CS559: Computer Graphics

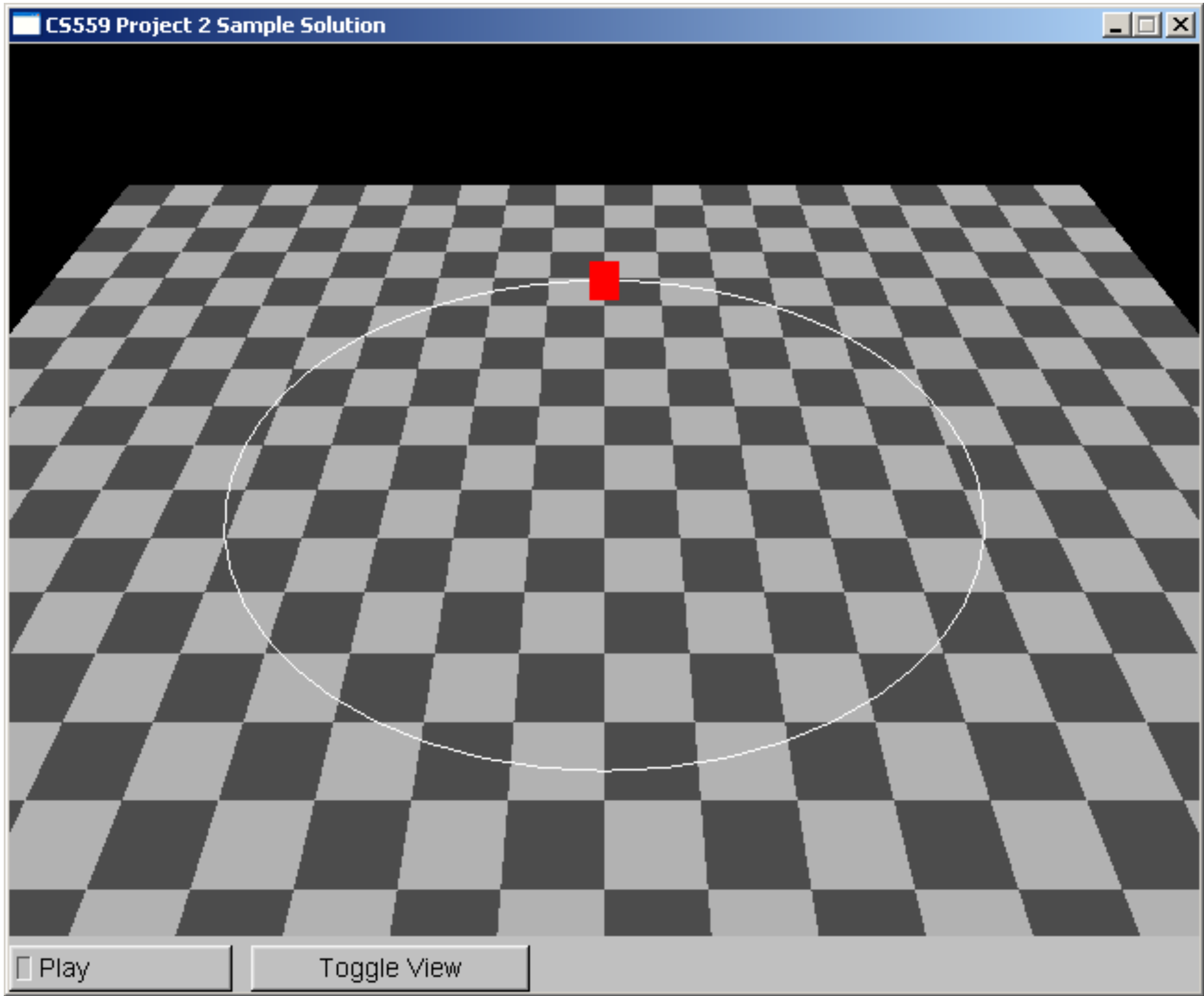
Lecture 20: Project 2 Review and Texture mapping

Li Zhang

Spring 2010

Many slides from Ravi Ramamoorthi, Columbia Univ, Greg Humphreys, UVA and Rosalee Wolfe, DePaul tutorial teaching texture mapping visually

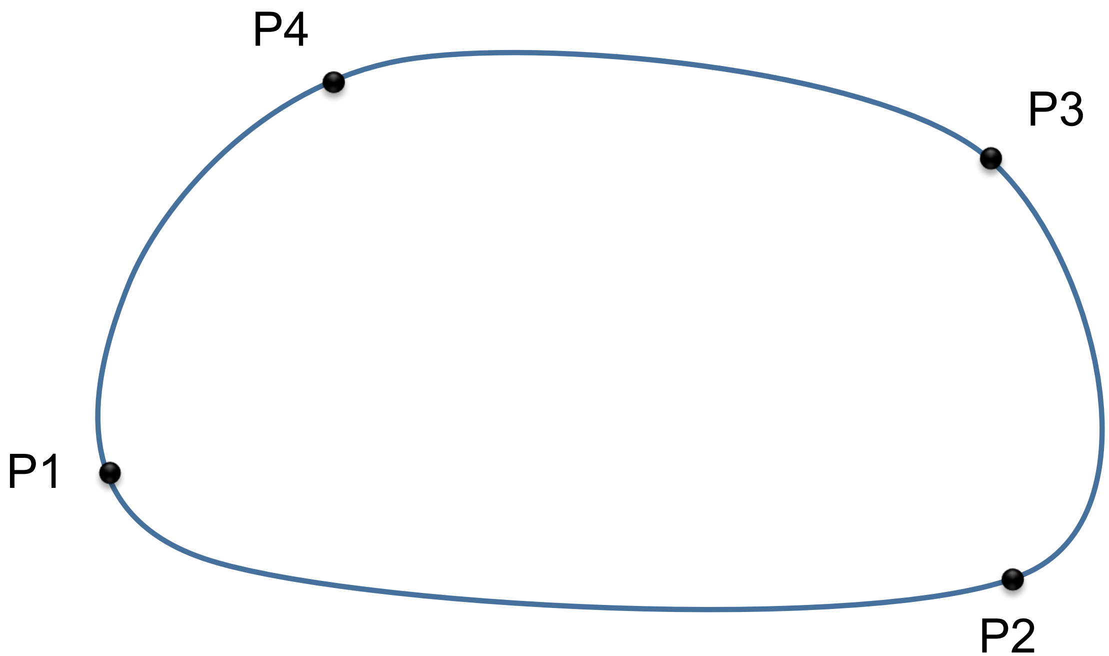
Sample Code by Chi Man Liu



Other basic features

- How to change speed?
  - Add a slider with a callback to change timePerFrame
- How to orient a car?
- How to draw trees?
  - A triangle + a rectangle
  - A cone + a cylinder
    - Using gluQuadrics

