

# Textures



Slide 1



Images courtesy, Foley, van Dam, Feiner, Hughes

Lecture 4

Copyright © Amitabh Varshney

# Textures

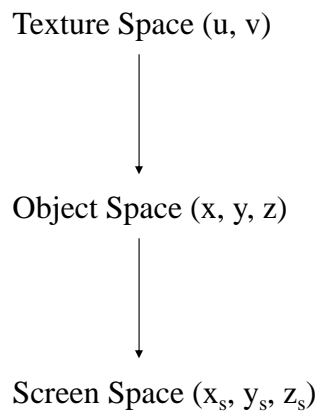
- Introduced by Catmull in 1974
- Simulate Surface Detail / Micro Geometry
- Initially, map images to geometry
- Now generalization to bumps, normals, displacements, environments, volumes, ...

Slide 2

Lecture 4

Copyright © Amitabh Varshney

# Texture Mapping

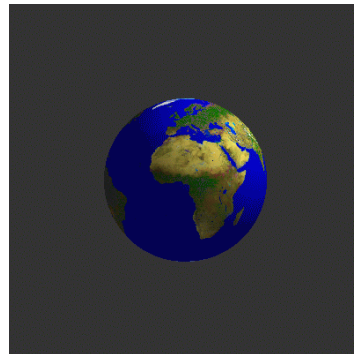
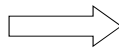


Slide 3

Lecture 4

Copyright © Amitabh Varshney

# Texture Image



Slide 4

Lecture 4

Copyright © Amitabh Varshney

### Texture Image

The image shows a red Coca-Cola Classic can label. The label features the 'Coca-Cola' script logo on the left, 'CLASSIC' in a vertical font, and 'Coke' in large white letters. A nutrition facts table is visible on the right side of the label. The label is placed within a square frame with a horizontal axis labeled 'u' and a vertical axis labeled 'v', both ranging from 0 to 1.

Nutrition Facts	
Serv Size 1 Can	
Amount Per Serving	
Calories 140	
% Daily Value*	
Total Fat 0g	
Sodium 50mg	
Total Carb 35g	
Sugars 39g	
Protein 0g	

\*Percent Daily Values are based on a diet of other people's secrets.

CARBONATED WATER, HIGH FRUCTOSE CORN SYRUP, SUGAR, CARAMEL COLOR, PHOSPHORIC ACID, NATURAL FLAVORS, CAFFEINE.

PLEASE RECYCLE

© 1998 THE COCA-COLA COMPANY

12 FL. OZ. (355 mL)

0 496340 6

Slide 5

Lecture 4

Copyright © Amitabh Varshney

### Procedural Texture

The image shows a black and white checkerboard pattern. The pattern consists of alternating black and white squares arranged in a grid. The grid is plotted on a 2D coordinate system with a horizontal axis labeled 'u' and a vertical axis labeled 'v', both ranging from 0 to 1.

Slide 6

Lecture 4

Copyright © Amitabh Varshney

## Procedural Texture

$Uwave = u \% 2$   
 $= u \& 0x01$

$Vwave = v \% 2$   
 $= v \& 0x01$

$Pix(u, v) = Uwave \oplus Vwave$

If  $u$  and  $v$  are  $n$  bits each,  
 Instead of  $2^{2n}$  bits we only  
 need  $2n$  bits + code

Slide 7
Lecture 4
Copyright © Amitabh Varshney

## Capability Gap

Year	Perf (ps/Inst) Growth	Delay/CPU Growth	Ratio (Perf:Delay)
1980	52%/year	74%/year	-
2000	-	-	30:1
2010	-	-	1,000:1
2015	-	-	30,000:1

Graph courtesy of Bill Dally

Slide 8
Lecture 4
Copyright © Amitabh Varshney

## Texture Mapping

- Back to the problem of mapping label to a Can
- For now, let the Can be represented by a cylinder

Parametric Equation of a Cylinder:

$$x = r \cos \theta, y = r \sin \theta, z = h, 0 \leq h \leq H, 0 \leq \theta \leq 2\pi$$

Let  $u = \theta / 2\pi$ ,  $v = h/H$ ,  $0 \leq u, v \leq 1$

$$x = r \cos 2\pi u, y = r \sin 2\pi u, z = Hv$$

Slide 9

Lecture 4

Copyright © Amitabh Varshney

## Texture Mapping

From the previous equations:

$$u = \tan^{-1}(y/x)$$

$$v = z/H$$

So given object-space  $(x, y, z)$ , we can find  $(u, v)$ .

