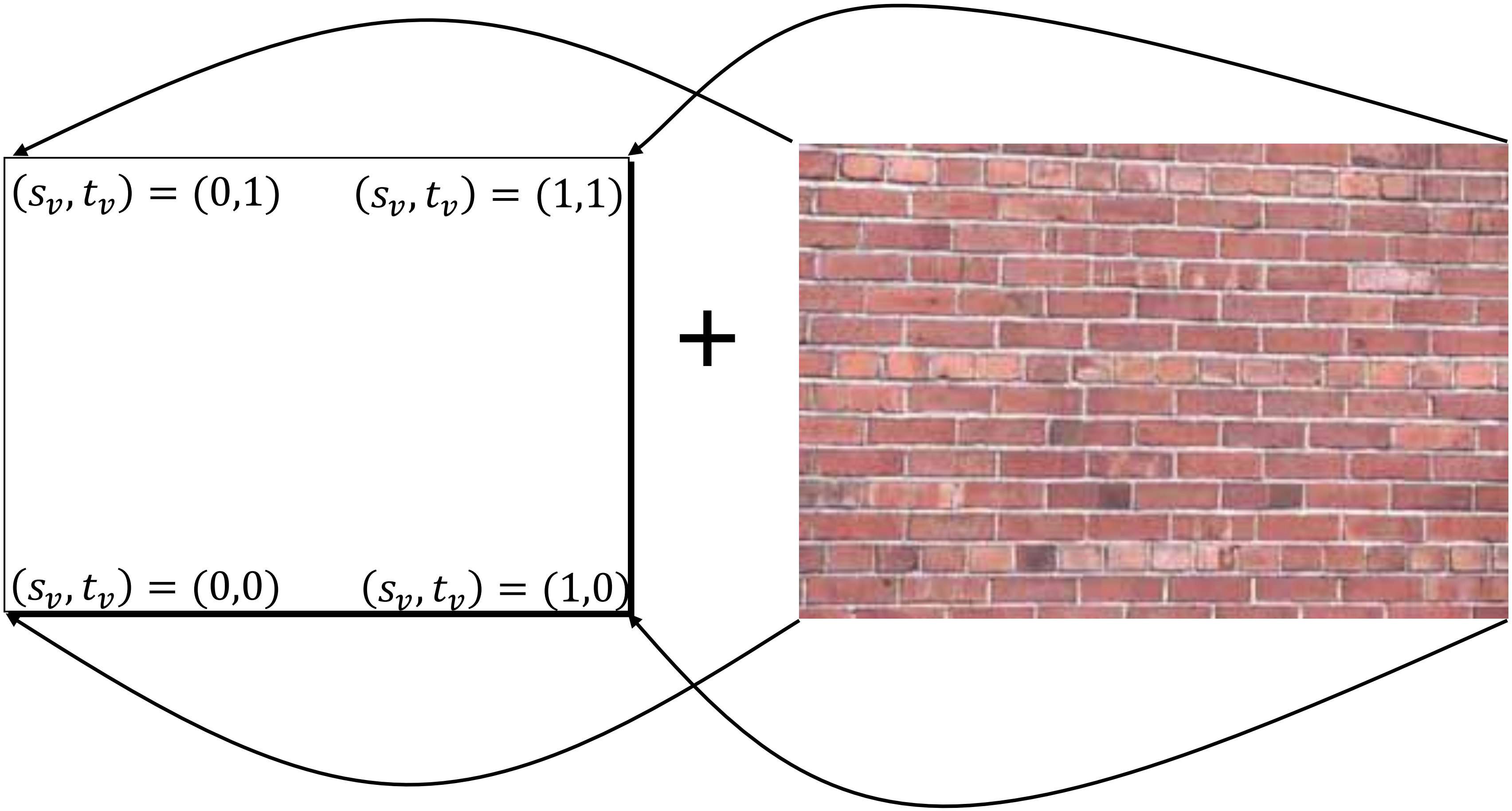
Simple Surface

Texture Image

Example: Brick Wall



Example: Brick Wall



Textures (2 dimensions)

<u>Implementation</u>:

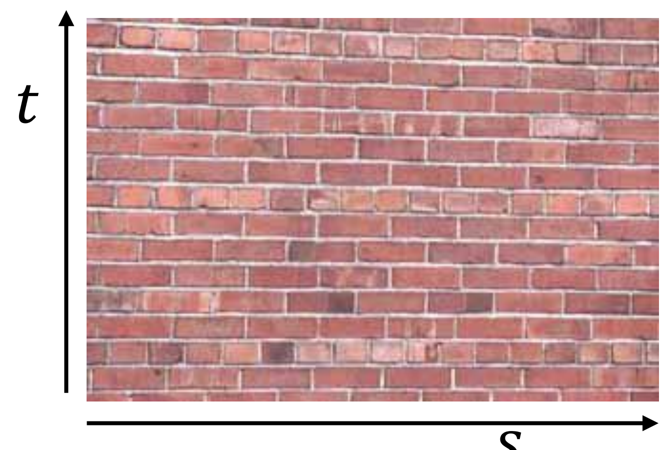
- Associate a *texture coordinate* to each vertex v:

$$(s_v, t_v) \text{ with } (0 \le s_v, t_v \le 1)$$

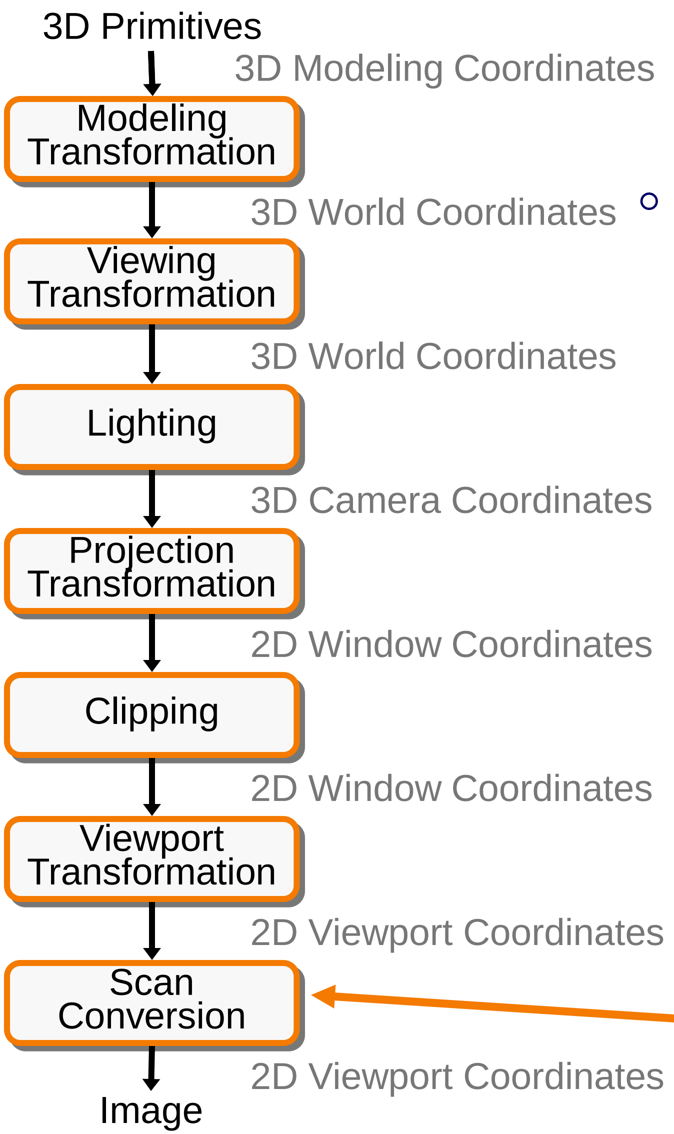
- When rasterizing, *interpolate* to get the texture coordinate to at a pixel:

$$(s_p, t_p)$$

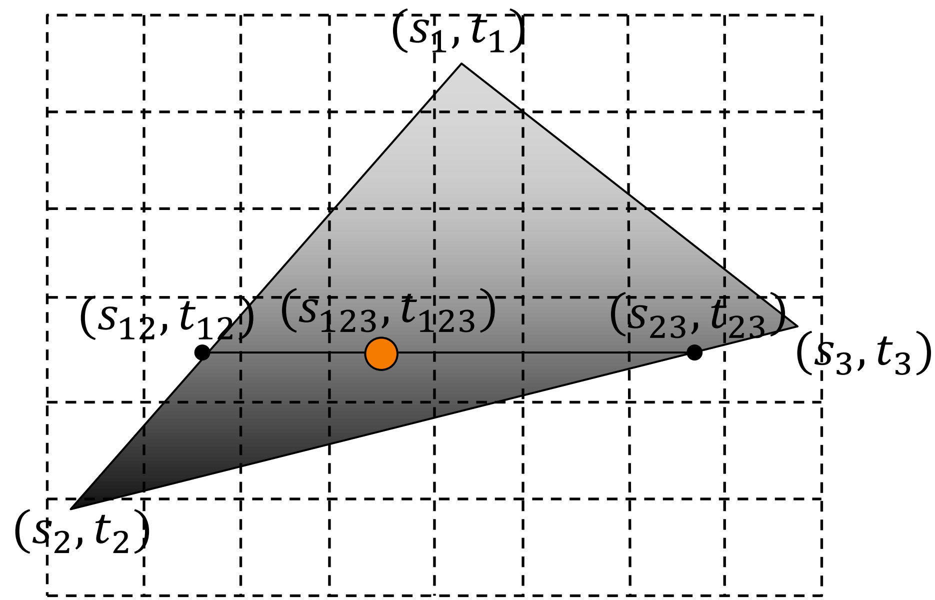
- *Sample* the texture at (s_p, t_p) to get the color at p.
- Texture elements are called *texels*
- Often 4 bytes (rgba) per texel



3D Rendering Pipeline



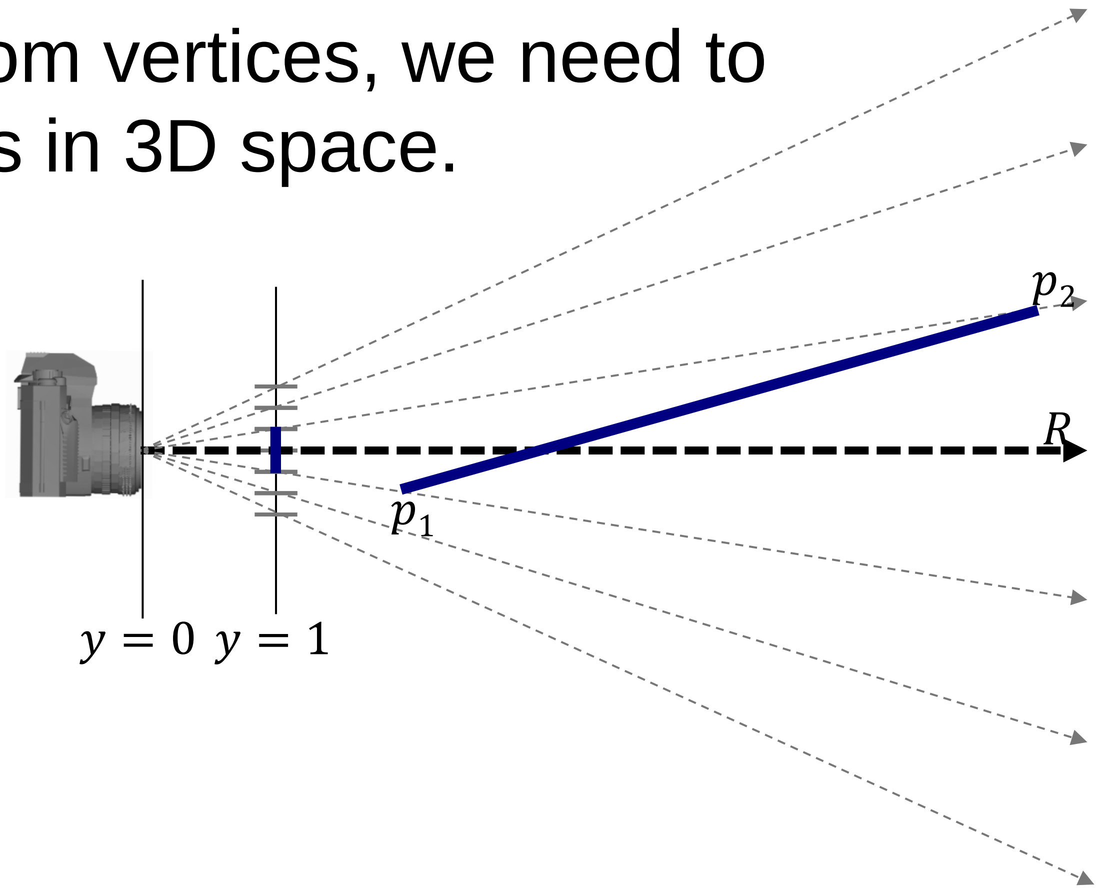
- Perform the texture interpolation and look-up while rasterizing the pixels.