Piecewise Cubics



Piecewise Cubics

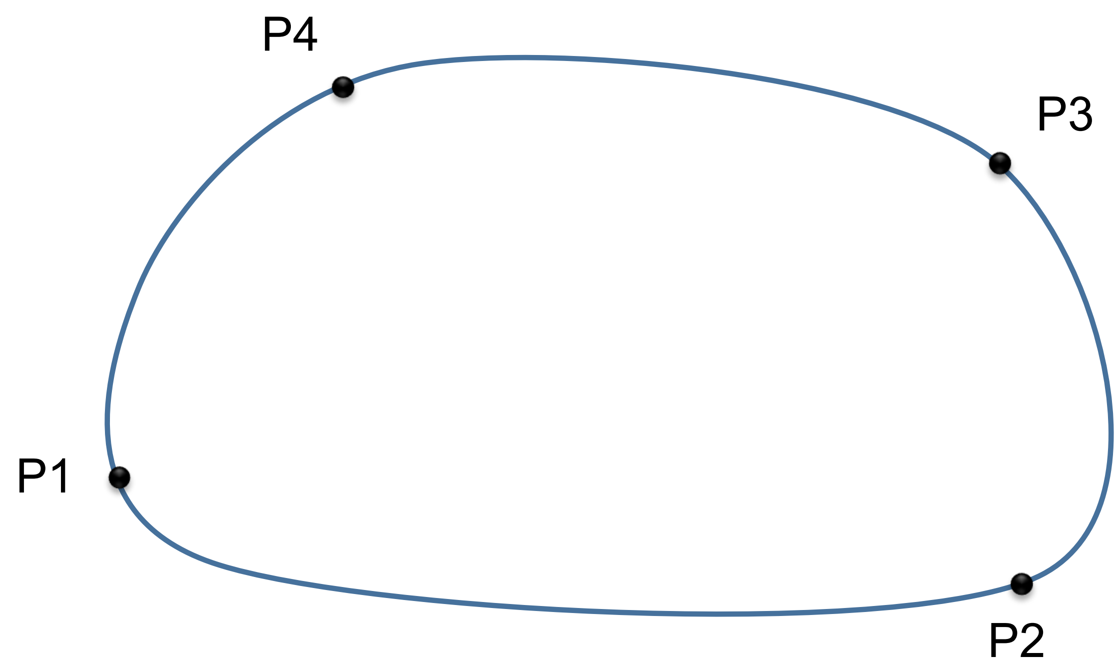
```
Task1: draw the whole curve
DrawCubic(P4,P1,P2,P3);
DrawCubic(P1,P2,P3,P4);
DrawCubic(P2,P3,P4,P1);
DrawCubic(P3,P4,P1,P2);

DrawCubic(Q1,Q2,Q3,Q4)
For (t=0; t < 0.1; t+=0.01)
{
A=computePoint(Q1,Q2,Q3,Q4,t);
B=computePoint(Q1,Q2,Q3,Q4,t+0.01);
drawLine(A,B);
}
```

Fine with Hermite, Catmull-Ron, Cardinal
How about Bezier?

```
Task 2: find where the train is at time t:
If (0<=t<1) return computePoint(P4,P1,P2,P3,t)
If (1<=t<2) return computePoint(P1,P2,P3,P4,t-1);
If (2<=t<3) return computePoint(P2,P3,P4,P1,t-2);
If (3<=t<4) return computePoint(P3,P4,P1,P2,t-3);
```

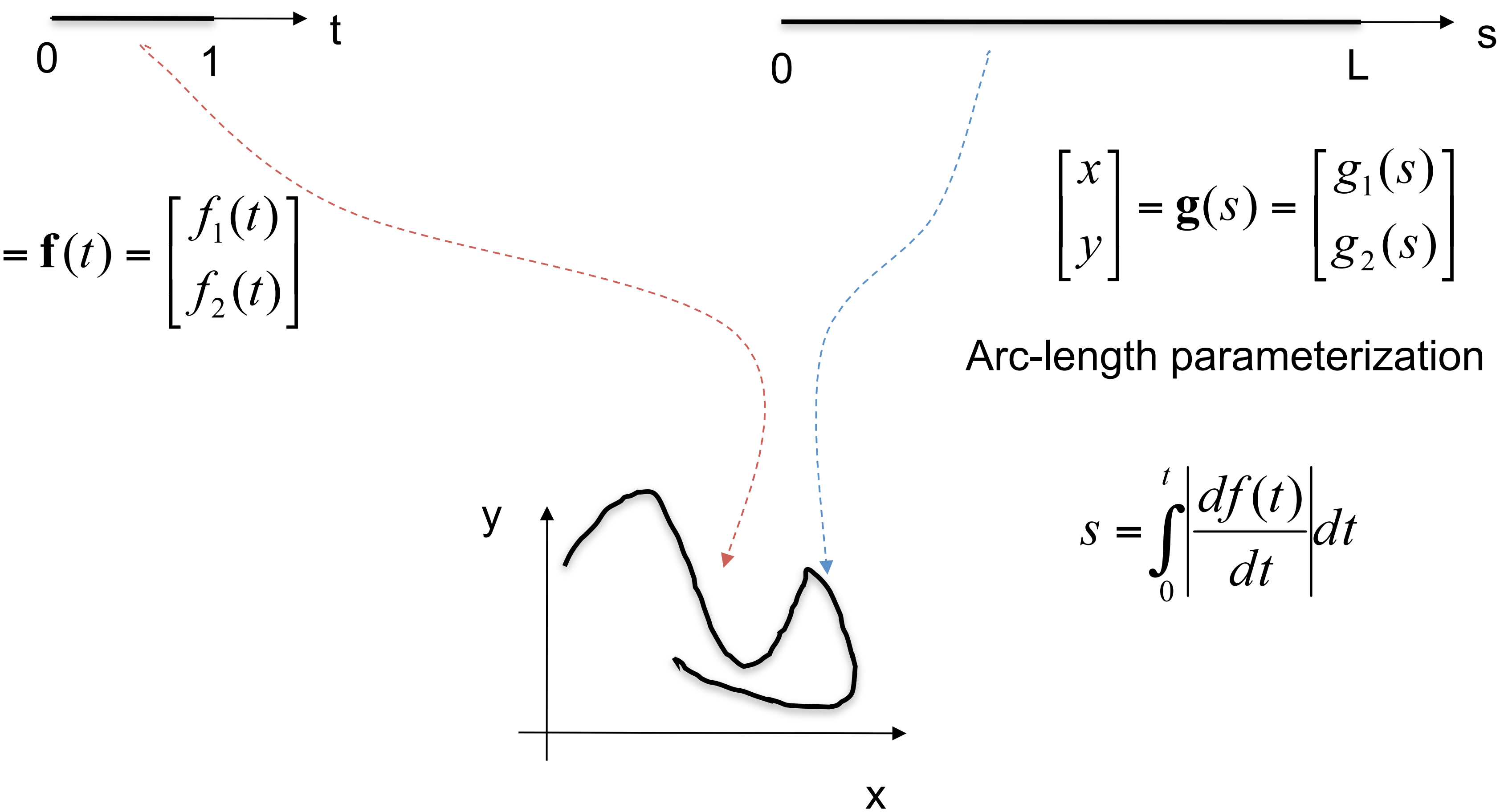
Arc-Length Parameterization

- Arbitrary curves?



$$s = \int_0^t \left| \frac{d\mathbf{f}(u)}{du} \right| du$$

Arc-length parameterization



$$\begin{bmatrix} x \\ y \end{bmatrix} = \mathbf{f}(t) = \begin{bmatrix} f_1(t) \\ f_2(t) \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \end{bmatrix} = \mathbf{g}(s) = \begin{bmatrix} g_1(s) \\ g_2(s) \end{bmatrix}$$