We then look up color at  $(u, v)$  and place it at  $(x_s, y_s, z_s)$

Slide 10

Lecture 4

Copyright © Amitabh Varshney

## General Texture Mapping

- In general, need a 2D parameterization of a 3D surface
  - Trivial for some objects: sphere, cylinder, cone, cube, ...
  - Difficult for most objects
- Current schemes involve :
  - Two-stage mapping
  - Converting a 3D surface into an atlas of different parameterizations

Slide 11

Lecture 4

Copyright © Amitabh Varshney

## Two-Stage Texture Mapping

- Introduced by Bier and Sloan (1986)
- *S-Map*: Map from 2D texture space to an intermediate 3D surface (cylinder, sphere, cube, ...)

$$T(u, v) \rightarrow S(r, \theta, \phi)$$

- *O-Map*: Map from 3D intermediate space to 3D object:

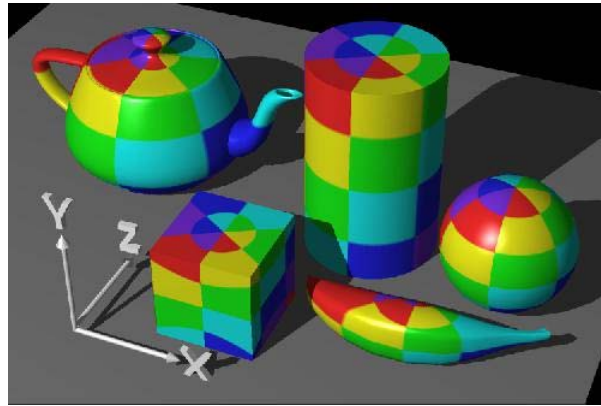
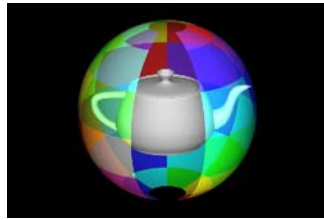
$$S(r, \theta, \phi) \rightarrow O(x, y, z)$$

Slide 12

Lecture 4

Copyright © Amitabh Varshney

# Spherical Mapping



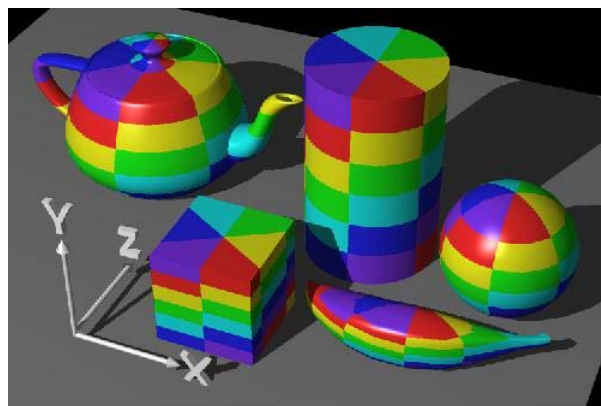
Images courtesy, David Ebert, Purdue

Slide 13

Lecture 4

Copyright © Amitabh Varshney

# Cylindrical Mapping



Images courtesy, David Ebert, Purdue

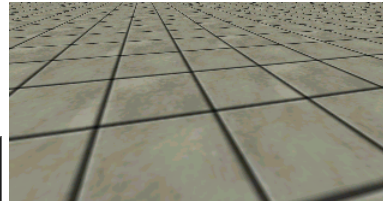
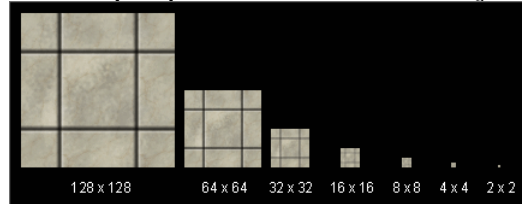
Slide 14