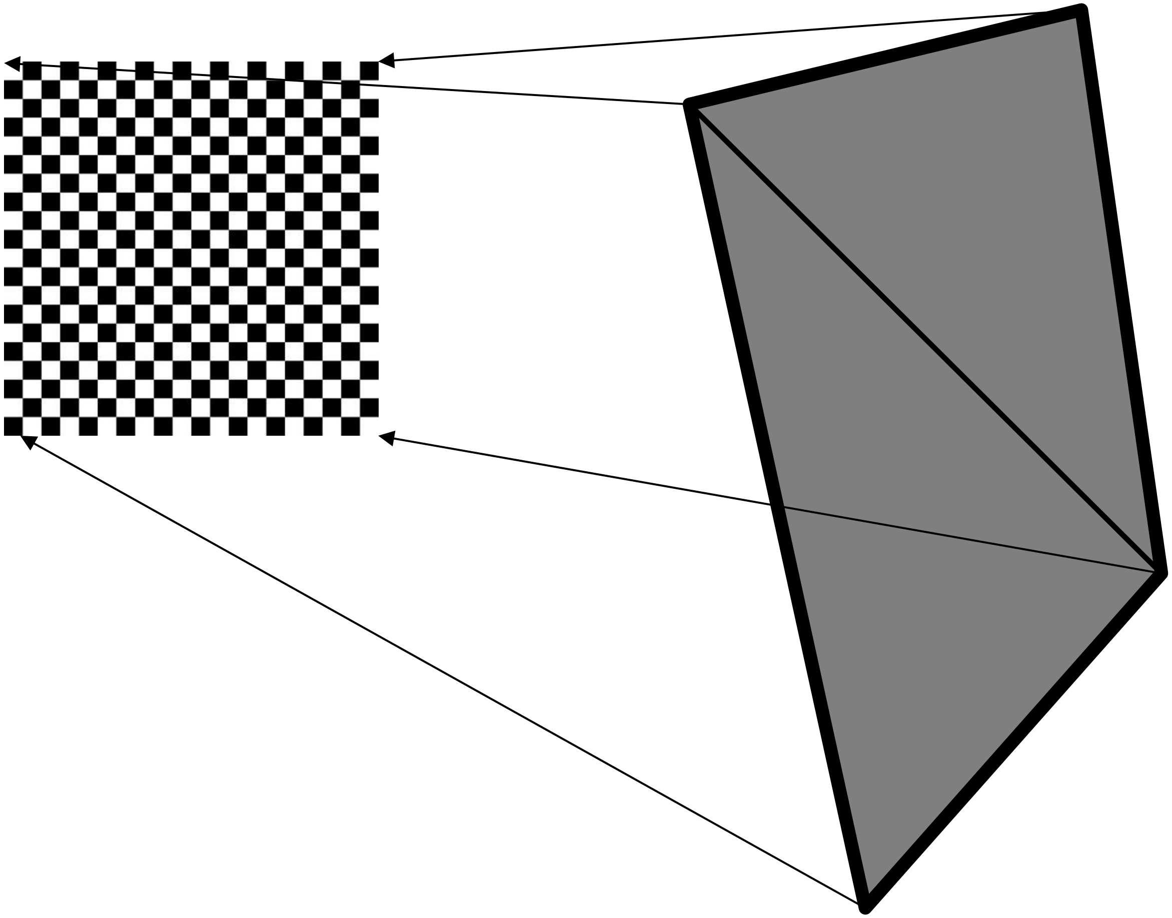
Texture Mapping

Recall (Perspective Divide):

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

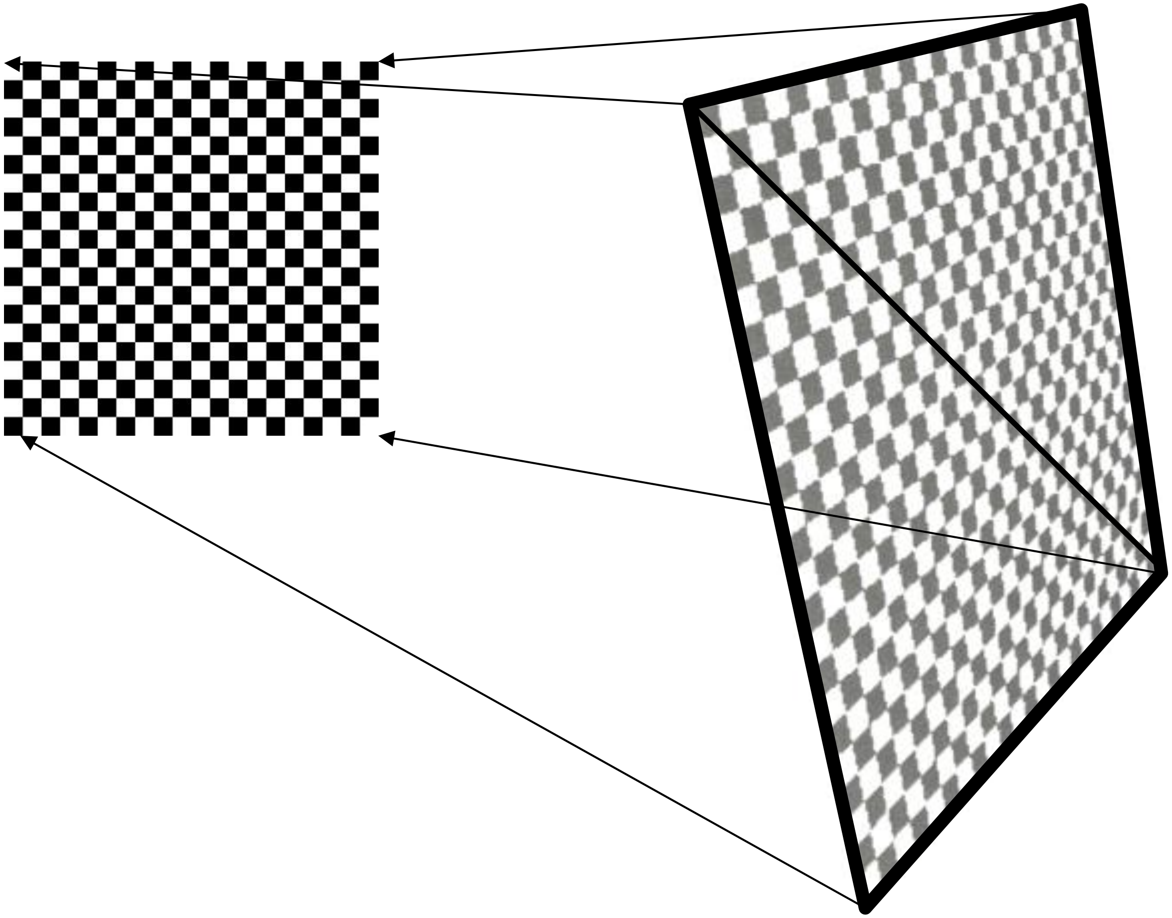


Texture Mapping



Hill Figure 8.42

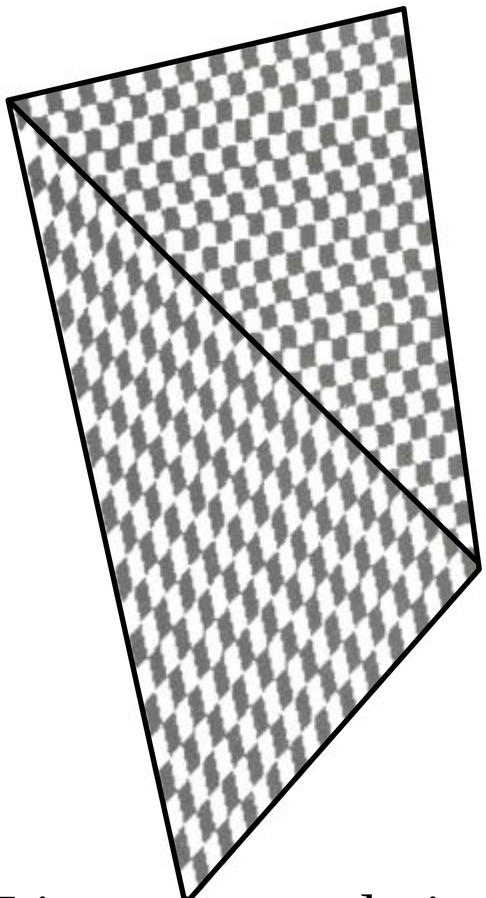
Texture Mapping



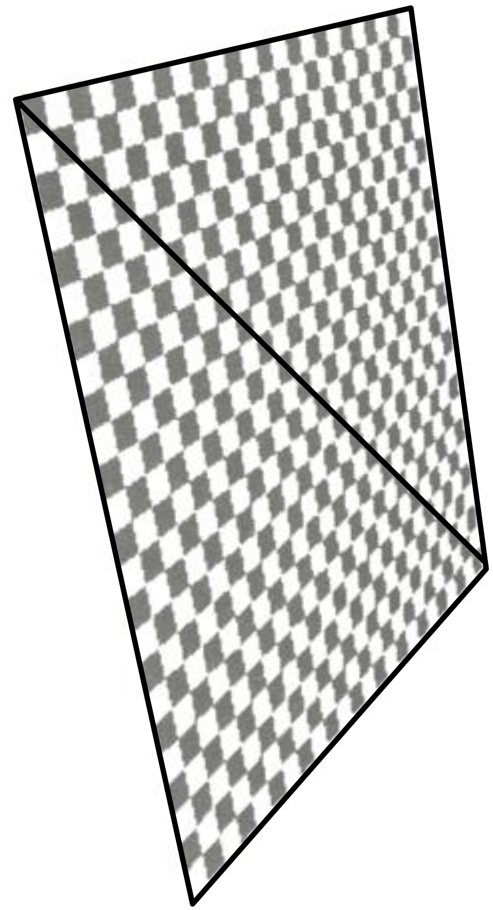
Correct interpolation
of texture coordinates
with perspective divide

Hill Figure 8.42

Texture Mapping



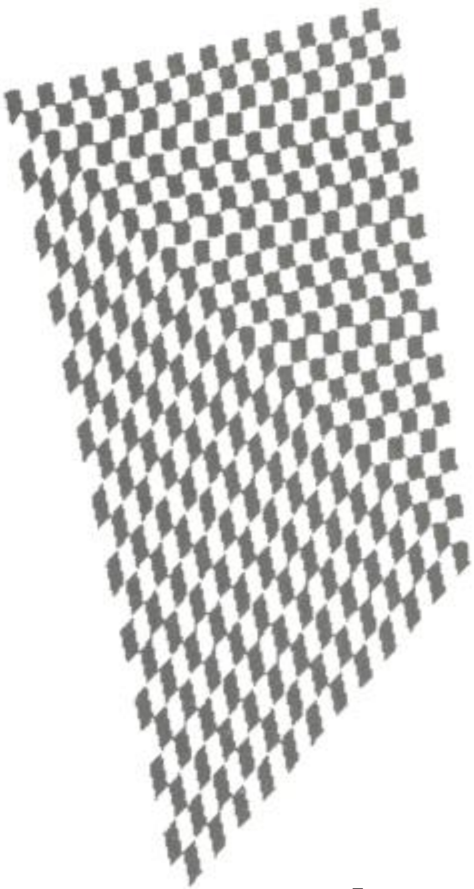
Linear interpolation
of texture coordinates
w/o perspective divide



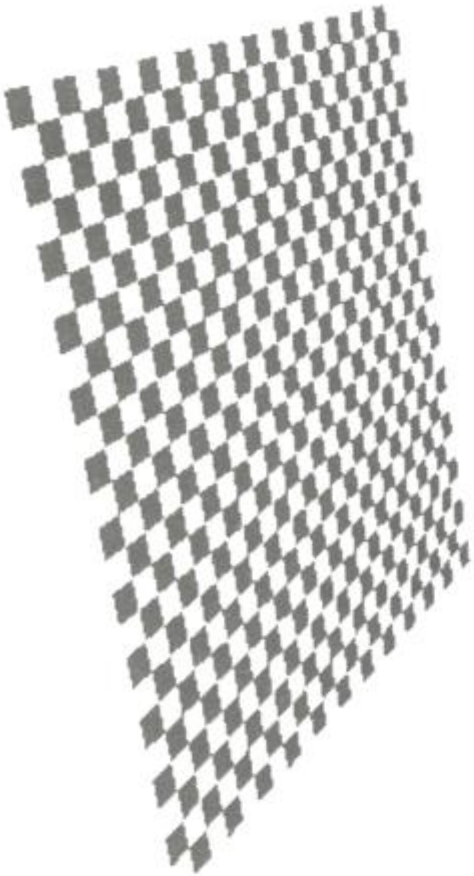
Correct interpolation
of texture coordinates
w/ perspective divide

Hill Figure 8.42

Texture Mapping



Linear interpolation
of texture coordinates
w/o perspective divide



Correct interpolation
of texture coordinates
w/ perspective divide

Hill Figure 8.42

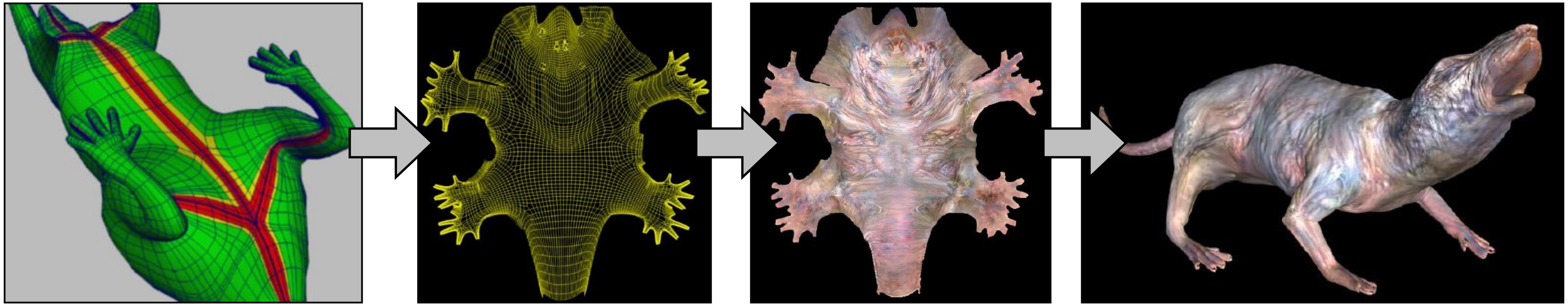
Overview

- 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

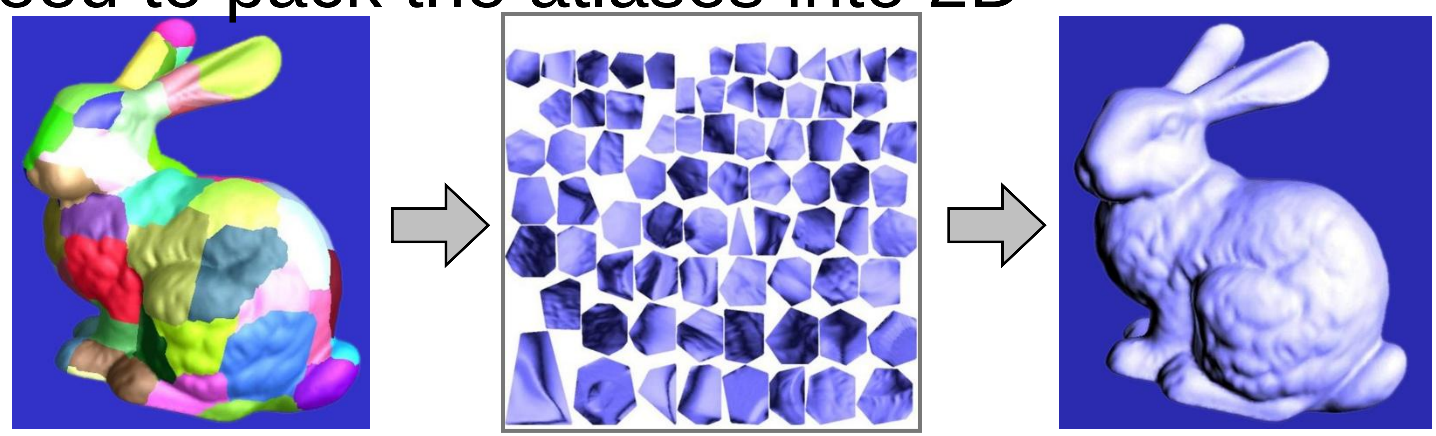


[Piponi, 2000]

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



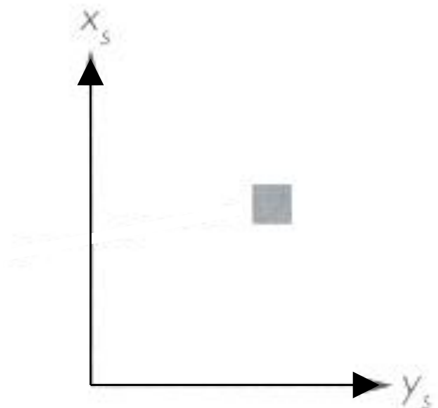
[Sander, 2001]

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:



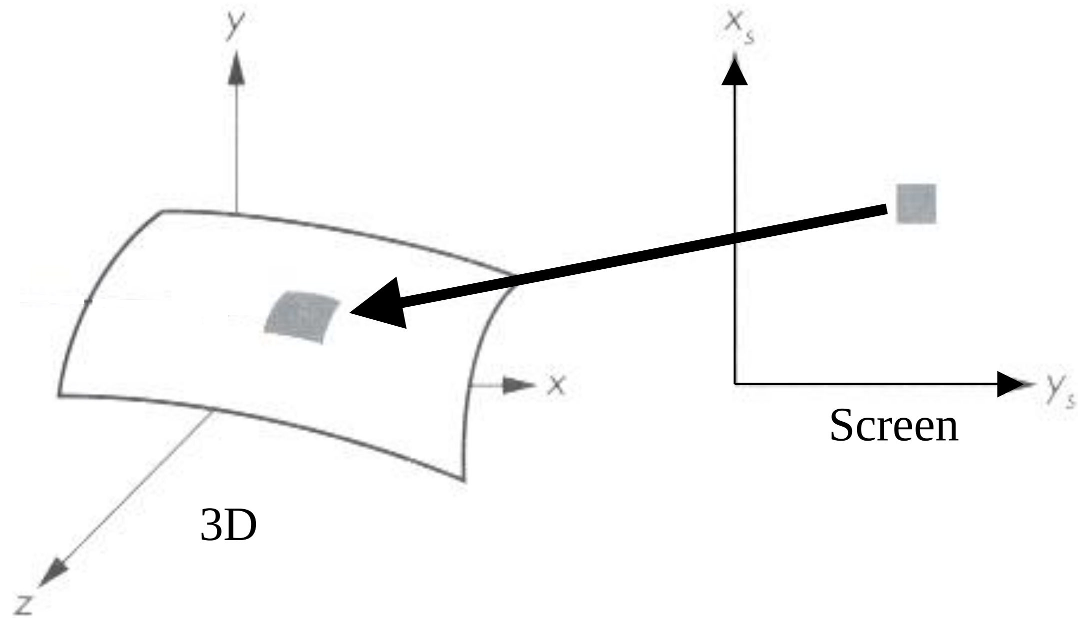
Screen

Angel Figure 9.4

Texture Filtering

Given pixel on a screen:

1. Determine the corresponding surface patch

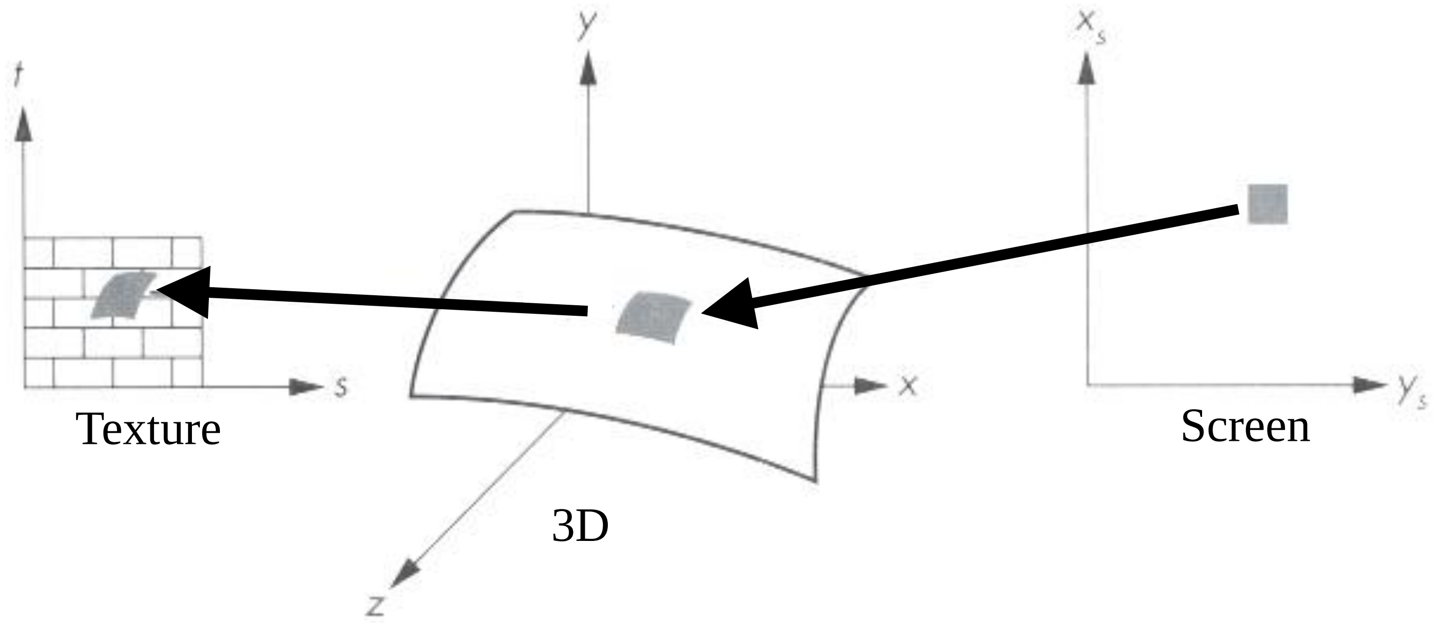


Angel Figure 9.4

Texture Filtering

Given pixel on a screen:

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

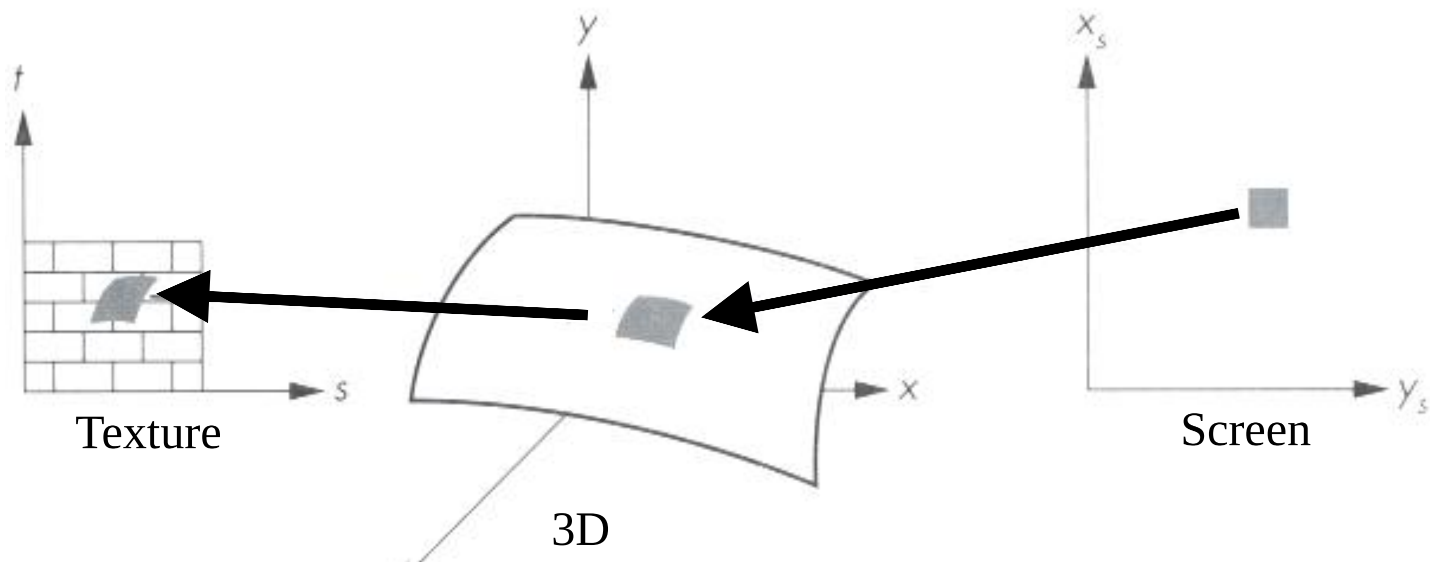


Angel Figure 9.4

Texture Filtering

Given pixel on a screen:

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

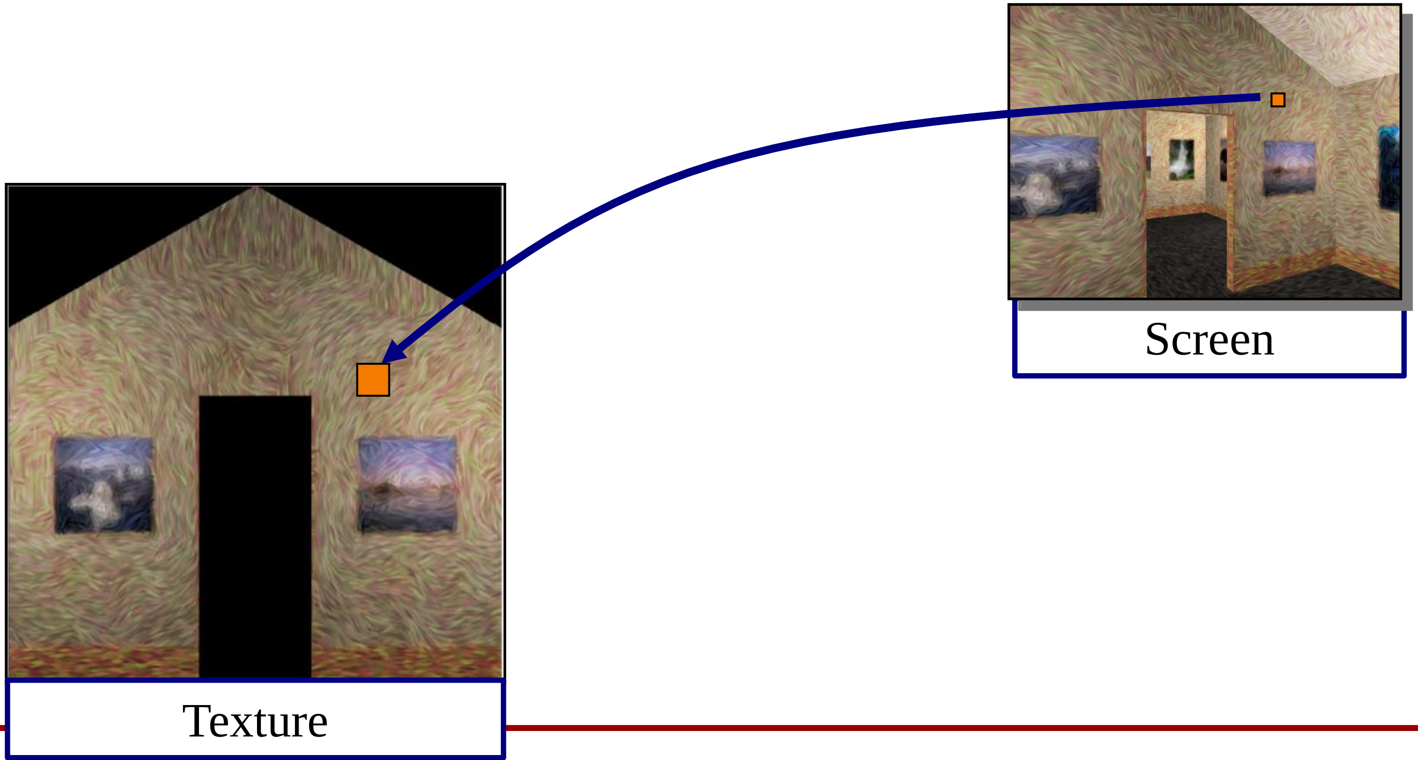


While the true shape of the texture patch mapping to a screen pixel may be hard to compute, we can approximate using the Jacobian.

Texture Filtering

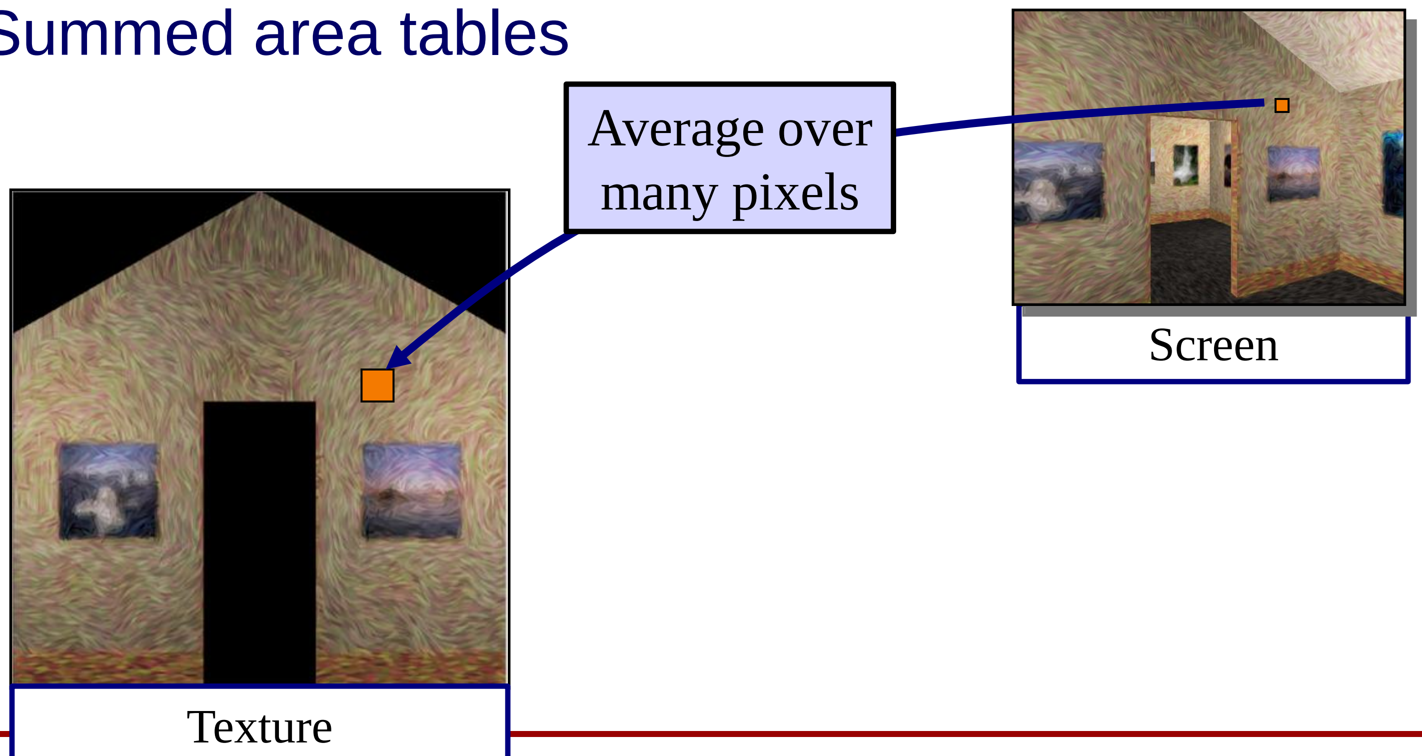
Given pixel on a screen:

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



Texture Filtering

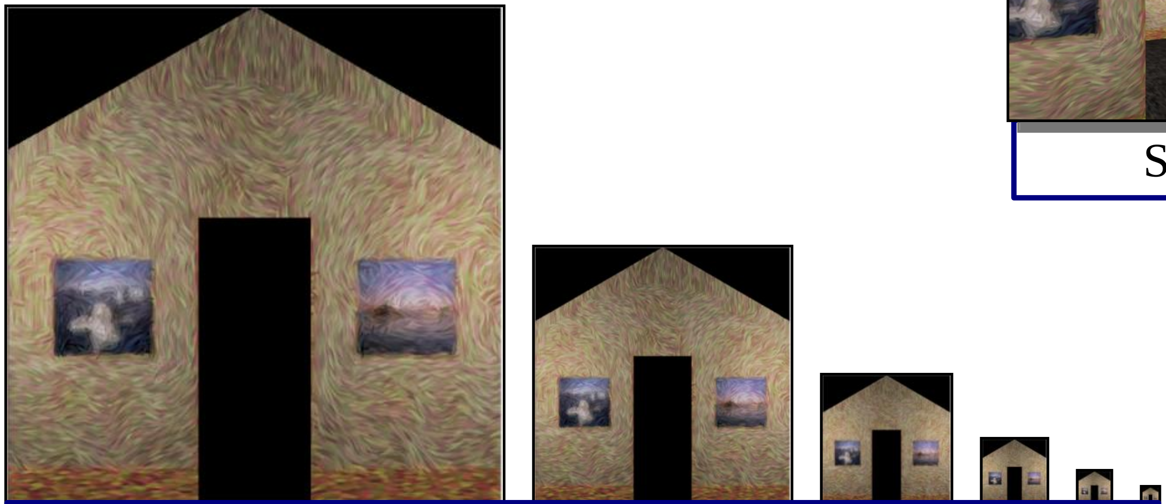
- Size of texture patch depends on the deformation
  - Computation is linear in the size of the pixel footprint
- Can pre-filter images for better performance
  - MIP (Multum In Parvo) maps
  - Summed area tables



MIP Maps

- Pre Processing: Compute a hierarchy of successively down-sampled texture images

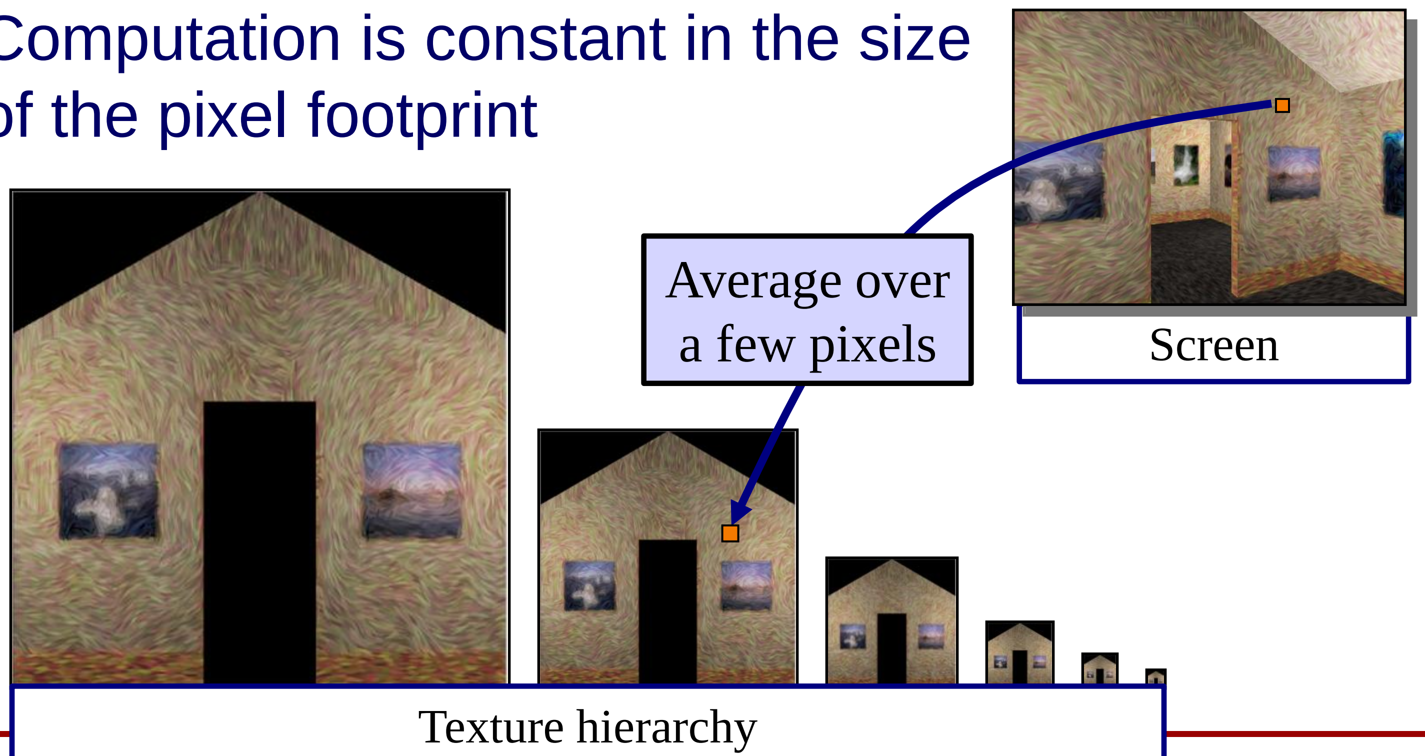
Screen



Texture hierarchy

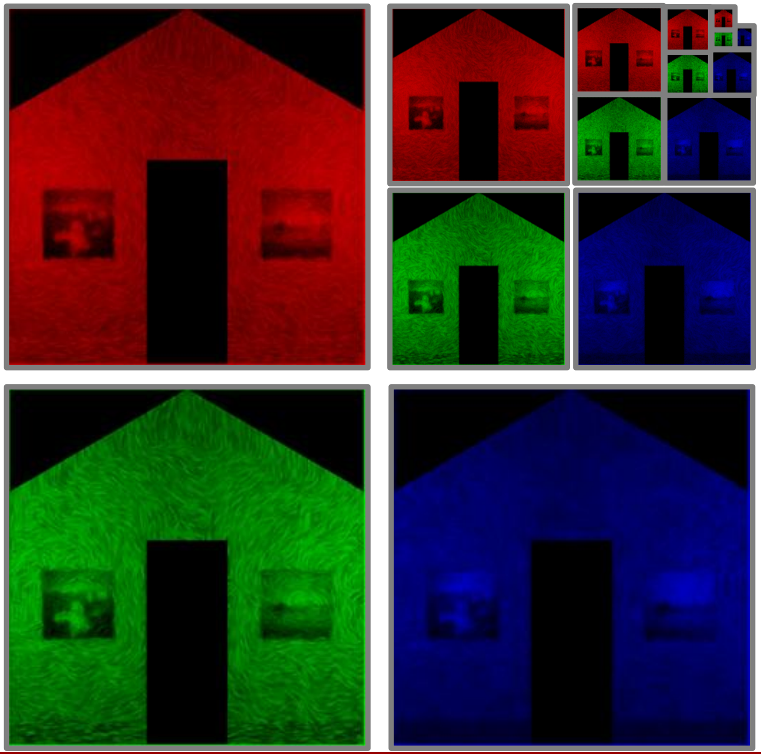
MIP Maps

- 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



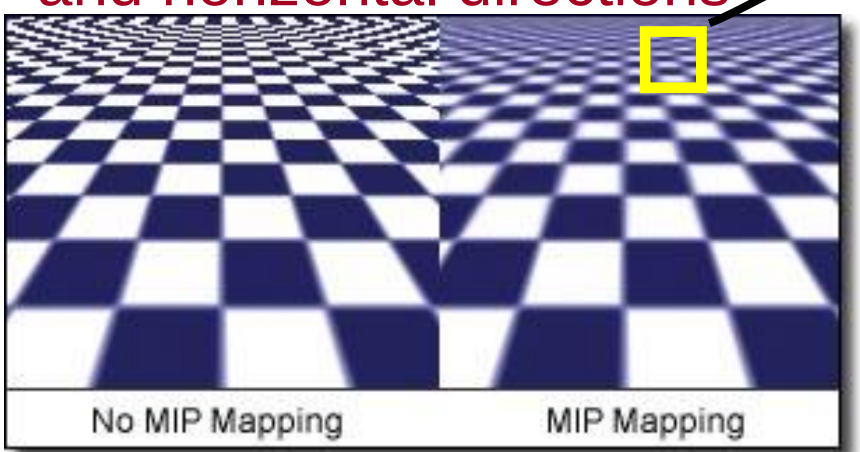
MIP Maps

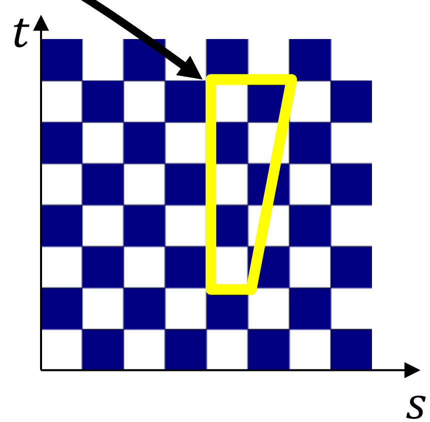
- Pre Processing: Compute a hierarchy of successively down-sampled texture images
  - ✓ Storage is only 4/3 the size of the input image



MIP Maps

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





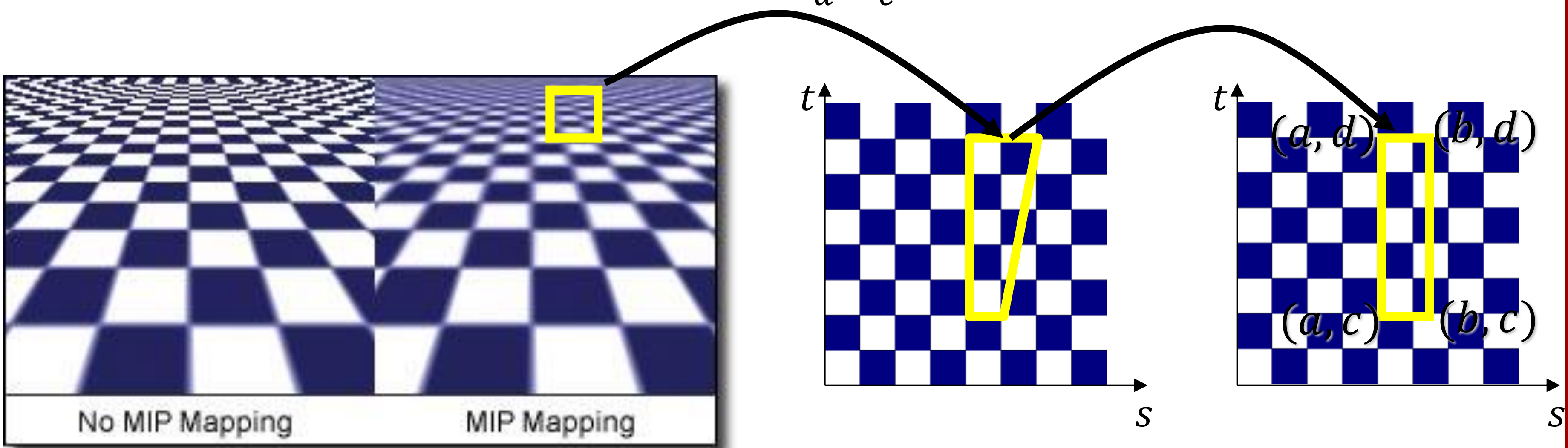
Again: we're trading aliasing for blurring!

Summed-Area Tables

Key Idea:

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

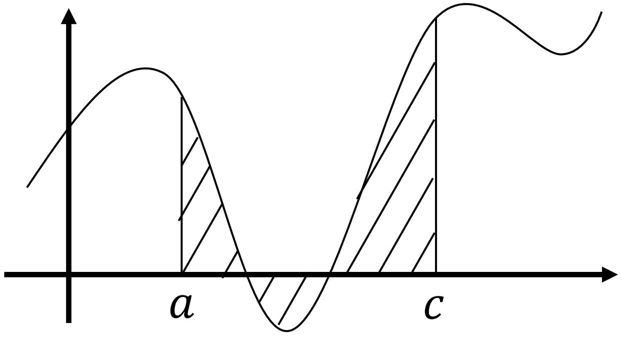
$$\mathrm{Sum}([a,b]\times[c,d]) = \int_a^b \int_c^d f(x,y)\,dy\,dx$$



Summed-Area Tables (1D)

Integration:

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



$$\int_a^c f(x)dx$$

Naïve Approach:

Pre-compute $S(a,b) \equiv \int_a^b f(x)\,dx$ and evaluate that

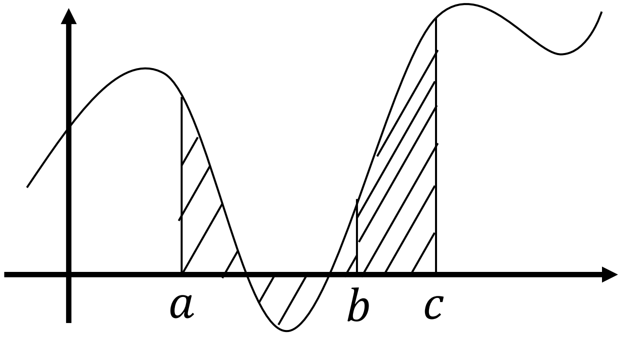
- ✓ Fast (constant time) look up
- ✗ Replaces 1D function f with 2D function S.