Arc-length parameterization

$$s = \int_0^t \left| \frac{df(t)}{dt} \right| dt$$

Freeform

Correct Orientation in 3D

- Define and interpolate up vector

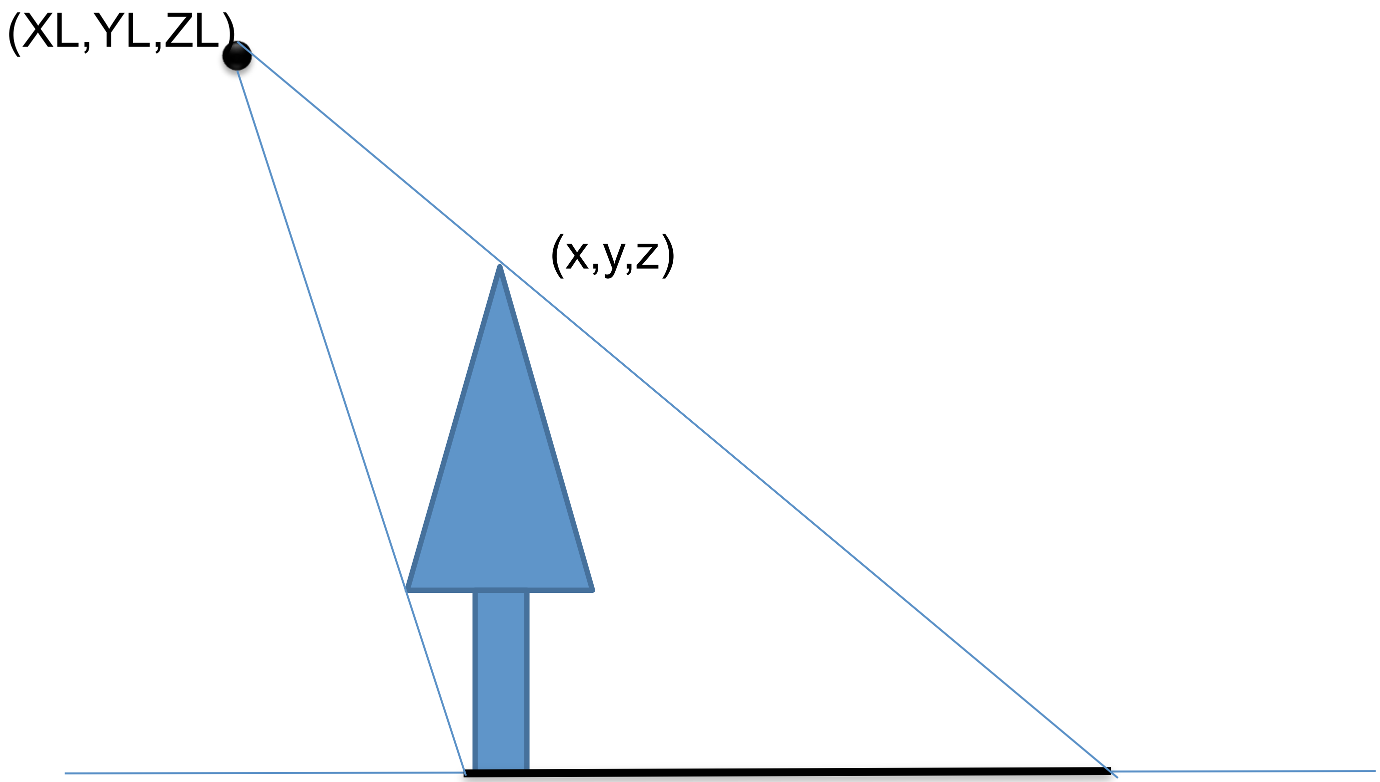
Implement simple physics

- Energy conservation

Multiple Cars

- Each has its own parameter t, assuming arc-length parameterization

Hack Shadow



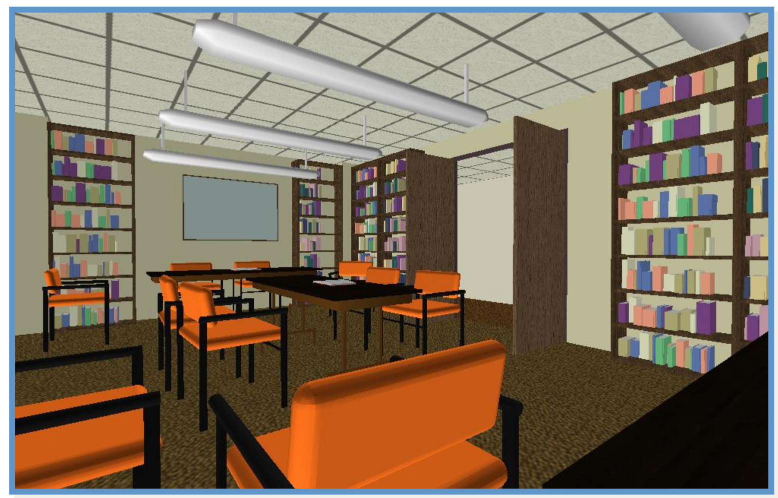
Train smoke?

- Balls moving upward and dissipating

Texture Mapping

- Important topic: nearly all objects textured
  - Wood grain, faces, bricks and so on
  - Adds visual detail to scenes

