

Computer Graphics

CSE 167 [Win 23], Lecture 18: Texture Mapping

Ravi Ramamoorthi

<http://viscomp.ucsd.edu/classes/cse167/wi23>

Many slides from Greg Humphreys, formerly UVA and Rosalee Wolfe, DePaul tutorial teaching texture mapping visually
Chapter 11 in text book covers some portions

1

To Do

- Submit HW4 milestone by tomorrow
- Prepare for final push on HW 4
- Written assignment due Mar 15, 11:59pm
 - Individually, no collaboration, piazza posts
 - Open notes, online. But understand what you turn in, and try to use your own words
 - No final

2

Texture Mapping

- Important topic: nearly all objects textured
 - Wood grain, faces, bricks and so on
 - Adds visual detail to scenes
- Meant as a fun and practically useful lecture



Polygonal model

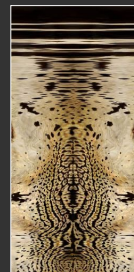


With surface texture

3

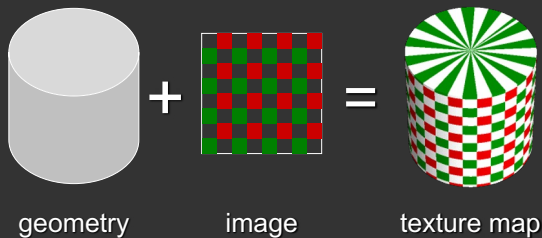
Adding Visual Detail

- Basic idea: use images instead of more polygons to represent fine scale color variation



4

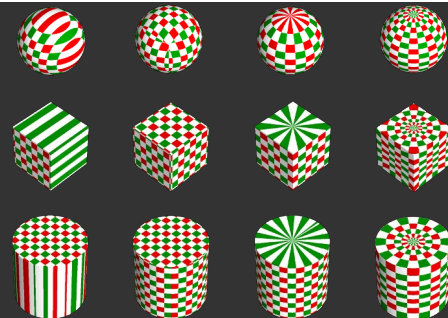
Parameterization



- Q: How do we decide *where* on the geometry each color from the image should go?

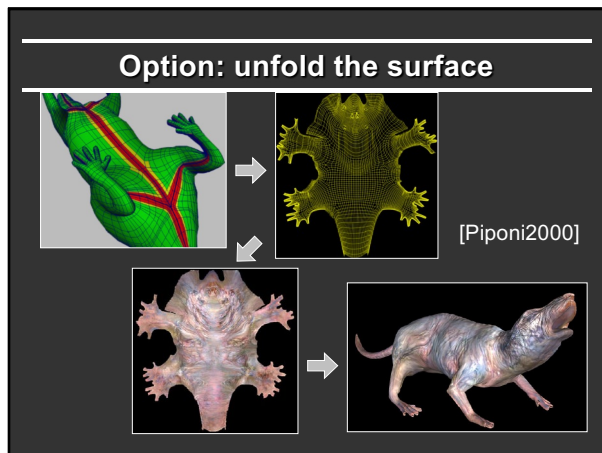
5

Option: Varieties of projections

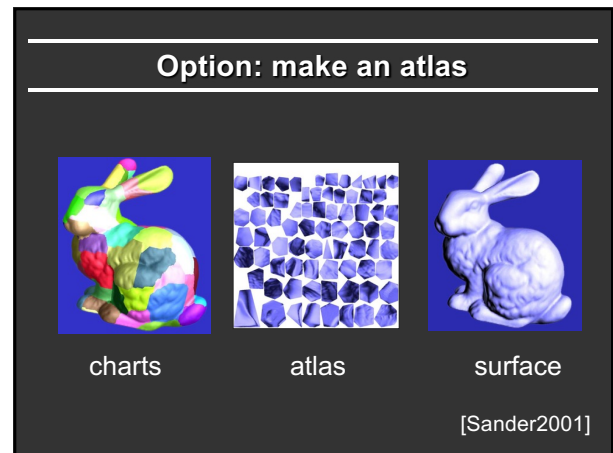


[Paul Bourke]