Summed-Area Tables (1D)

Integration:

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

$$\int_a^c f(x)dx$$



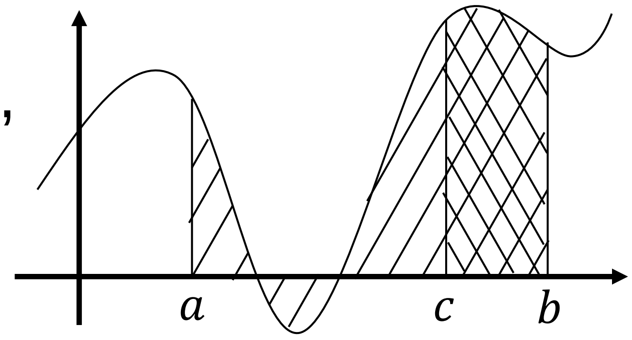
Recall:

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

$$\int_a^c f(x)\,dx = \int_a^b f(x)\,dx + \int_b^c f(x)\,dx$$

Summed-Area Tables (1D)

<u>Integration</u>:



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

$$\int_a^c f(x)dx$$

<u>Recall</u>:

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

$$\int_a^c f(x)\,dx = \int_a^b f(x)\,dx + \int_b^c f(x)\,dx$$

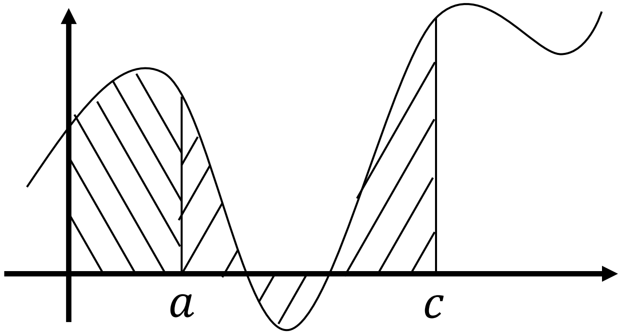
This is true even if c is outside the interval since:

$$\int_a^b f(x)\,dx = -\int_b^a f(x)\,dx$$

Summed-Area Tables (1D)

Approach:

Replace the integral over an interval, with two variable end-points with the difference between integrals with one variable end-point:



$$\int_{\boldsymbol{a}}^{\boldsymbol{c}} f(x)dx = \int_{0}^{\boldsymbol{c}} f(x)dx - \int_{0}^{\boldsymbol{a}} f(x)dx$$

⇒ Replace a look-up in the 2D function $S(a,c) = \int_a^c f(x)\,dx$
with two look-ups in in the 1D function $S_0(b) = \int_0^b f(x)\,dx$

Summed-Area Tables (2D)

Integration:

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

$$\int_a^b \int_c^d f(s,t)dt\,ds$$

Summed-Area Tables (2D)

Integration:

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

$$\int_a^b\int_c^d f(s,t)dt\,ds = \int_a^b\left(\int_0^d f(s,t)dt - \int_0^c f(s,t)dt\right)ds$$

Summed-Area Tables (2D)

Integration:

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

$$\begin{aligned}\int_a^b\int_c^d f(s,t)dt\,ds &= \int_a^b\left(\int_0^d f(s,t)dt-\int_0^c f(s,t)dt\right)ds\\ &= \int_0^b\left(\int_0^d f(s,t)dt-\int_0^c f(s,t)dt\right)ds-\int_0^a\left(\int_0^d f(s,t)dt-\int_0^c f(s,t)dt\right)ds\end{aligned}$$

Summed-Area Tables (2D)

Integration:

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

$$\begin{aligned}\int_a^b\int_c^d f(s,t)dt\,ds &= \int_a^b\left(\int_0^d f(s,t)dt-\int_0^c f(s,t)dt\right)ds\\ &= \int_0^b\left(\int_0^d f(s,t)dt-\int_0^c f(s,t)dt\right)ds-\int_0^a\left(\int_0^d f(s,t)dt-\int_0^c f(s,t)dt\right)ds\\ &= \int_0^b\int_0^d f(s,t)dt\,ds-\int_0^b\int_0^c f(s,t)dt\,ds-\int_0^a\int_0^d f(s,t)dt\,ds+\int_0^a\int_0^c f(s,t)dt\,ds\end{aligned}$$

Summed-Area Tables (2D)

Precomputing the 2D function:

$$S_{(0,0)}(x,y) \equiv \int_0^x \int_0^y f(s,t)\,dt\,ds$$

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

$$\int_a^b \int_c^d f(s,t)\,dt\,ds = S_{(0,0)}(b,d) - S_{(0,0)}(b,c) - S_{(0,0)}(a,d) + S_{(0,0)}(a,c)$$

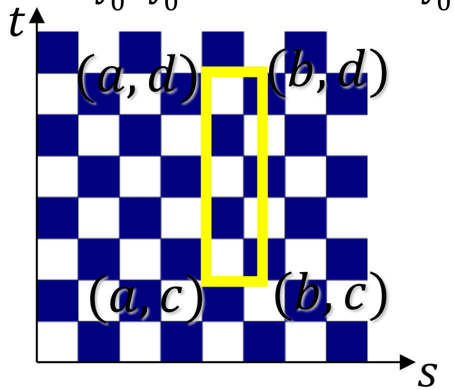
$$\begin{aligned}
\int_a^b \int_c^d f(s,t)dt\,ds &= \int_a^b \left(\int_0^d f(s,t)dt - \int_0^c f(s,t)dt \right) ds \\
&= \int_0^b \left(\int_0^d f(s,t)dt - \int_0^c f(s,t)dt \right) ds - \int_0^a \left(\int_0^d f(s,t)dt - \int_0^c f(s,t)dt \right) ds \\
&= \int_0^b \int_0^d f(s,t)dt\,ds - \int_0^b \int_0^c f(s,t)dt\,ds - \int_0^a \int_0^d f(s,t)dt\,ds + \int_0^a \int_0^c f(s,t)dt\,ds
\end{aligned}$$

Summed-Area Tables (2D)

Integration:

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

$$\boxed{\int_a^b\int_c^d f(s,t)dt\,ds} = \int_0^b\int_0^d f(s,t)dt\,ds - \int_0^b\int_0^c f(s,t)dt\,ds - \int_0^a\int_0^d f(s,t)dt\,ds + \int_0^a\int_0^c f(s,t)dt\,ds$$

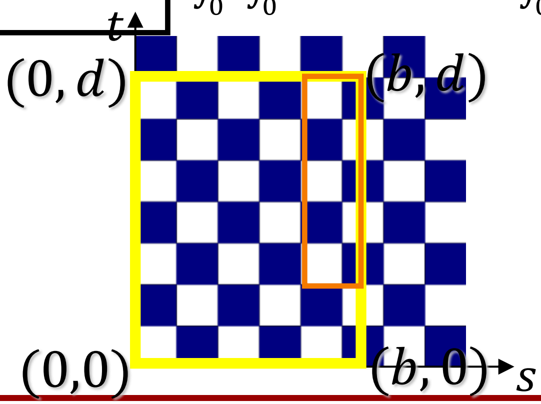


Summed-Area Tables (2D)

Integration:

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

$$\int_a^b\int_c^d f(s,t)dt\,ds = \boxed{\int_0^b\int_0^d f(s,t)dt\,ds} - \int_0^b\int_0^c f(s,t)dt\,ds - \int_0^a\int_0^d f(s,t)dt\,ds + \int_0^a\int_0^c f(s,t)dt\,ds$$

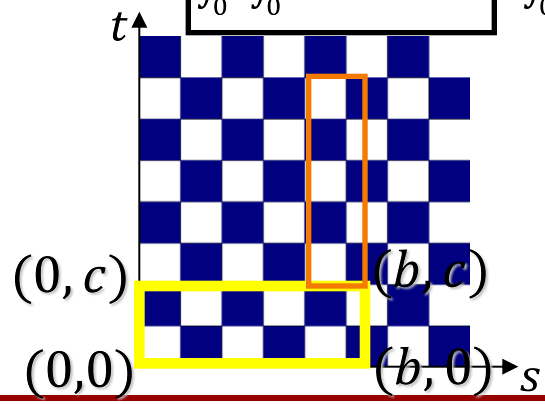


Summed-Area Tables (2D)

Integration:

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

$$\int_a^b\int_c^d f(s,t)dt\,ds = \int_0^b\int_0^d f(s,t)dt\,ds - \boxed{\int_0^b\int_0^c f(s,t)dt\,ds} - \int_0^a\int_0^d f(s,t)dt\,ds + \int_0^a\int_0^c f(s,t)dt\,ds$$

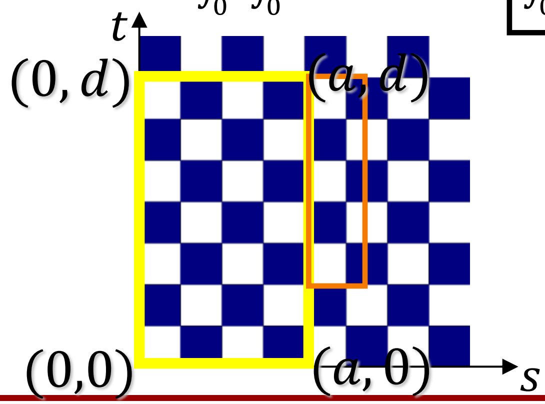


Summed-Area Tables (2D)

Integration:

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

$$\int_a^b\int_c^d f(s,t)dt\,ds = \int_0^b\int_0^d f(s,t)dt\,ds - \int_0^b\int_0^c f(s,t)dt\,ds - \boxed{\int_0^a\int_0^d f(s,t)dt\,ds} + \int_0^a\int_0^c f(s,t)dt\,ds$$

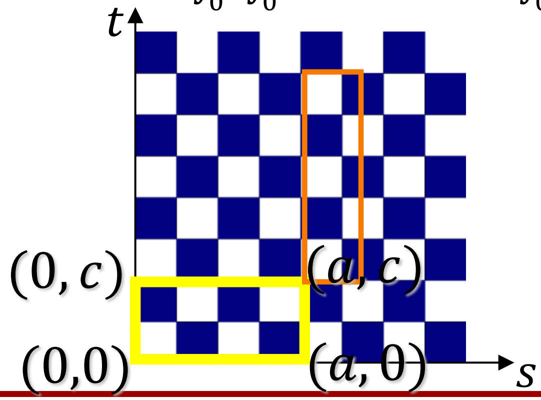


Summed-Area Tables (2D)

Integration:

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

$$\int_a^b\int_c^d f(s,t)dt\,ds = \int_0^b\int_0^d f(s,t)dt\,ds - \int_0^b\int_0^c f(s,t)dt\,ds - \int_0^a\int_0^d f(s,t)dt\,ds + \boxed{\int_0^a\int_0^c f(s,t)dt\,ds}$$



Summed-Area Tables (Pre-Process)

- Precompute the values of the integral:

$$S_{(0,0)}(a,b) = \int_0^a \int_0^b f(s,t)\, dt\, ds$$

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

Input image → **Summed area table**

1	2	4	0
0	3	1	1
4	2	0	1
1	2	1	3

Summed-Area Tables (Pre-Process)

- Precompute the values of the integral:

$$S_{(0,0)}(a,b) = \int_0^a \int_0^b f(s,t)\, dt\, ds$$

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

Input image

1	2	4	0
0	3	1	1
4	2	0	1
$\boxed{1}$	2	1	3

$\rightarrow$

Summed area table

1			

Summed-Area Tables (Pre-Process)

- Precompute the values of the integral:

$$S_{(0,0)}(a,b) = \int_0^a \int_0^b f(s,t)\,dt\,ds$$

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

Input image

1	2	4	0
0	3	1	1
4	2	0	1
1	2	1	3

Summed area table

1	3		

Summed-Area Tables (Pre-Process)

- Precompute the values of the integral:

$$S_{(0,0)}(a,b) = \int_0^a \int_0^b f(s,t)\, dt\, ds$$

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

Input image

1	2	4	0
0	3	1	1
4	2	0	1
1	2	1	3

Summed area table

1	3	4	

Summed-Area Tables (Pre-Process)

- Precompute the values of the integral:

$$S_{(0,0)}(a,b) = \int_0^a \int_0^b f(s,t)\, dt\, ds$$

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

Input image

1	2	4	0
0	3	1	1
4	2	0	1
1	2	1	3

Summed area table

1	3	4	7

Summed-Area Tables (Pre-Process)

- Precompute the values of the integral:

$$S_{(0,0)}(a,b) = \int_0^a \int_0^b f(s,t)\, dt\, ds$$

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

Input image

1	2	4	0
0	3	1	1
4	2	0	1
1	2	1	3

Summed area table

5			
1	3	4	7

Summed-Area Tables (Pre-Process)

- Precompute the values of the integral:

$$S_{(0,0)}(a,b) = \int_0^a \int_0^b f(s,t)\, dt\, ds$$

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

Input image

1	2	4	0
0	3	1	1
4	2	0	1
1	2	1	3

Summed area table

5	9		
1	3	4	7

Summed-Area Tables (Pre-Process)

- Precompute the values of the integral:

$$S_{(0,0)}(a,b) = \int_0^a \int_0^b f(s,t)\, dt\, ds$$

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

Input image

1	2	4	0
0	3	1	1
4	2	0	1
1	2	1	3

→

Summed area table

6	15	21	26
5	12	14	19
5	9	10	14
1	3	4	7

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

1	2	4	0
0	3	1	1
4	2	0	1
1	2	1	3

Input image

6	15	21	26
5	12	14	19
5	9	10	14
1	3	4	7

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

$$\mathrm{Sum}([1,3] \times [2,3]) = S_{(0,0)}(3,3)$$

1	2	4	0
0	3	1	1
4	2	0	1
1	2	1	3

Input image

6	15	21	26
5	12	14	19
5	9	10	14
1	3	4	7

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

$$\mathrm{Sum}([1,3] \times [2,3]) = S_{(0,0)}(3,3) - S_{(0,0)}(0,3)$$

1	2	4	0
0	3	1	1
4	2	0	1
1	2	1	3

Input image

6	15	21	26
5	12	14	19
5	9	10	14
1	3	4	7

Summed-area table

Summed-Area Tables (Run-Time)

<u>Example</u>:

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

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

$$\mathrm{Sum}([1,3] \times [2,3]) = S_{(0,0)}(3,3) - S_{(0,0)}(0,3) - S_{(0,0)}(3,1)$$

1	2	4	0
0	3	1	1
4	2	0	1
1	2	1	3

Input image

6	15	21	26
5	12	14	19
5	9	10	14
1	3	4	7

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

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

1	2	4	0
0	3	1	1
4	2	0	1
1	2	1	3

Input image

6	15	21	26
5	12	14	19
5	9	10	14
1	3	4	7

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

$$\mathrm{Sum}([1,3]\times[2,3]) = S_{(0,0)}(3,3) - S_{(0,0)}(0,3) - S_{(0,0)}(3,1) + S_{(0,0)}(0,1)$$
$$= 26 - 6 - 14 + 5 = 11$$
$$\mathrm{Average}([1,3]\times[2,3]) = \frac{\mathrm{Sum}([1,3]\times[2,3])}{\mathrm{Area}([1,3]\times[2,3])} = \frac{11}{6}$$

1	2	4	0
0	3	1	1
4	2	0	1
1	2	1	3

Input image

6	15	21	26
5	12	14	19
5	9	10	14
1	3	4	7

Summed-area table

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 $[0,255 \cdot \text{width} \cdot \text{height}]$