Polygonal model

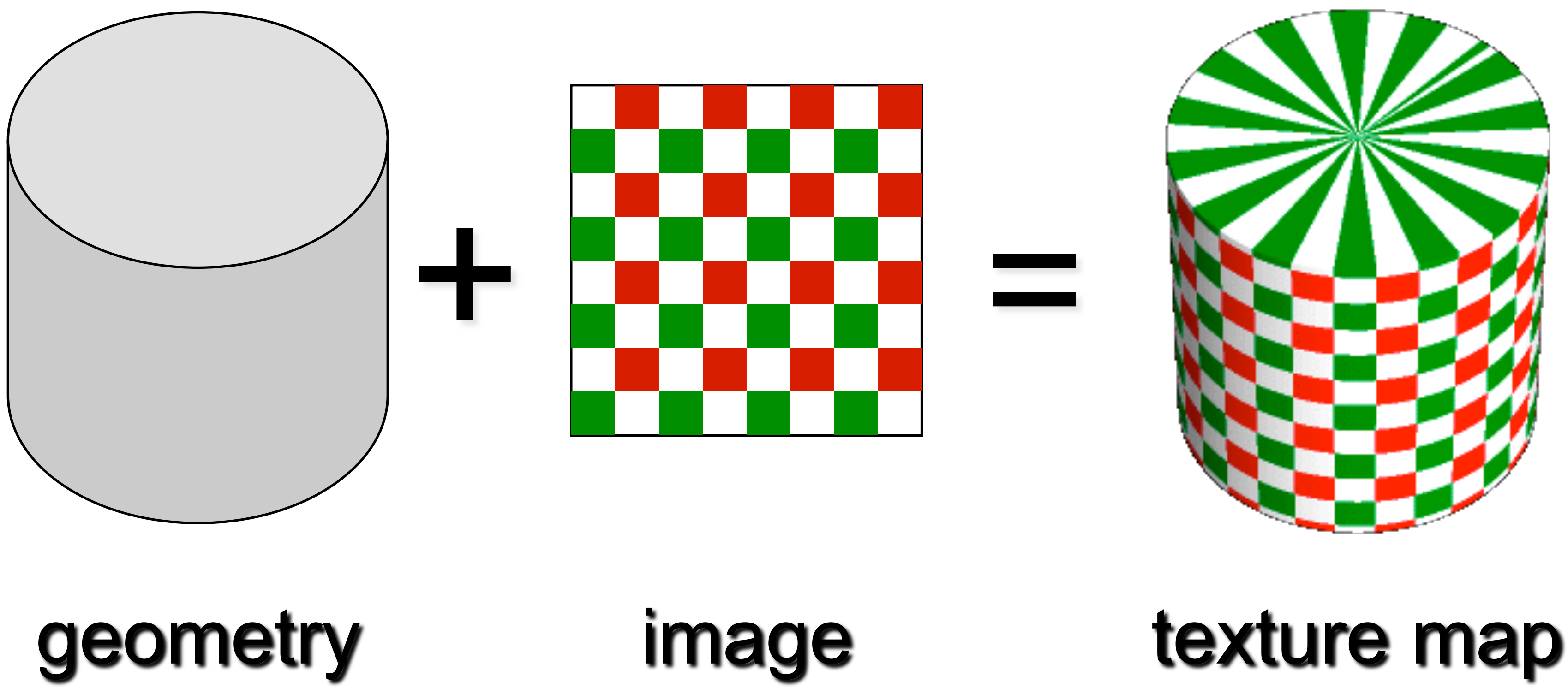


With surface texture

Adding Visual Detail

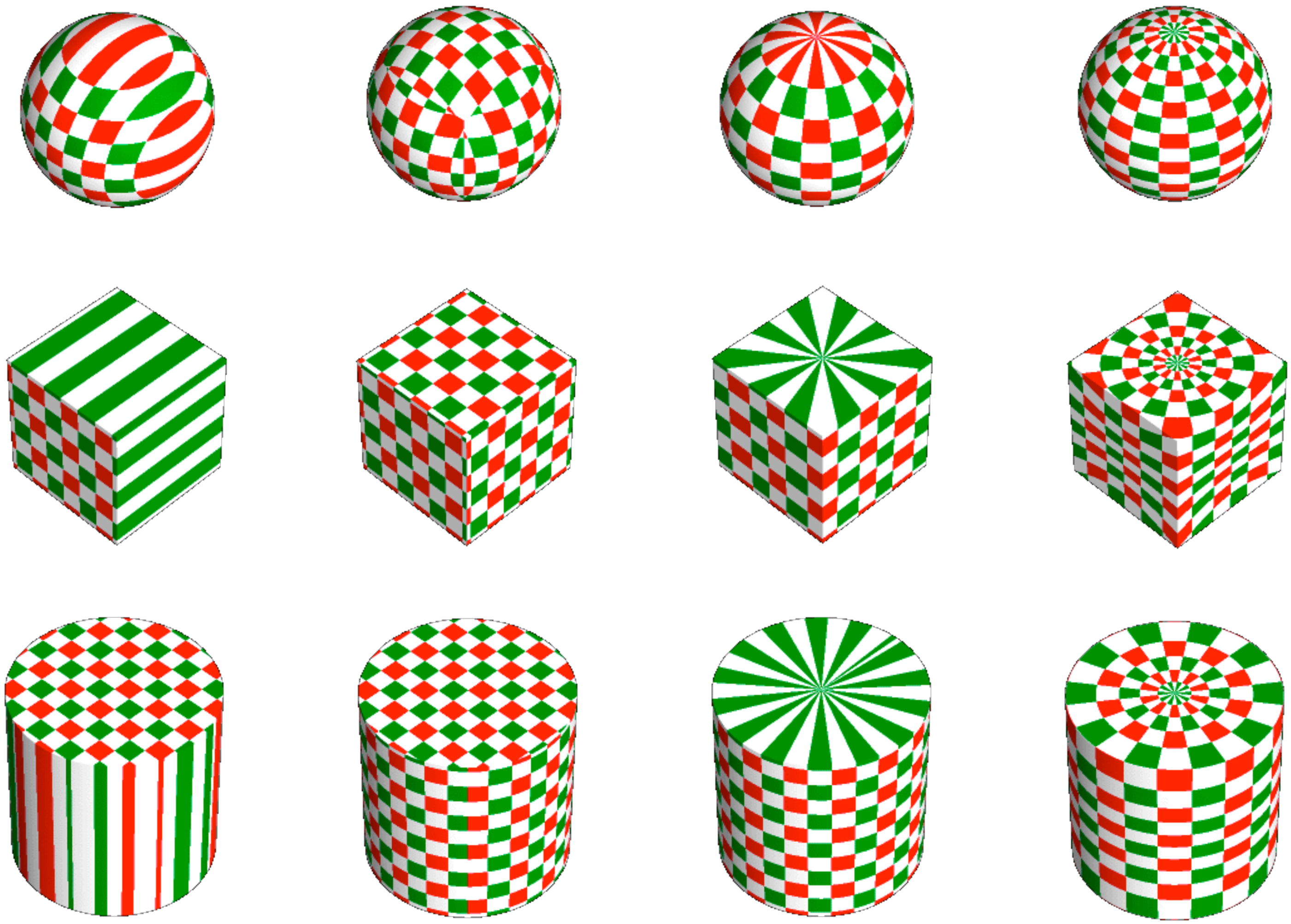
- Basic idea: use images instead of more polygons to represent fine scale color variation

Parameterization



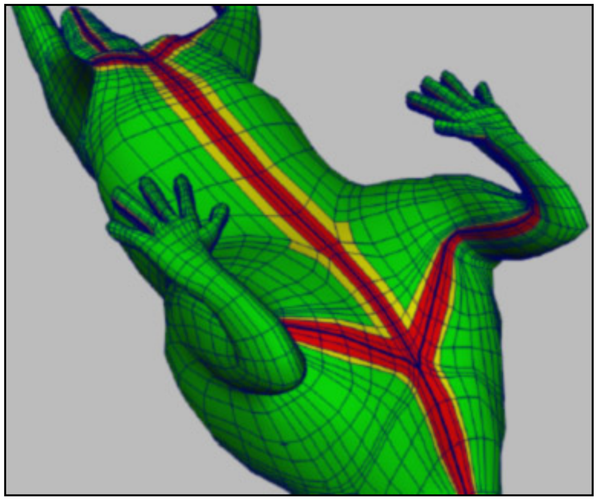
- Q: How do we decide *where* on the geometry each color from the image should go?

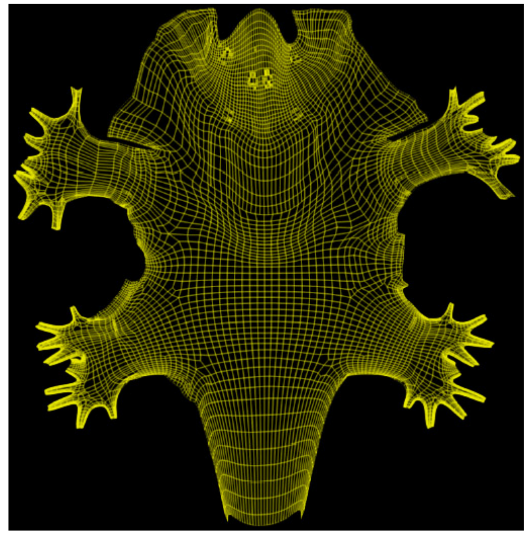
Option: Varieties of mappings



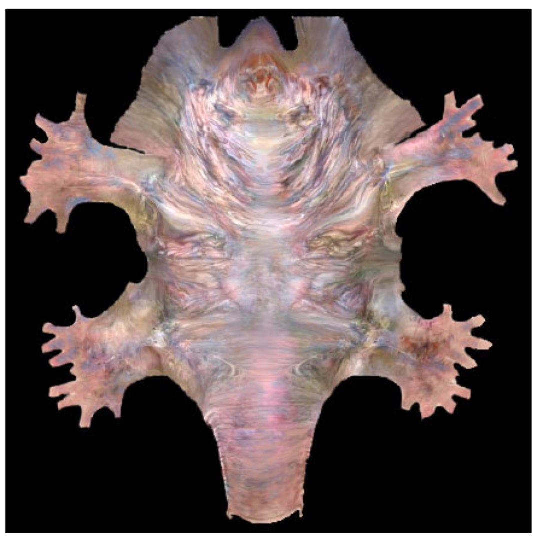
[Paul Bourke]