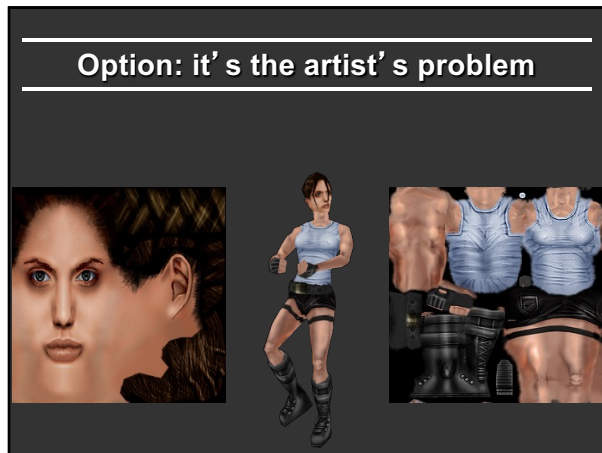
6



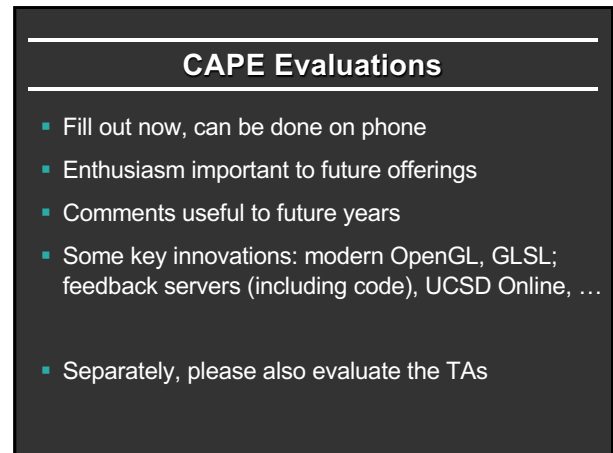
7



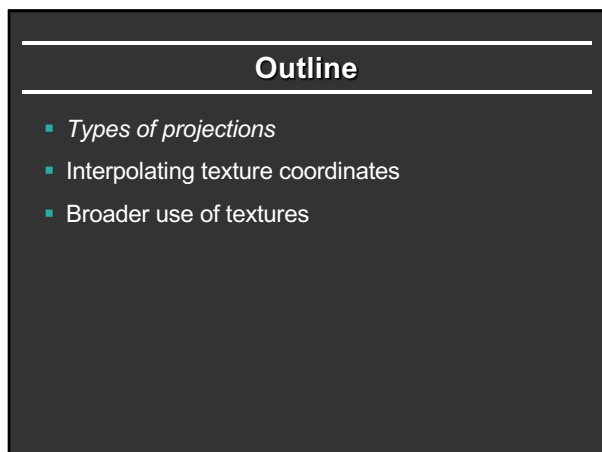
8



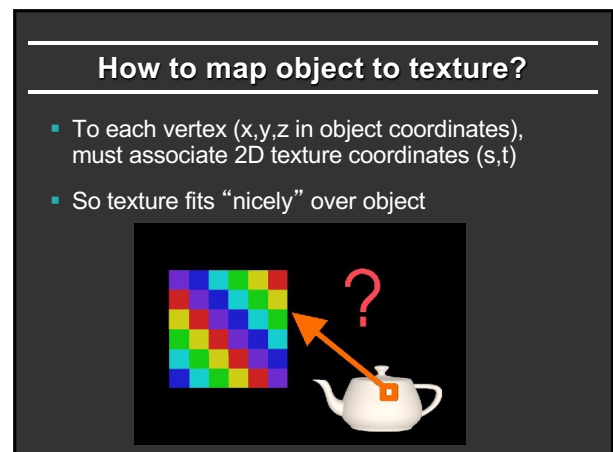
9



10



11



12

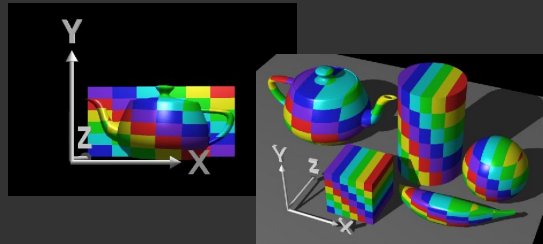
Idea: Use Map Shape

- Map shapes correspond to various projections
 - Planar, Cylindrical, Spherical
- First, map (square) texture to basic map shape
- Then, map basic map shape to object
 - Or vice versa: Object to map shape, map shape to square
- Usually, this is straightforward
 - Maps from square to cylinder, plane, sphere well defined
 - Maps from object to these are simply spherical, cylindrical, cartesian coordinate systems