Lecture 4

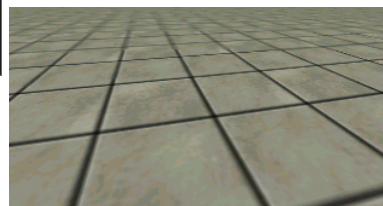
Copyright © Amitabh Varshney

# Pre-Filtering

- What if one screen-space pixel corresponds to multiple texture image pixels (texels)?
- Do successive averaging to build up a powers of 2 hierarchy



Without MIP-mapping



With MIP-mapping

(MIP-map) and index at the right scale factor into this hierarchy.

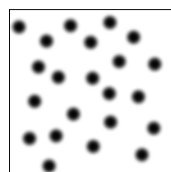
Slide 15

Lecture 4

Copyright © Amitabh Varshney

# Bump Mapping

- Introduced by Blinn (1978)



Images courtesy, Alan Watt, *3D Computer Graphics*

Slide 16

Lecture 4

Copyright © Amitabh Varshney



# Procedural Bump Mapping



Images courtesy, Alan Watt, *3D Computer Graphics*

Slide 17

Lecture 4

Copyright © Amitabh Varshney

# Bump & Texture Mapping



Images courtesy, Alan Watt, *3D Computer Graphics*

Slide 18

Lecture 4

Copyright © Amitabh Varshney

## Bump Mapping

- Consider a surface  $P(u, v)$
- Its normal is given by:  $N = P_u \times P_v$
- Perturb the surface by a bump map function  $B(u, v)$  along its normals:  $Q(u, v) = P(u, v) + B(u, v)N$
- Now compute and use the perturbed normal for illumination :

$$N' = Q_u \times Q_v = N + B_u N \times P_v - B_v N \times P_u$$

Slide 19

Lecture 4

Copyright © Amitabh Varshney

## Normal Mapping

- Similar to bump mapping, except normals are directly specified instead of the bump map function
- Similar generalizations exist for Phong shading using texture mapping

Slide 20

Lecture 4

Copyright © Amitabh Varshney

## Environment Mapping

- Introduced by Blinn & Newell (1976) as *reflection mapping*
- Compute the reflection vector at each mesh vertex:
 
$$\mathbf{R} = 2(\mathbf{N} \cdot \mathbf{V})\mathbf{N} - \mathbf{V}$$
- Interpolate the reflection vector across the triangle
- Modern GPUs provide hardware *cube-maps* for this
- Use the reflection vectors to index into the environment (texture) map

Slide 21

Lecture 4

Copyright © Amitabh Varshney

## Environment Mapping

- Index into texture map not by (u, v) but by reflection rays



Image by Jim Blinn via debevec.org

Slide 22



Image by Aravind Kalaiah &amp; Amitabh Varshney

Lecture 4

Copyright © Amitabh Varshney

# Displacement Mapping

- In contrast to Bump Mapping, the actual geometry is displaced here
- The surface as well as silhouettes appear non-smooth
- Presents a challenge for deferred shading systems (when shading is performed after visibility)

Image courtesy SIGGRAPH Education Materials  
Rendered by Pixar Renderman



Slide 23

# Displacement Mapping

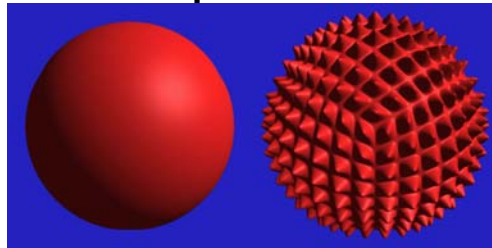


Image by Michael Capps (NPS)