Option: unfold the surface



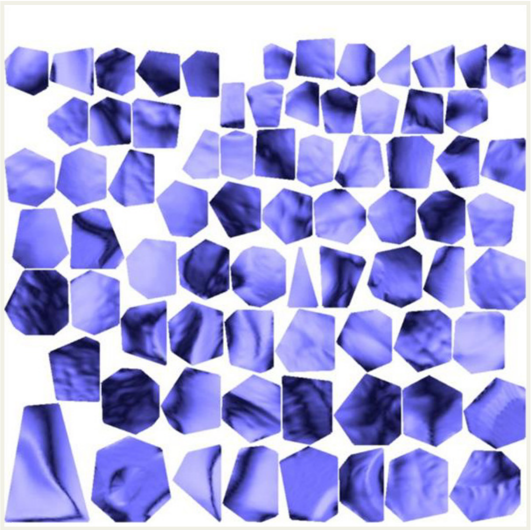


[Piponi2000]

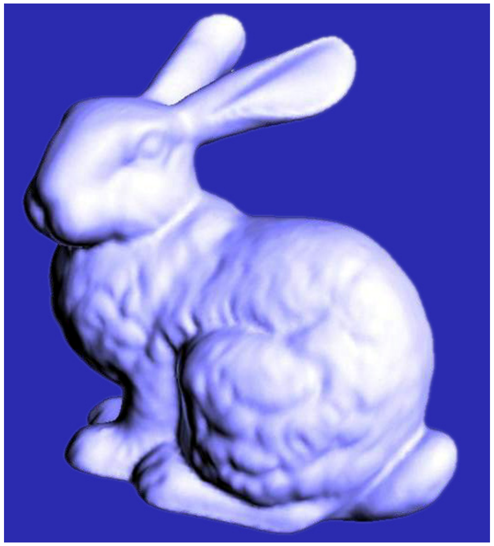


Option: make an atlas

charts



atlas



surface

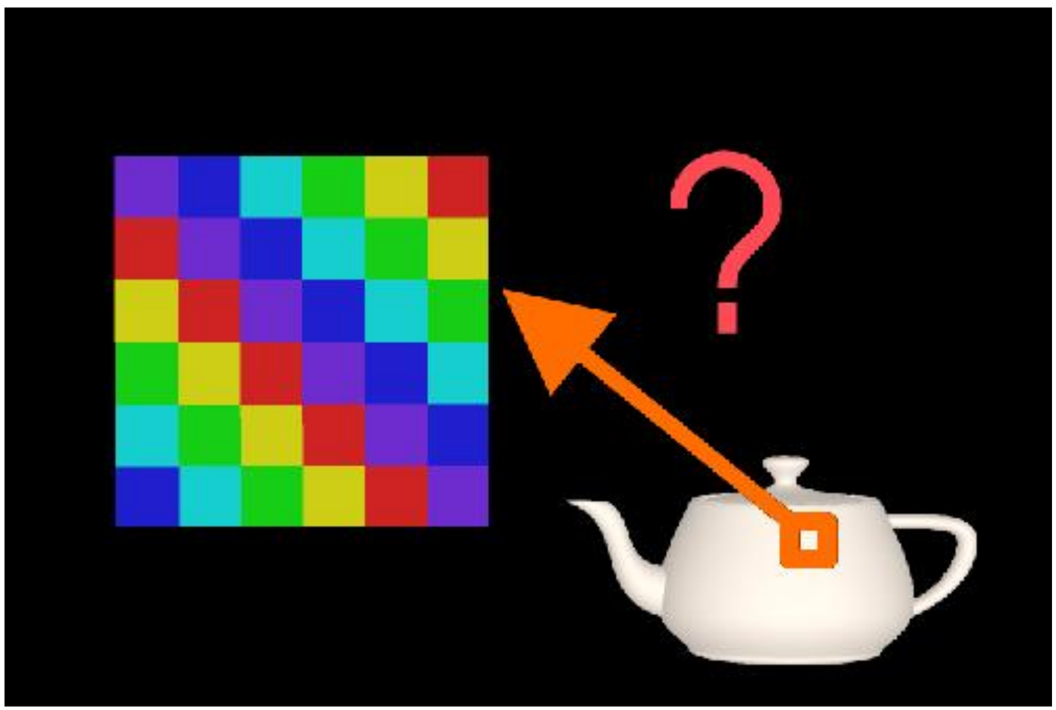
[Sander2001]

Outline

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

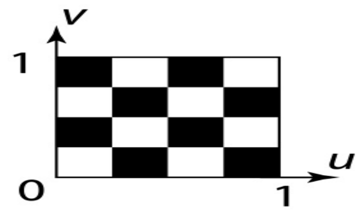
How to map object to texture?

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

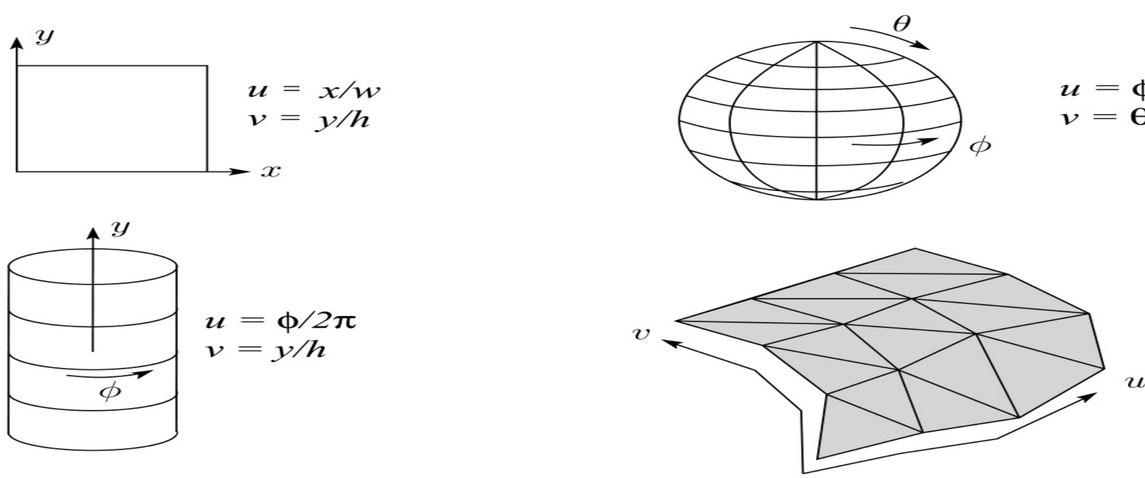


Implementing texture mapping

- A texture lives in it own abstract image coordinates paramaterized by (*u,v*) in the range ([0..1], [0..1]):