13

Planar mapping

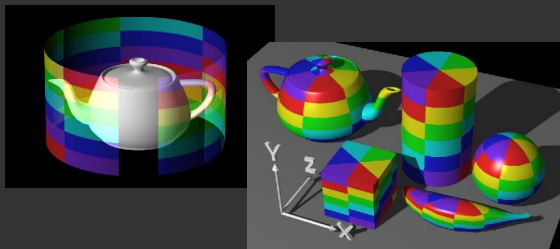
- Like projections, drop z coord $(s,t) = (x,y)$
- Problems: what happens near $z = 0$?



14

Cylindrical Mapping

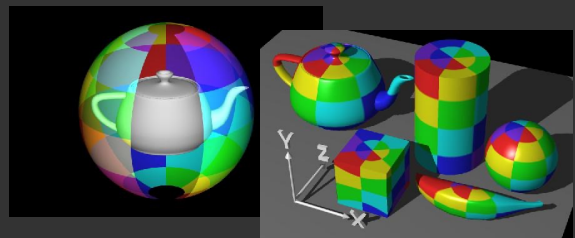
- Cylinder: r, θ, z with $(s,t) = (\theta/(2\pi), z)$
 - Note seams when wrapping around ($\theta = 0$ or 2π)



15

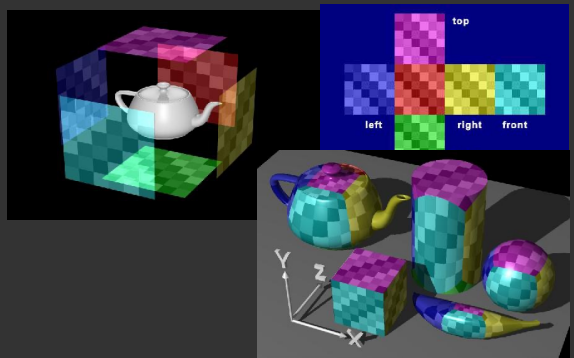
Spherical Mapping

- Convert to spherical coordinates: use latitude/long.
 - Singularities at north and south poles



16

Cube Mapping



17

Cube Mapping



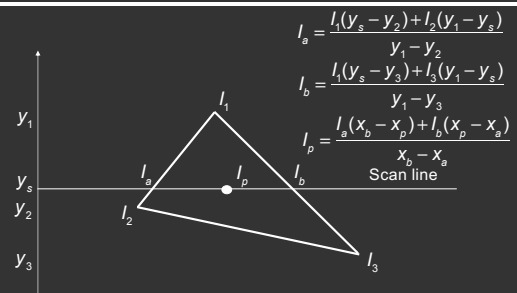
18

Outline

- Types of projections
- Interpolating texture coordinates
- Broader use of textures

19

Gouraud Shading – Details



Actual implementation efficient: difference equations while scan converting

20

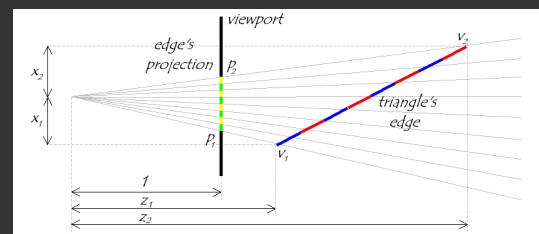
Artifacts

- [Wikipedia page](https://en.wikipedia.org/wiki/File:Perspective_correct_texture_mapping.jpg)
https://en.wikipedia.org/wiki/File:Perspective_correct_texture_mapping.jpg
- What artifacts do you see?
- Why?
- Why not in standard Gouraud shading?
- Hint: problem is in interpolating parameters