Slide 24

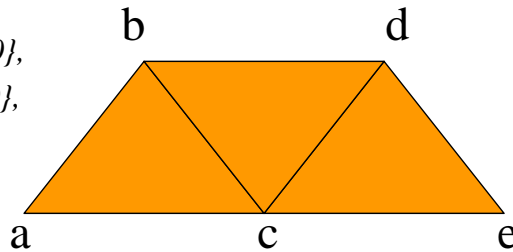
Lecture 4

Image by Wang *et al.*  
(SIGGRAPH 2003)

Copyright © Amitabh Varshney

## Direct Mesh Representation

Let  $a = \{0, 0, 0\}$ ,  $b = \{1, 1, 0\}$ ,  
 $c = \{2, 0, 0\}$ ,  $d = \{3, 1, 0\}$ ,  
 $e = \{4, 0, 0\}$



### ***Simplest File Format***

```
t 3 // # triangles = 3
0, 0, 0, 1, 1, 0, 2, 0, 0 // abc
1, 1, 0, 2, 0, 0, 3, 1, 0 // bcd
2, 0, 0, 3, 1, 0, 4, 0, 0 // cde
```

Slide 25

Lecture 4

Copyright © Amitabh Varshney

## Direct Mesh Representation

### ***More Generally:***

```
f n // # faces = n
i1 x11 y11 z11 x12 y12 z12 ... x1i1 y1i1 z1i1 // first face has i1 vertices
i2 x21 y21 z21 x22 y22 z22 ... x2i2 y2i2 z2i2 // second face has i2 vertices
...
...
...
in xn1 yn1 zn1 xn2 yn2 zn2 ... xnin ynin znin // nth face has in vertices
```

### ***For vertices with colors, normals etc:***

```
i1 x11 y11 z11 r11 g11 b11 nx11 ny11 nz11
x12 y12 z12 r12 g12 b12 nx12 ny12 nz12 ...
x1i1 y1i1 z1i1 r1i1 g1i1 b1i1 nx1i1 ny1i1 nz1i1
```

Slide 26

Lecture 4

Copyright © Amitabh Varshney

## Indexed Mesh Representation

```

v 5 // #vertices = 5
0 0 0
1 1 0
2 0 0
3 1 0
4 0 0
t 3 // #triangles = 3
0 1 2 // abc
1 2 3 // bcd
2 3 4 // cde
        
```

Diagram showing a mesh with 5 vertices (a, b, c, d, e) and 3 triangles (abc, bcd, cde). The mesh is colored orange.

Slide 27
Lecture 4
Copyright © Amitabh Varshney

## Indexed Mesh Representation

```

v 5 // #vertices = 5
0 0 0
1 1 0
2 0 0
3 1 0
4 0 0
c 2 // # colors = 2
255 0 0 // red in byte rep
0 255 0 // green in byte rep
t 3 // #triangles = 3
0 0 1 0 2 0 // abc
1 0 2 0 3 0 // bcd
2 1 3 1 4 1 // cde
        
```

Diagram showing a mesh with 5 vertices (a, b, c, d, e) and 3 triangles (abc, bcd, cde). The mesh is colored red and green.

Slide 28
Lecture 4
Copyright © Amitabh Varshney

# Indexed Mesh Representation

```

v 5 // #vertices = 5
0 0 0
1 1 0
2 0 0
3 1 0
4 0 0
c 2 // # colors = 2
255 0 0 // red in byte rep
0 255 0 // green in byte rep
t 3 // # triangles = 3
0 0 1 0 2 0 // abc
1 0 2 0 3 0 // bcd
2 1 3 1 4 1 // cde

```

- File format consists of a *Vertex Array*, *Color Array*, *Normal Array*, ... and a *Face Array* with indices pointing to the above arrays.

- Indexed face sets used in numerous file formats including VRML/X3D, Wavefront OBJ, ...

Slide 29

Lecture 4

Copyright © Amitabh Varshney