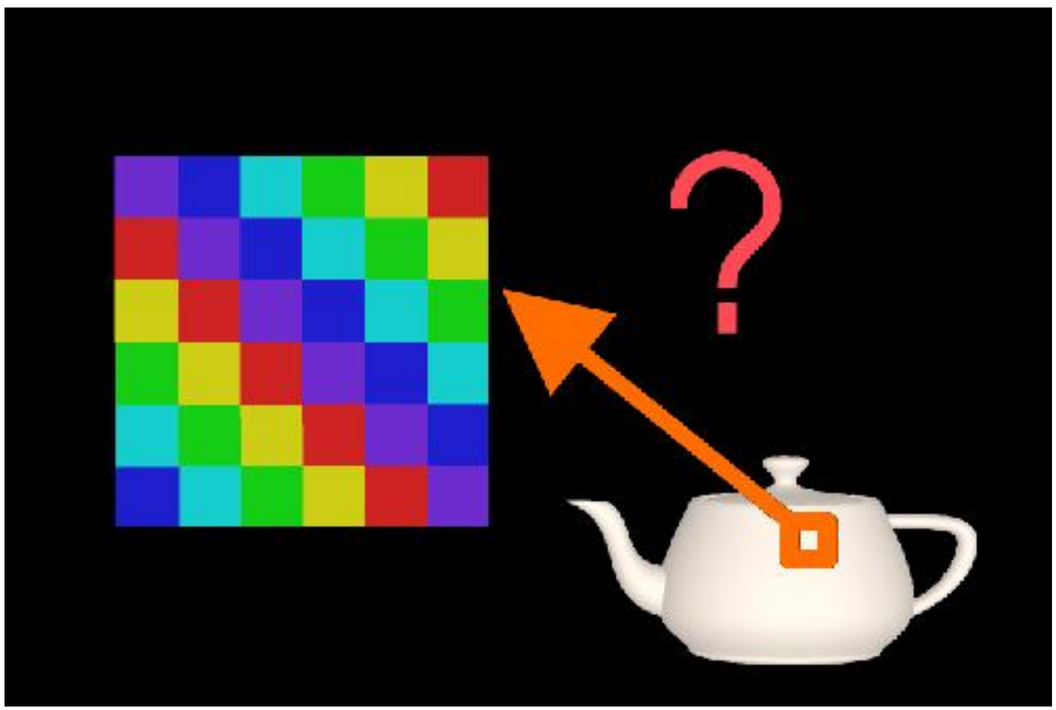
- It can be wrapped around many different surfaces:



- Note: if the surface moves/deforms, the texture goes with it.

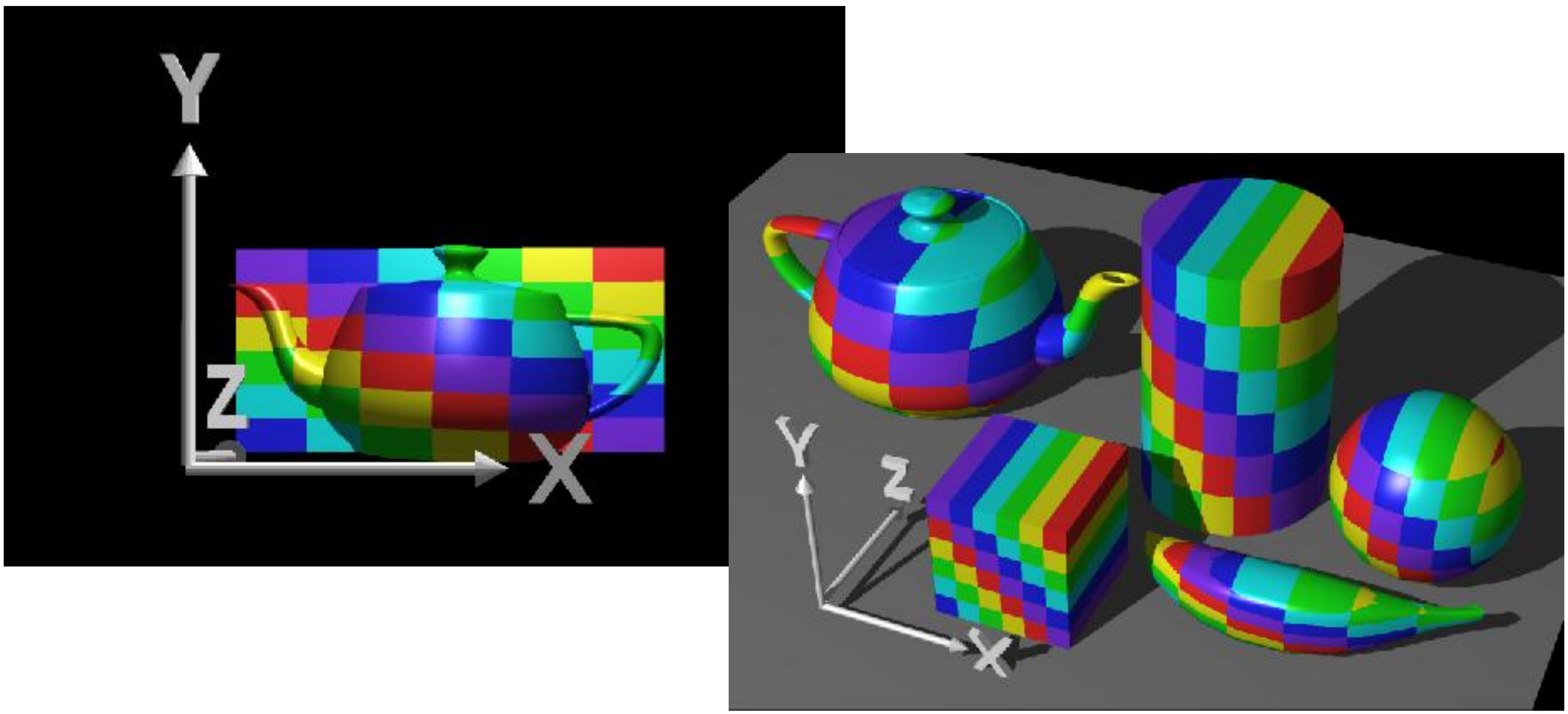
How to map object to texture?

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



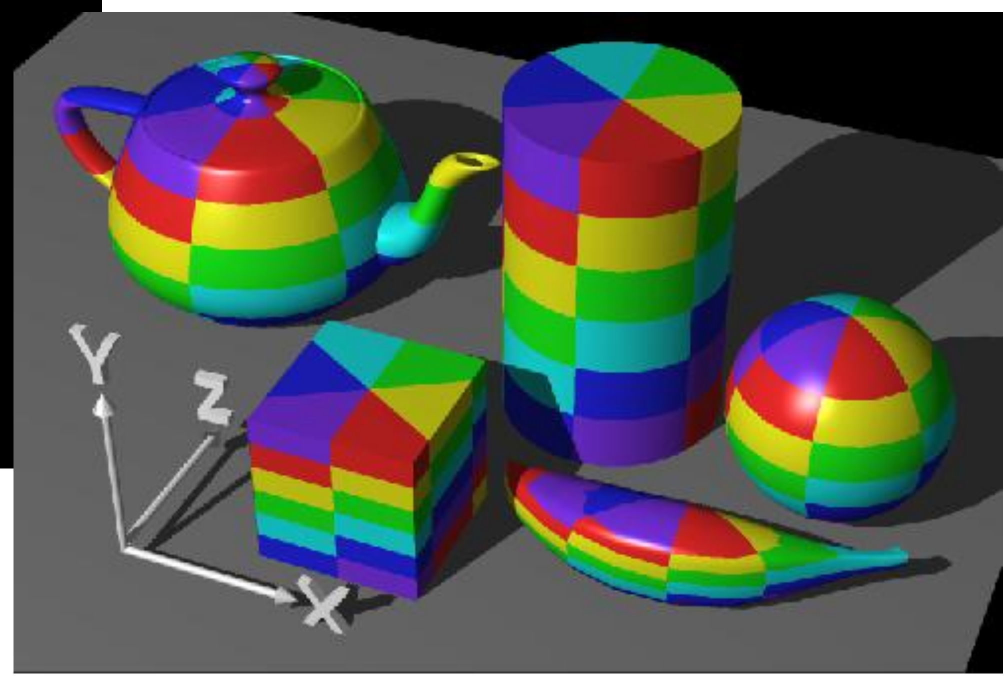
Planar mapping

- Like projections, drop z coord (u,v) = (x/W,y/H)
- Problems: what happens near silhouettes?