21

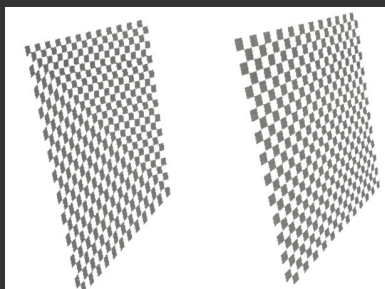
Interpolating Parameters

- The problem turns out to be fundamental to interpolating parameters in screen-space
 - Uniform steps in screen space \neq uniform steps in world space



22

Texture Mapping



Linear interpolation of texture coordinates Correct interpolation with perspective divide

Hill Figure 8.42

23

Interpolating Parameters

- Perspective foreshortening is not getting applied to our interpolated parameters
 - Parameters should be compressed with distance
 - Linearly interpolating them in screen-space doesn't do this

24

Perspective-Correct Interpolation

- Skipping a bit of math to make a long story short...
 - Rather than interpolating u and v directly, interpolate u/z and v/z
 - These do interpolate correctly in screen space
 - Also need to interpolate Z and multiply per-pixel
 - Problem: we don't know z anymore
 - Solution: we do know $w \sim 1/z$
 - So...interpolate uw and vw and w , and compute $u = uw/w$ and $v = vw/w$ for each pixel
 - This unfortunately involves a divide per pixel
- [Wikipedia page](#)

25

Texture Map Filtering

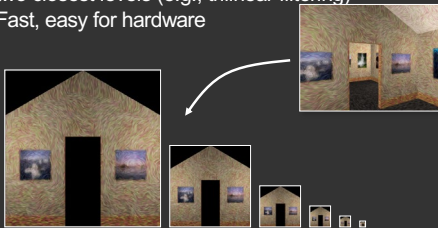
- Naive texture mapping aliases badly
- Look familiar?


```
int uval = (int) (u * denom + 0.5f);
int vval = (int) (v * denom + 0.5f);
int pix = texture.getPixel(uval, vval);
```
- Actually, each pixel maps to a region in texture
 - $|PIX| < |TEX|$
 - Easy: interpolate (bilinear) between texel values
 - $|PIX| > |TEX|$
 - Hard: average the contribution from multiple texels
 - $|PIX| \sim |TEX|$
 - Still need interpolation!

26

Mip Maps

- Keep textures prefiltered at multiple resolutions
 - For each pixel, linearly interpolate between two closest levels (e.g., trilinear filtering)
 - Fast, easy for hardware



- Why "Mip" maps?

27

MIP-map Example

- No filtering:



- MIP-map texturing:



28

Outline

- Types of projections
- Interpolating texture coordinates
- *Broader use of textures*

29

Texture Mapping Applications

- Modulation, light maps
- Bump mapping
- Displacement mapping
- Illumination or Environment Mapping
- Procedural texturing
- And many more

30

Modulation textures

Map texture values to scale factor

Wood texture

Texture value

$$I = T(s, t)(I_E + K_A I_A + \sum_L (K_D(N \cdot L) + K_S(V \cdot R)^n) S_L I_L + K_T I_T + K_S I_S)$$

31

Bump Mapping

- Texture = change in surface normal!

Sphere w/ diffuse texture Swirly bump map Sphere w/ diffuse texture and swirly bump map

32

Displacement Mapping

33

Illumination Maps

- Quake introduced *illumination maps* or *light maps* to capture lighting effects in video games

Texture map:

Light map

Texture map + light map:

34

Environment Maps

Images from *Illumination and Reflection Maps: Simulated Objects in Simulated and Real Environments*
 Gene Miller and C. Robert Hoffman
 SIGGRAPH 1984 "Advanced Computer Graphics Animation" Course Notes

