Computer Graphics

CSE 167 [Win 17], Lecture 18: Texture Mapping

Ravi Ramamoorthi

http://viscomp.ucsd.edu/classes/cse167/wi17

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

To Do

- Prepare for final push on HW 4
- We may have a brief written assignment

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

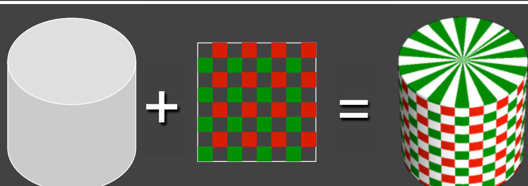
Polygonal model

With surface texture

Adding Visual Detail

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

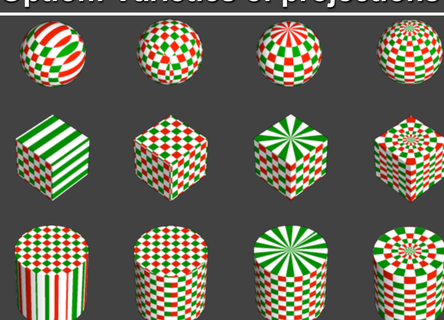
Parameterization



geometry | image | texture map

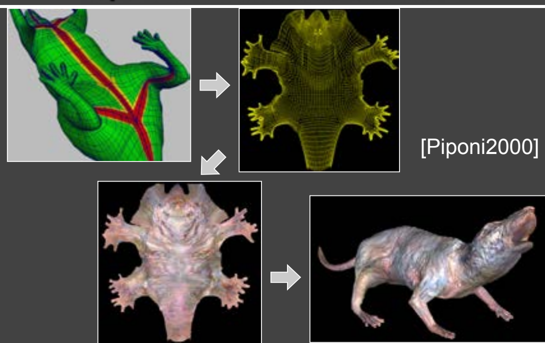
- Q: How do we decide *where* on the geometry each color from the image should go?

Option: Varieties of projections



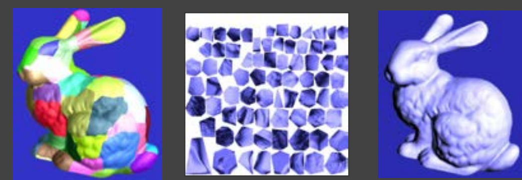
[Paul Bourke]

Option: unfold the surface



[Piponi2000]

Option: make an atlas



charts atlas surface

[Sander2001]

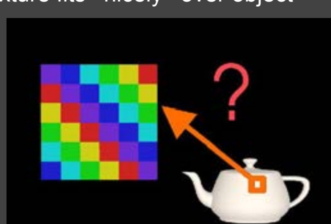
Option: it's the artist's problem

Outline

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

How to map object to texture?

- To each vertex (x,y,z in object coordinates), must associate 2D texture coordinates (s,t)
- So texture fits "nicely" over object

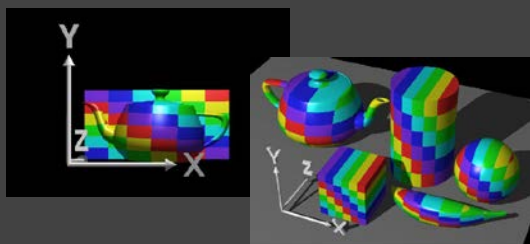


Idea: Use Map Shape

- 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

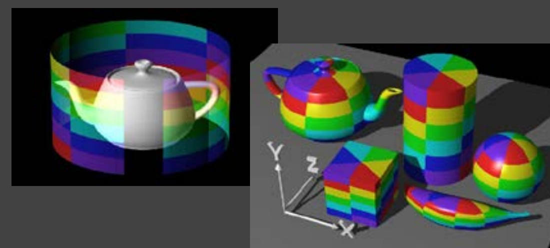
Planar mapping

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



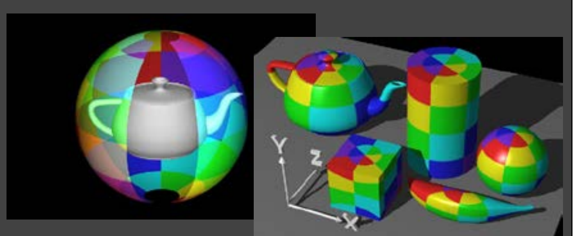
Cylindrical Mapping

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

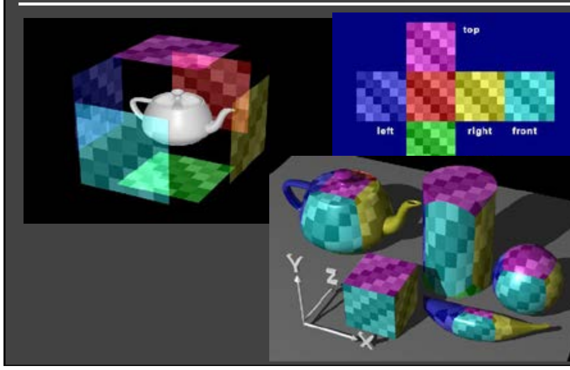


Spherical Mapping

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



Cube Mapping



Cube Mapping

Outline

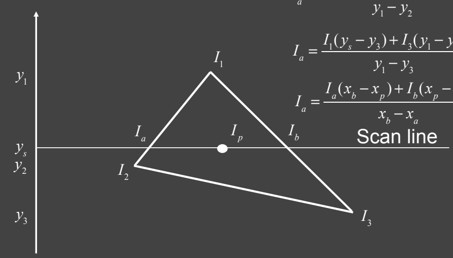
- Types of projections
- *Interpolating texture coordinates*
- Broader use of textures