Cylindrical Mapping

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

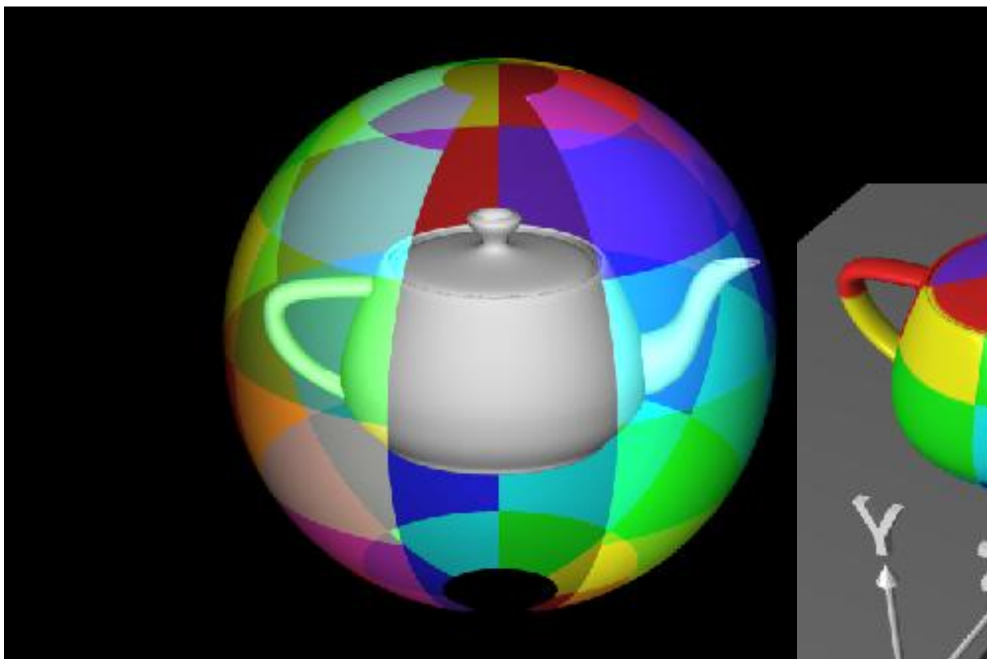


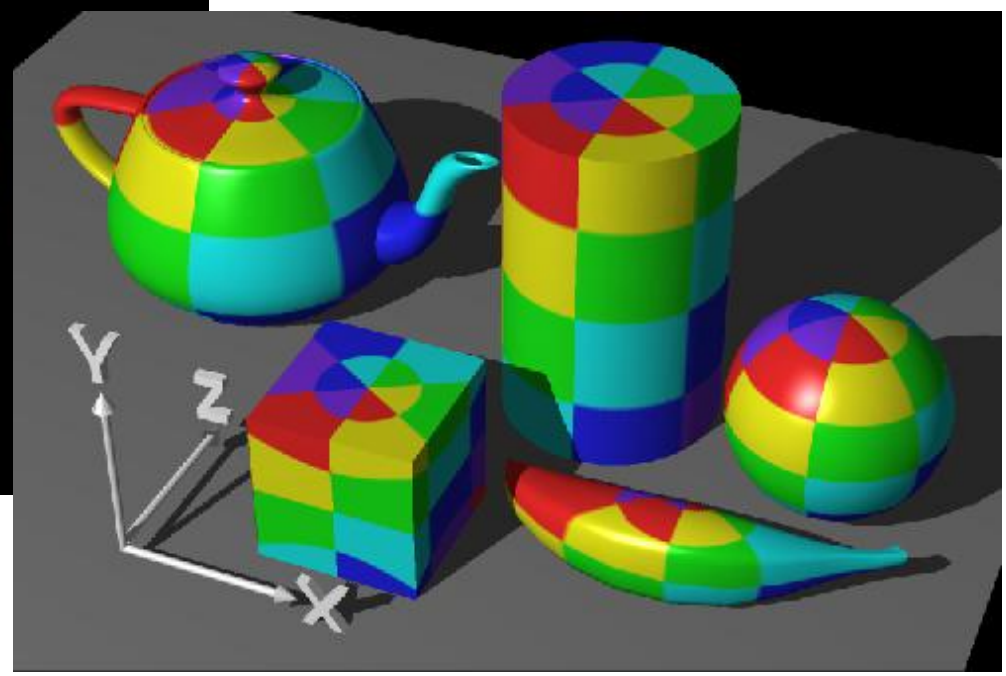
Basic procedure

- 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, …
  - Maps from object to these are simply coordinate transform

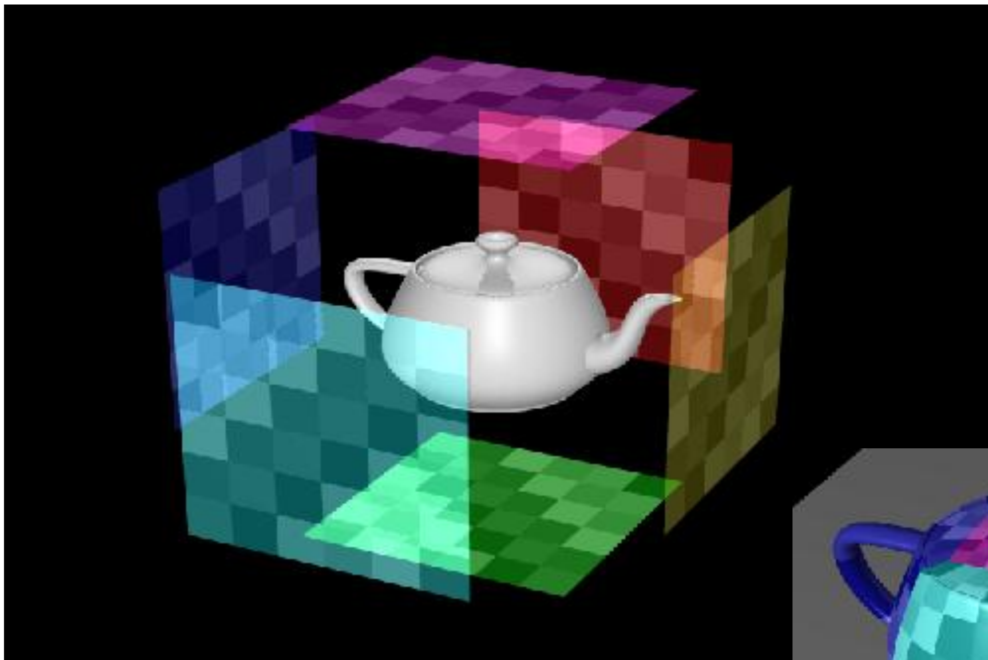
Spherical Mapping

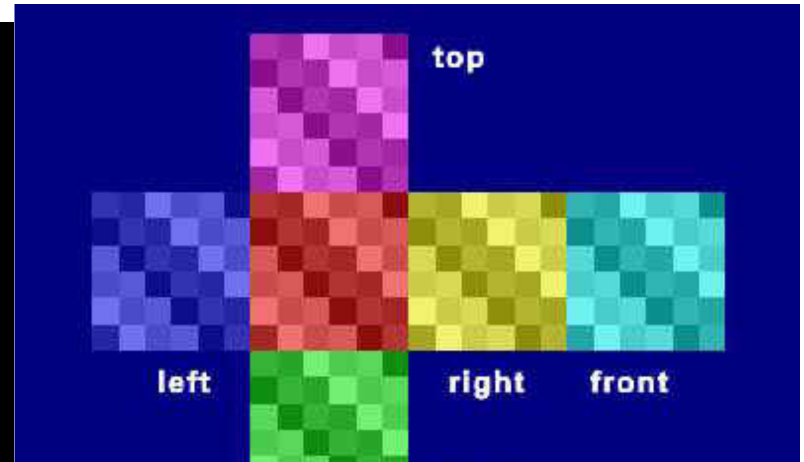
- Convert to spherical coordinates: use latitude/ long.
  - Singularities at north and south poles