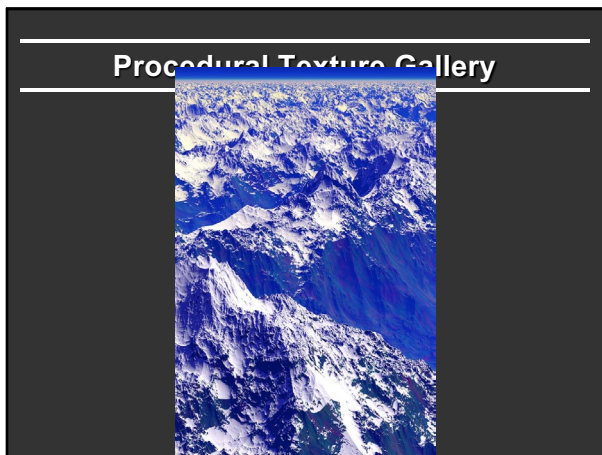
35

Solid textures

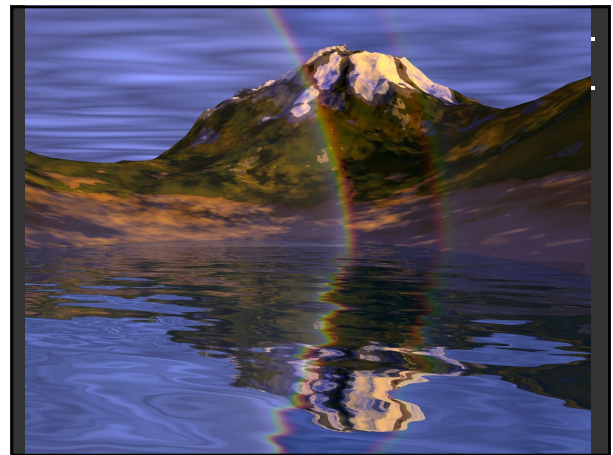
Texture values indexed by 3D location (x,y,z)

- Expensive storage, or
- Compute on the fly, e.g. Perlin noise →

36



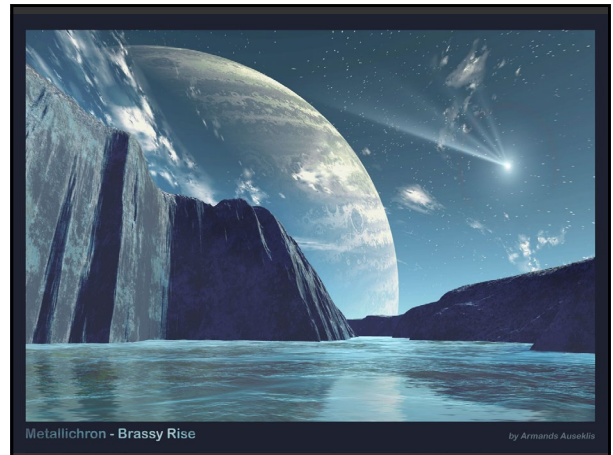
37



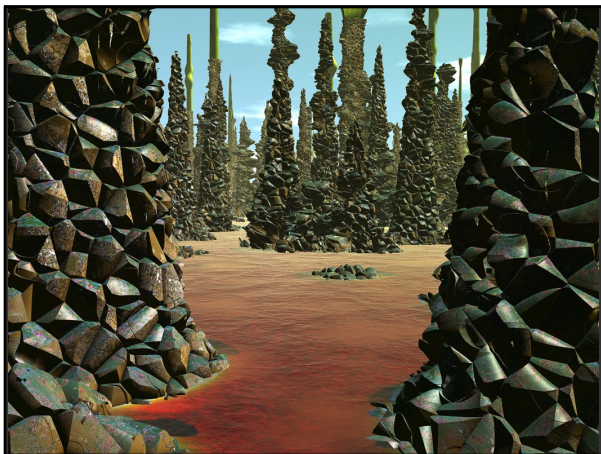
38



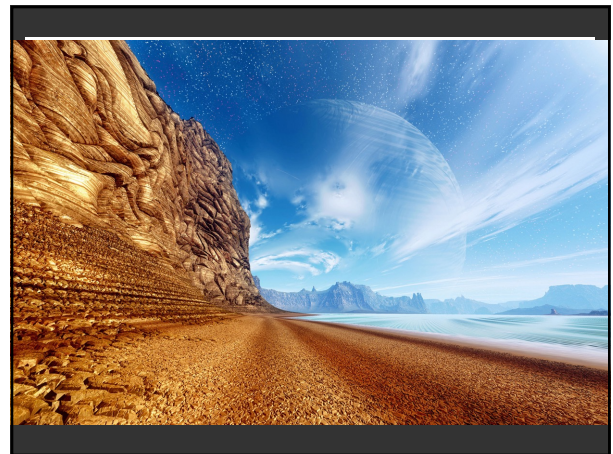
39



40



41



42