1	2	4	0
0	3	1	1
4	2	0	1
1	2	1	3

Input image

6	15	21	26
5	12	14	19
5	9	10	14
1	3	4	7

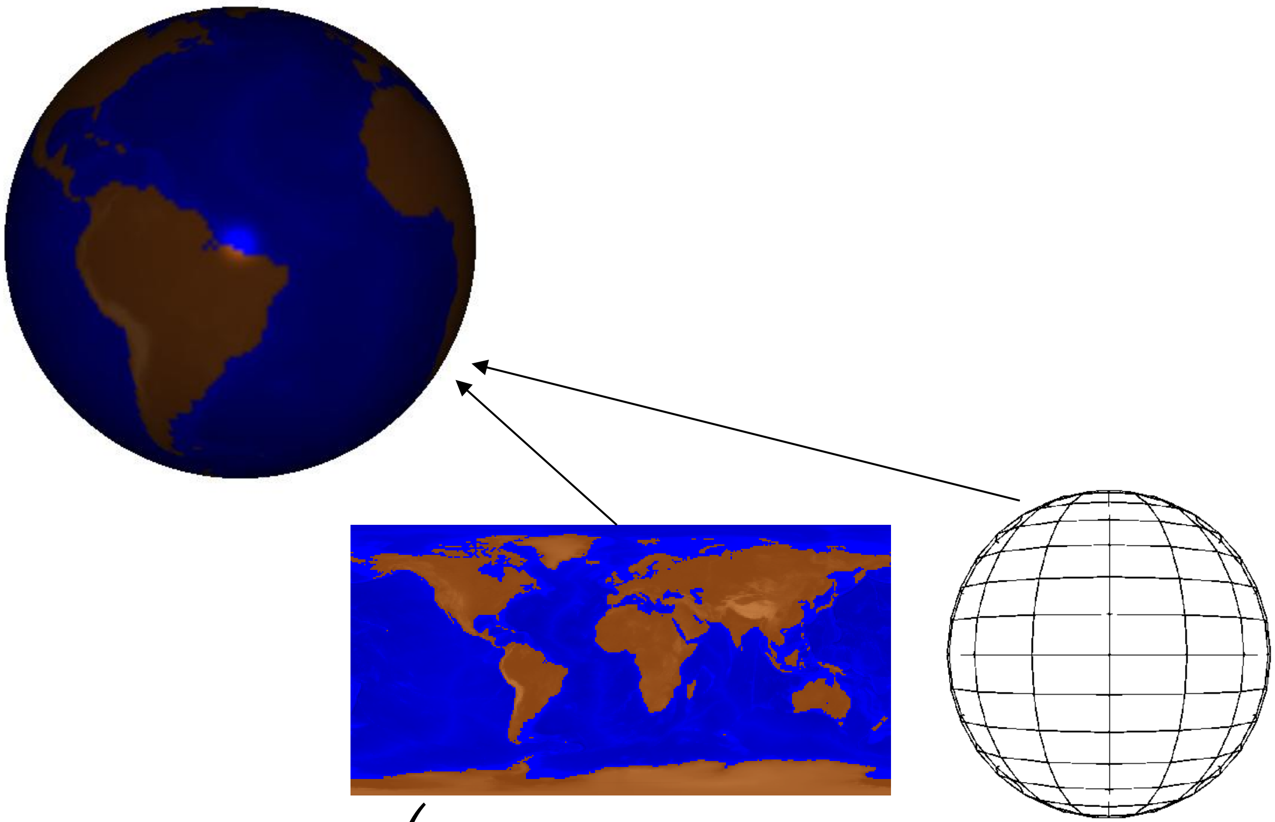
Summed-area table

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

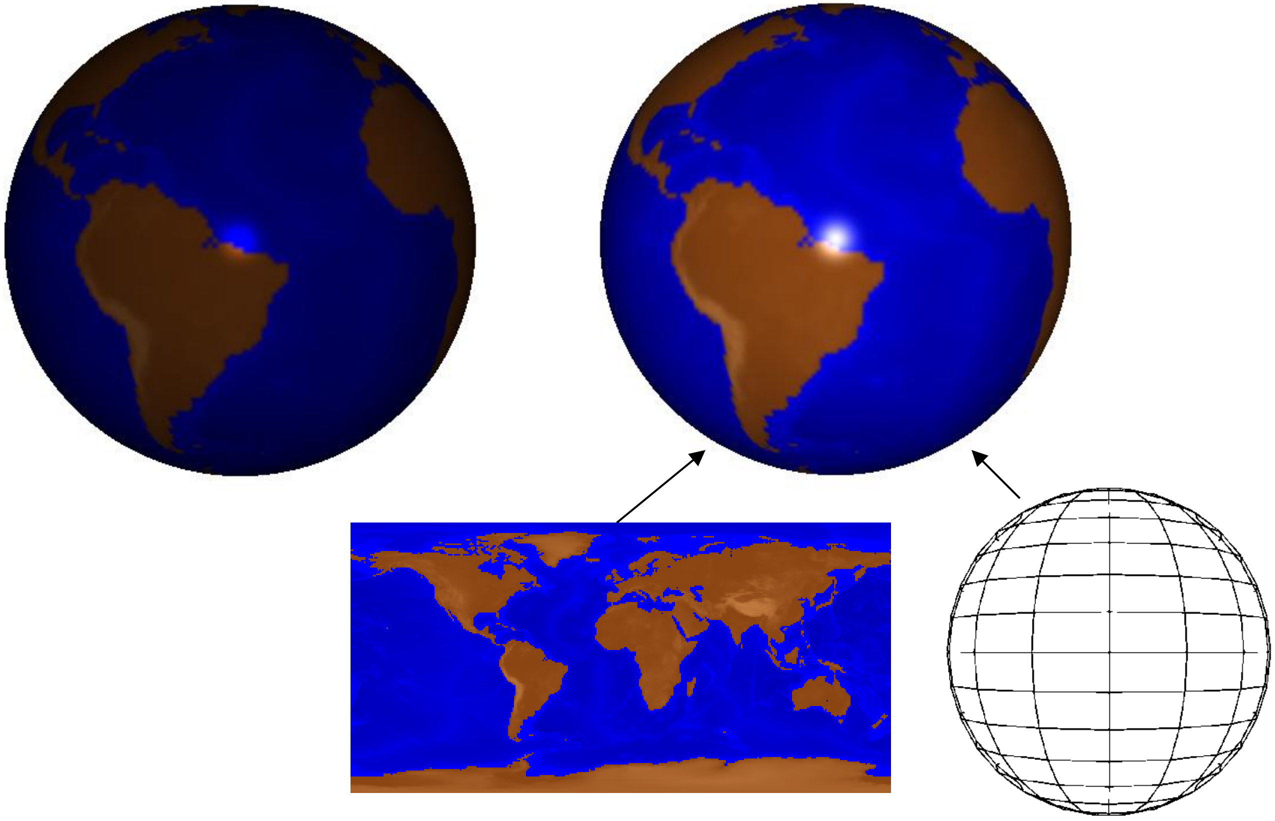


$$I = \boldsymbol{T}(\boldsymbol{s},\boldsymbol{t})\left(I_E + K_A I_{AL} + \sum_i \left(K_D\langle \vec{N},\vec{L}_i\rangle + K_S\langle \vec{V},\vec{R}_i\rangle^n\right) I_i\right)$$

Illumination Mapping

Map texture values to a material parameter

Modulation Diffuse



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

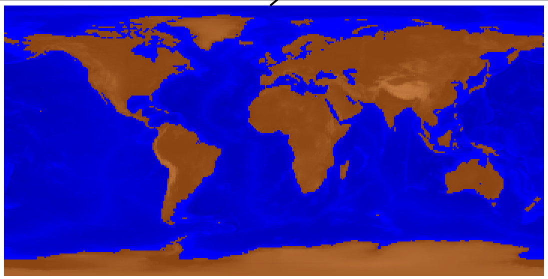
Illumination Mapping

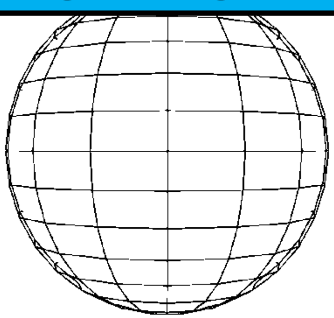
Map texture values to a material parameter

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

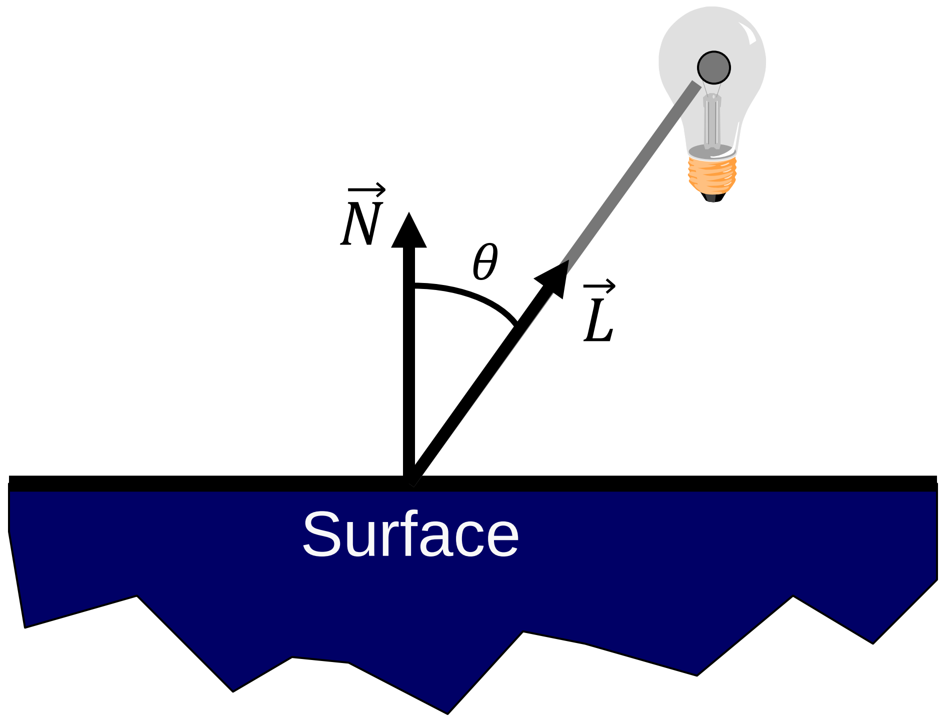




$$I = I_E + K_A I_{AL} + \sum_i \left(\boldsymbol{T}(\boldsymbol{s},\boldsymbol{t}) \langle \vec{N}, \vec{L}_i \rangle + K_S \langle \vec{V}, \vec{R}_i \rangle^n \right) I_i$$

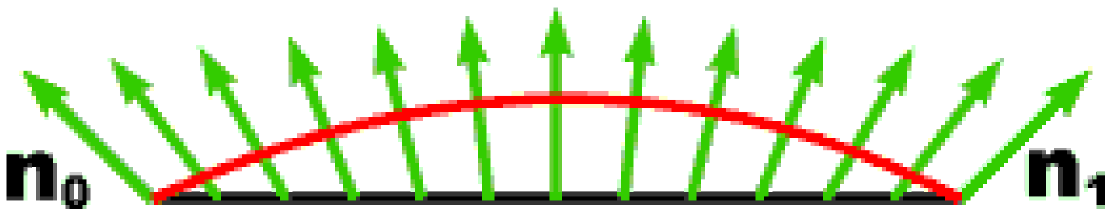
Bump Mapping

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



$$I = I_E + K_A I_{AL} + \sum_i \left(K_D \langle \boxed{\vec{N}}, \vec{L}_i \rangle + K_s \langle \vec{V}, \boxed{\vec{R}_i} \rangle^n \right) I_i$$

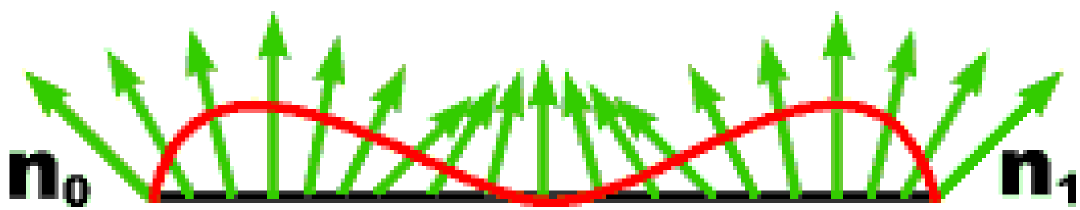
Bump Mapping



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

⇓

approximates a smoothly curved surface



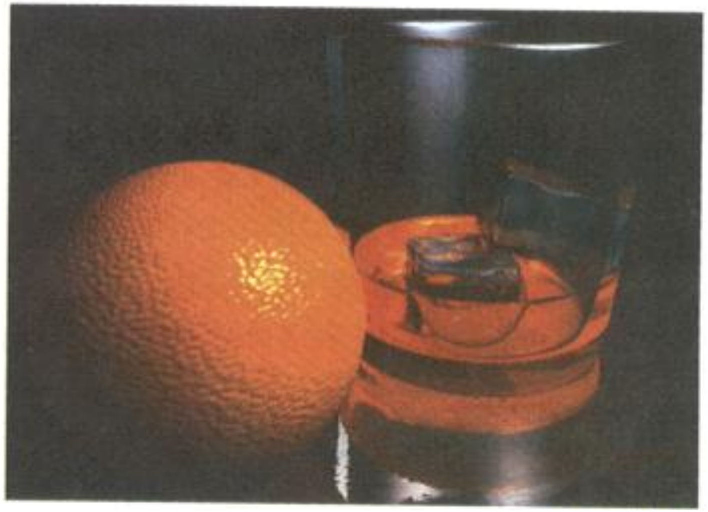
Bump maps encode the normals in the texture