1st idea: Gouraud interp. of texcoords

$$I_a = \frac{I_1(y_s - y_2) + I_2(y_1 - y_s)}{y_1 - y_2}$$

$$I_a = \frac{I_1(y_s - y_3) + I_3(y_1 - y_s)}{y_1 - y_3}$$

$$I_a = \frac{I_a(x_b - x_p) + I_b(x_p - x_a)}{x_b - x_a}$$



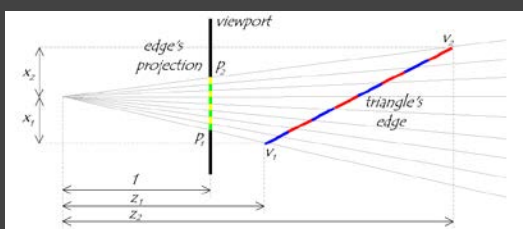
Actual implementation efficient: difference equations while scan converting

Artifacts

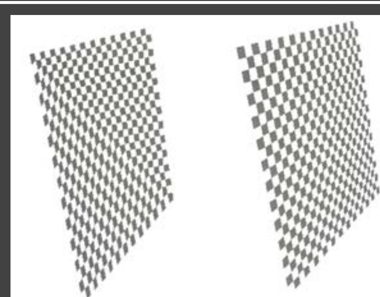
- Wikipedia page
- What artifacts do you see?
- Why?
- Why not in standard Gouraud shading?
- Hint: problem is in interpolating parameters

Interpolating Parameters

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



Texture Mapping



Linear interpolation of texture coordinates

Correct interpolation with perspective divide

Hill Figure 8.42

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

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 ~ 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

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| ~ |TEX|
    - Still need interpolation!

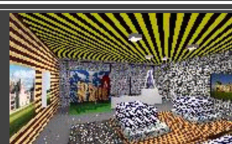
Mip Maps

- 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?

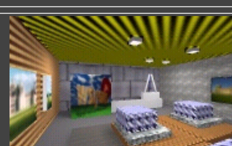
MIP-map Example

- No filtering:



AAAAAAAGH
MY EYES ARE BURNING

- MIP-map texturing:



Where are my glasses?

Outline

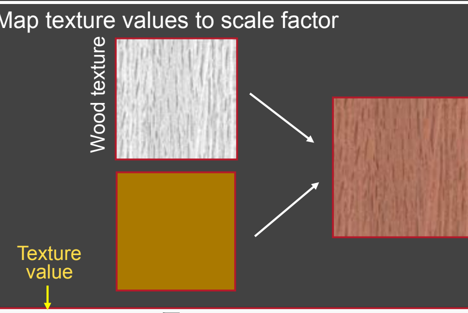
- Types of projections
- Interpolating texture coordinates
- *Broader use of textures*

Texture Mapping Applications

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

Modulation textures

Map texture values to scale factor



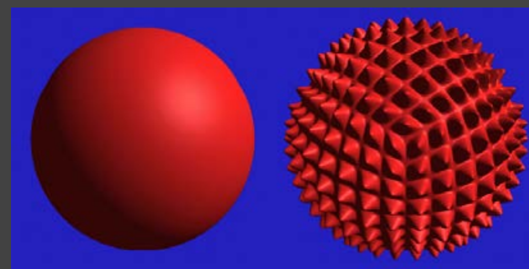
$$I = T(s,t)(I_E + K_A I_A + \sum_L (K_D(N \bullet L) + K_S(V \bullet R)^n) S_L I_L + K_T I_T + K_S I_S)$$

Bump Mapping

- Texture = change in surface normal!

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

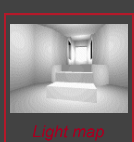
Displacement Mapping

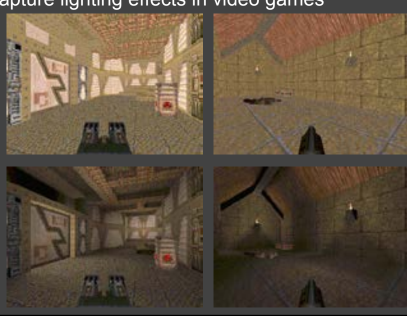


Illumination Maps

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

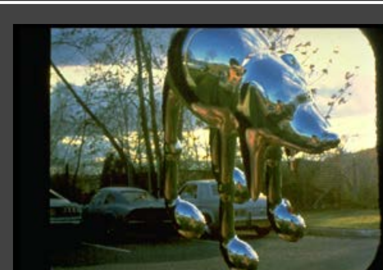
Texture map:





Texture map
+ light map:

Environment Maps



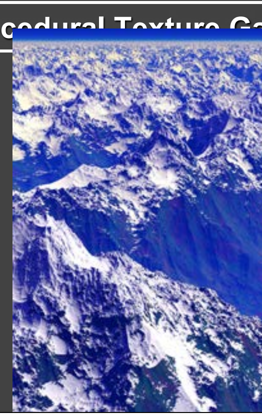
Images from *Illumination and Reflection Maps: Simulated Objects in Simulated and Real Environments*
Gene Miller and C. Robert Hoffman
SIGGRAPH 1984 "Advanced Computer Graphics Animation" Course Notes

Solid textures

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

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

Procedural Texture Gallery




Metallichron - Brassy Rise
by Armands Auseklis