⇓

approximates a more complex undulating surface

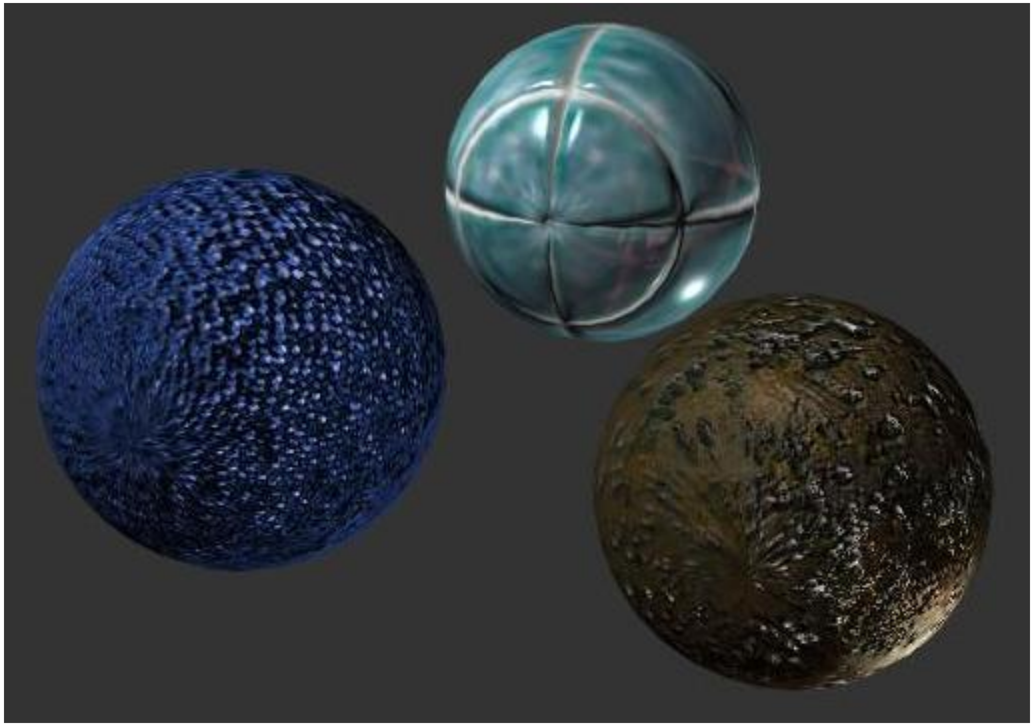
P. Rheingans

Bump Mapping



H&B Figure 14.100

Bump Mapping



Note that bump mapping
does not change object silhouette

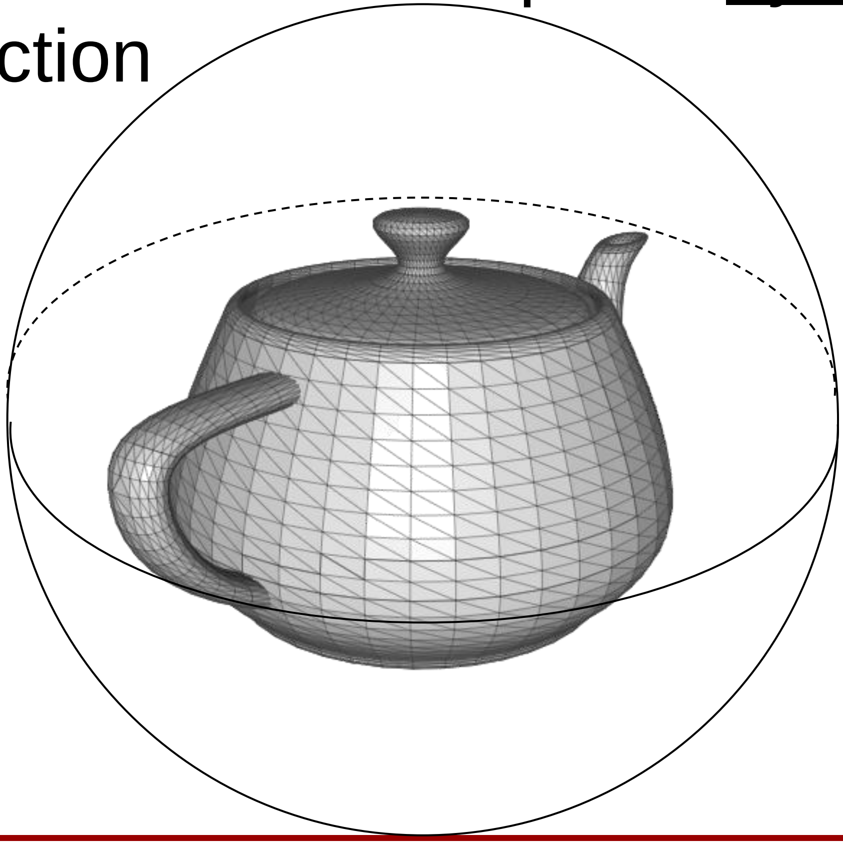
Siggraph.org

Environment Mapping

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

Environment Mapping

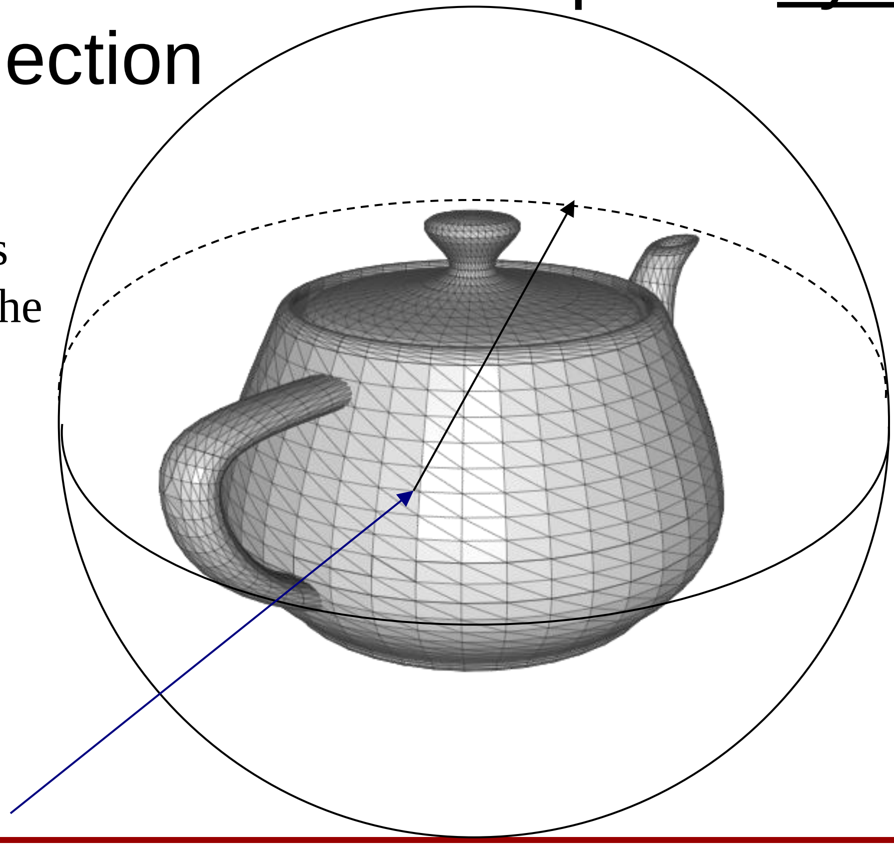
- Generate a spherical/cubic map of the environment around the model.
- Texture coordinates are computed <u>dynamically</u> through reflection

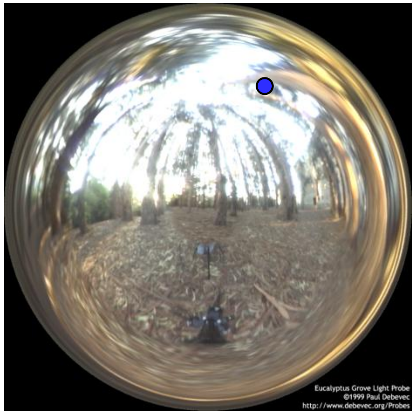


Environment Mapping

- Generate a spherical/cubic map of the environment around the model.
- Texture coordinates are computed <u>dynamically</u> through reflection

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

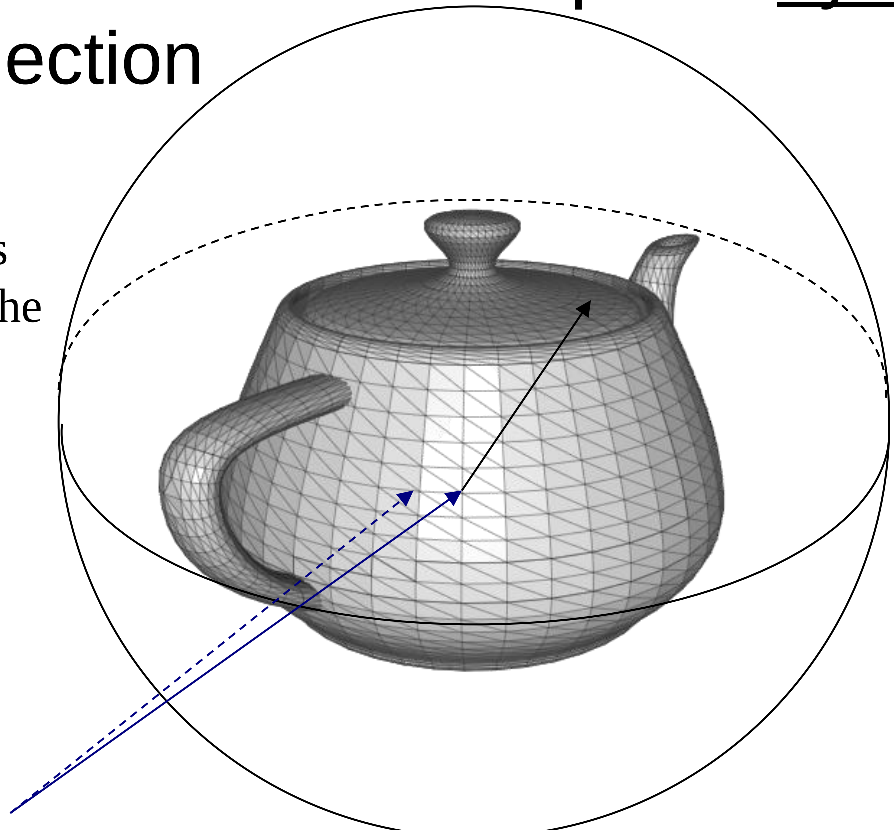


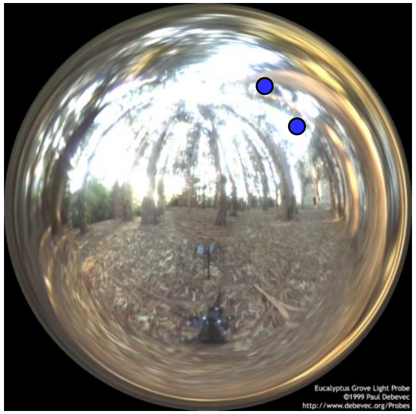


Environment Mapping

- Generate a spherical/cubic map of the environment around the model.
- Texture coordinates are computed <u>dynamically</u> through reflection

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

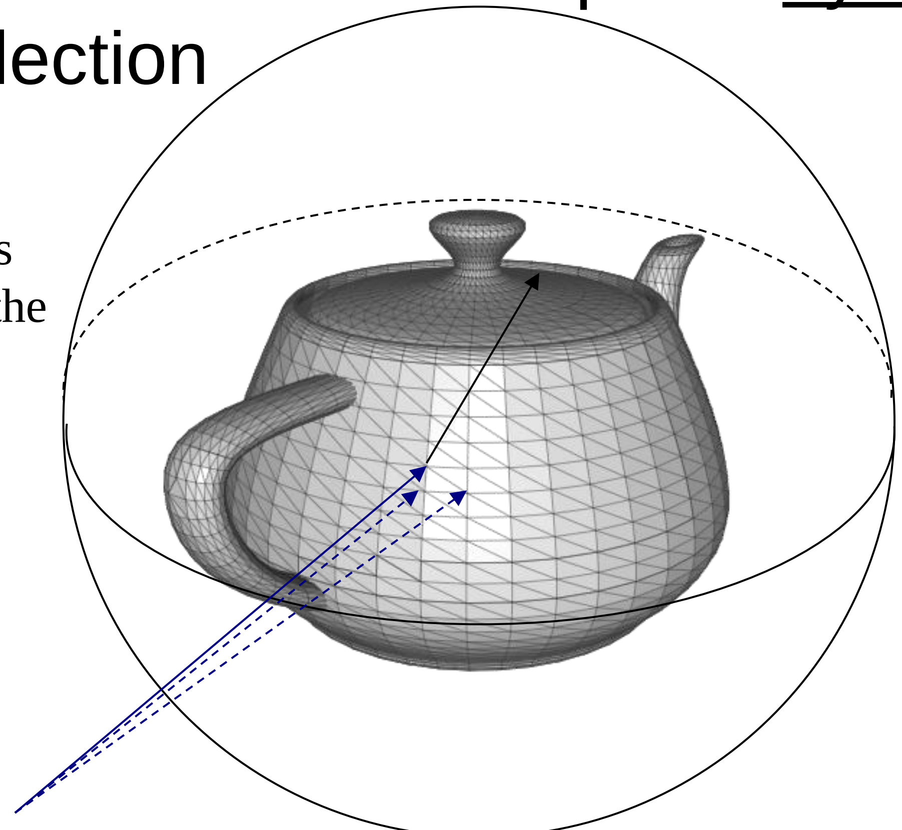


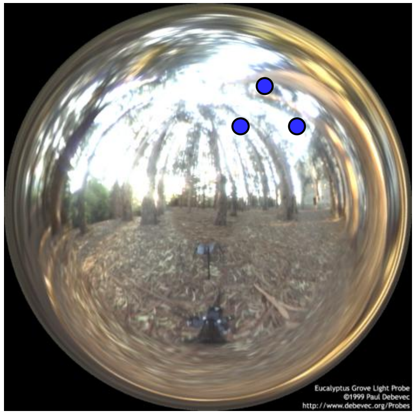


Environment Mapping

- Generate a spherical/cubic map of the environment around the model.
- Texture coordinates are computed <u>dynamically</u> through reflection

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



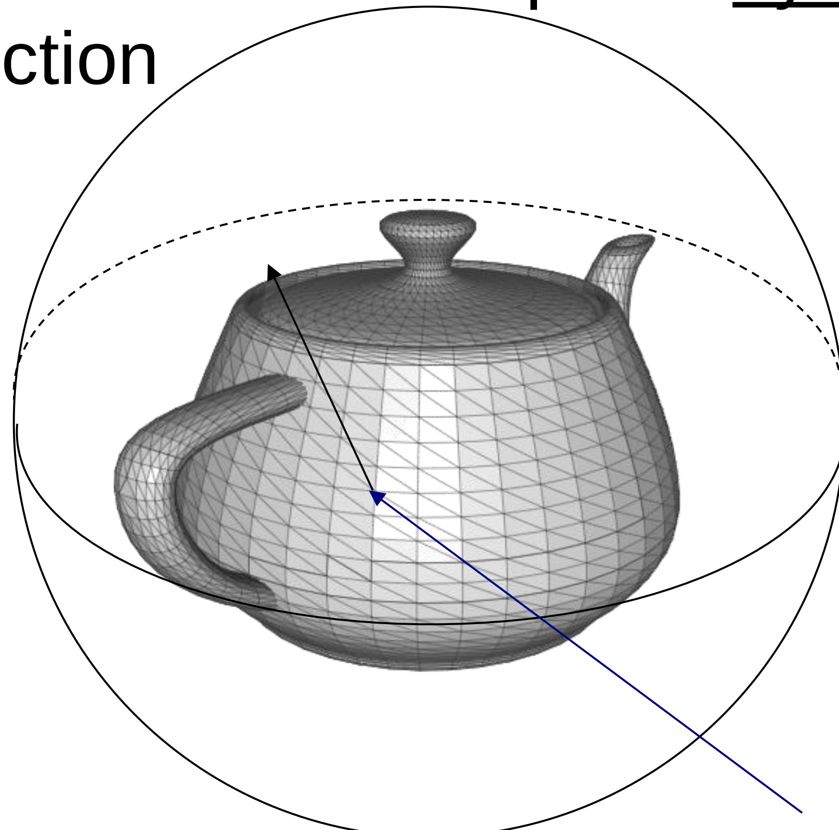


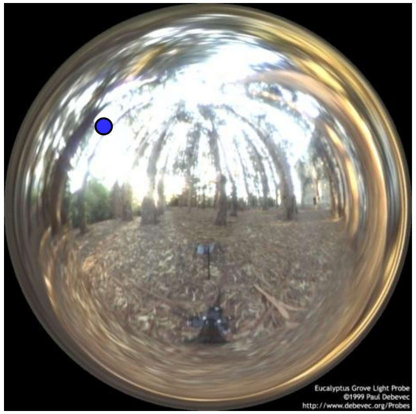
Environment Mapping

- Generate a spherical/cubic map of the environment around the model.
- Texture coordinates are computed <u>dynamically</u> 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.

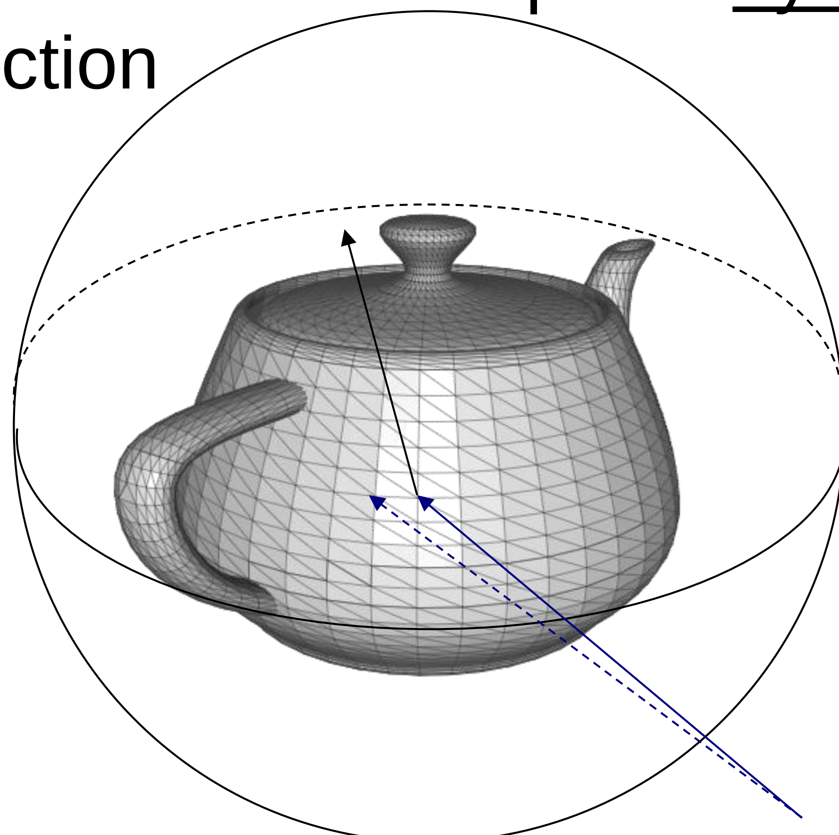




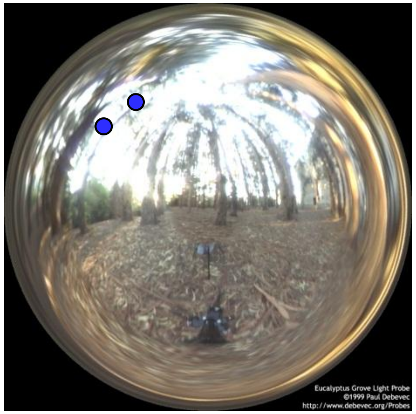
Environment Mapping

- Generate a spherical/cubic map of the environment around the model.
- Texture coordinates are computed <u>dynamically</u> through reflection

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



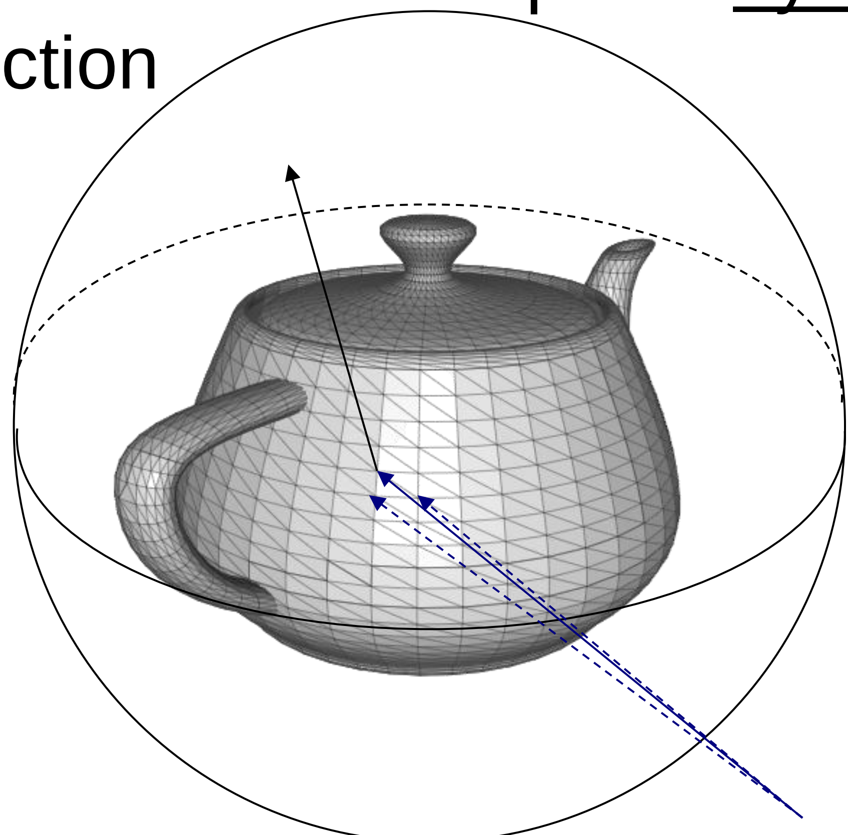
At the same triangle, changing the position of the camera changes the texture coordinates.

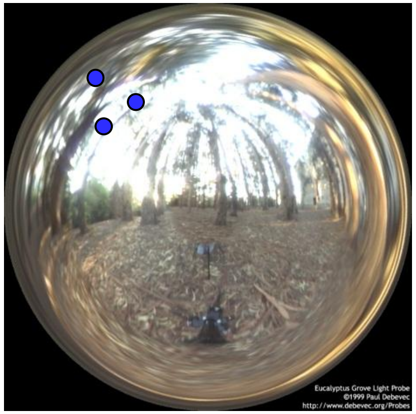


Environment Mapping

- Generate a spherical/cubic map of the environment around the model.
- Texture coordinates are computed <u>dynamically</u> 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.

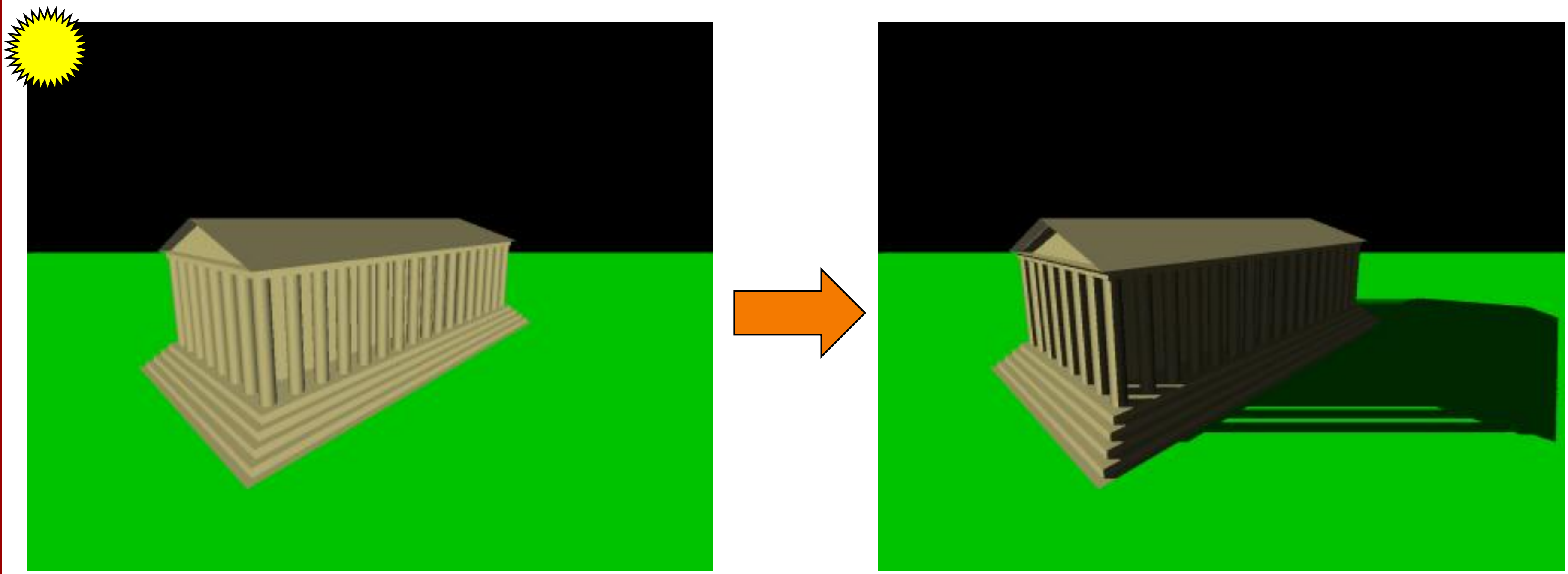
Environment Mapping

Texture coordinates are computed <u>dynamically</u> through reflection of the view direction through the surface normal

P. Debevec

Shadow Mapping (Williams 1978)

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



Images courtesy of https://en.wikipedia.org/wiki/Shadow_mapping

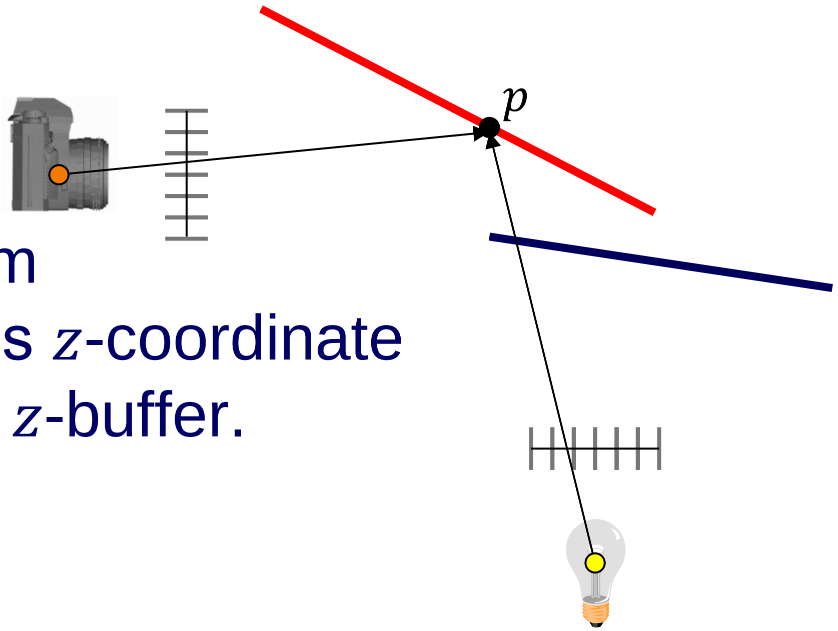
Shadow Mapping (Williams 1978)

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.

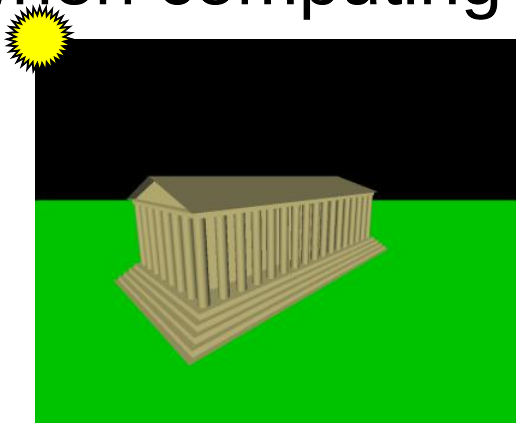
⇒ Rendering the scene from the light's perspective, p's z-coordinate is the value stored in the z-buffer.



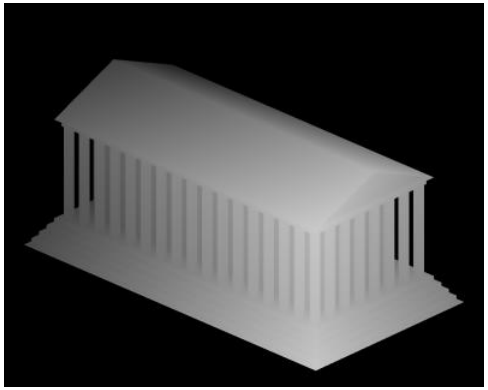
Shadow Mapping (Williams 1978)

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



Camera view



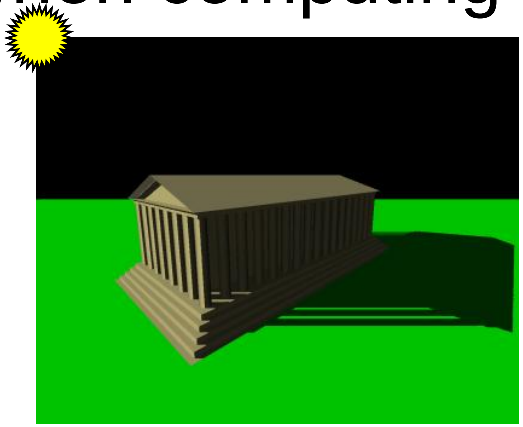
Shadow map

Images courtesy of https://en.wikipedia.org/wiki/Shadow_mapping

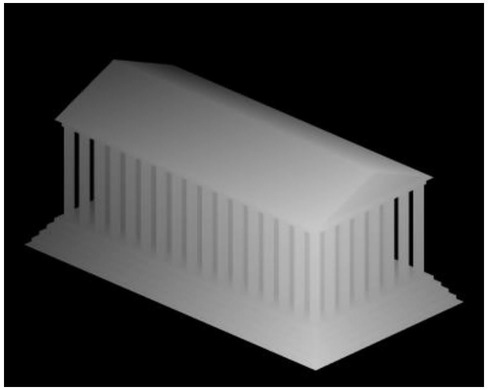
Shadow Mapping (Williams 1978)

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



Camera view



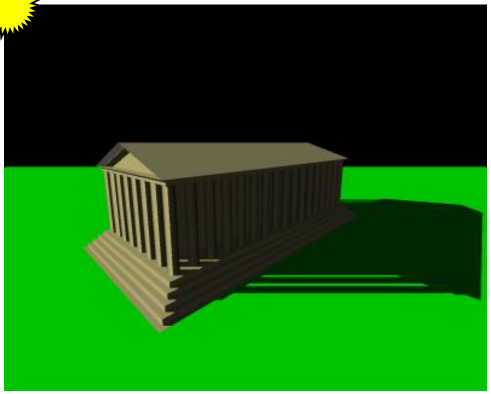
Shadow map

Images courtesy of https://en.wikipedia.org/wiki/Shadow_mapping

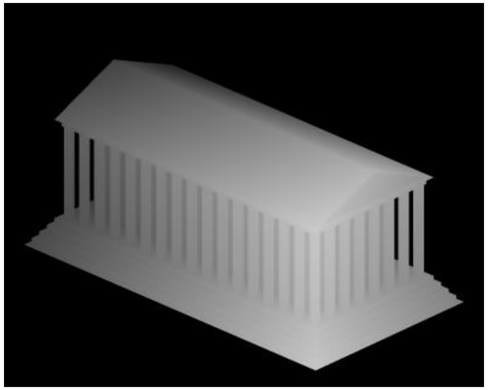
Shadow Mapping (Williams 1978)

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



Camera view



Shadow map

Images courtesy of https://en.wikipedia.org/wiki/Shadow_mapping