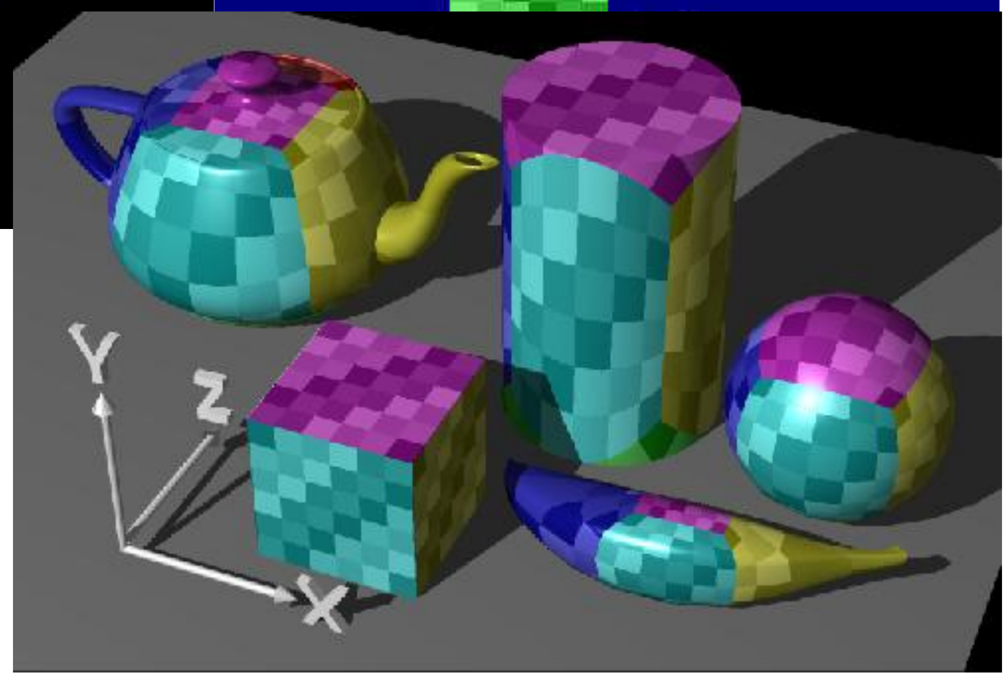




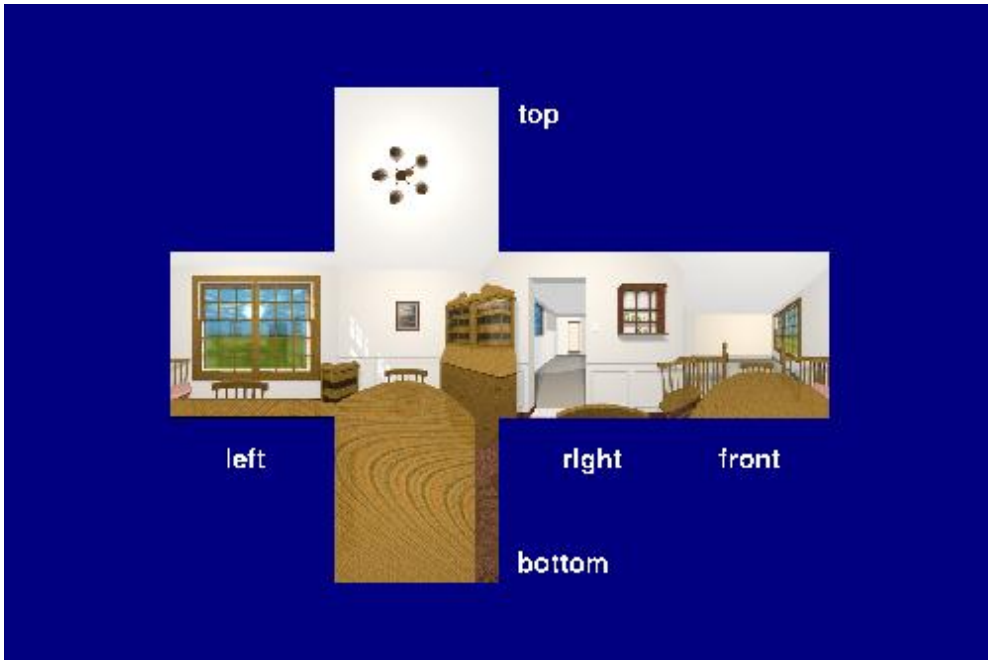
Cube Mapping



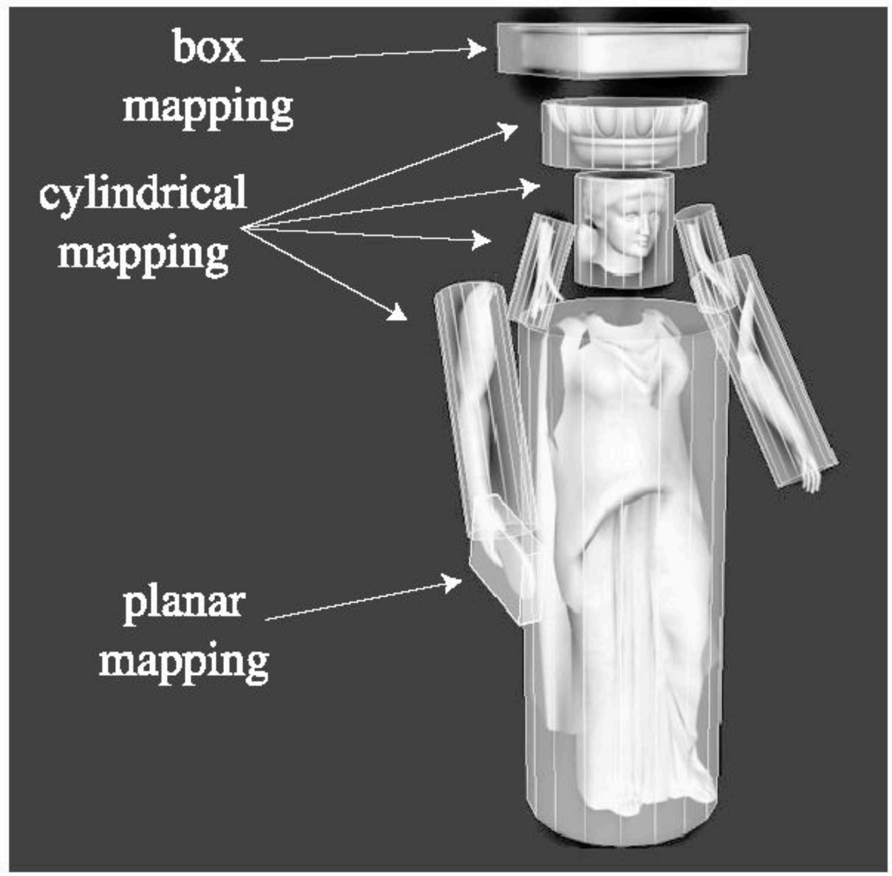




Cube Mapping



Piecewise Mapping

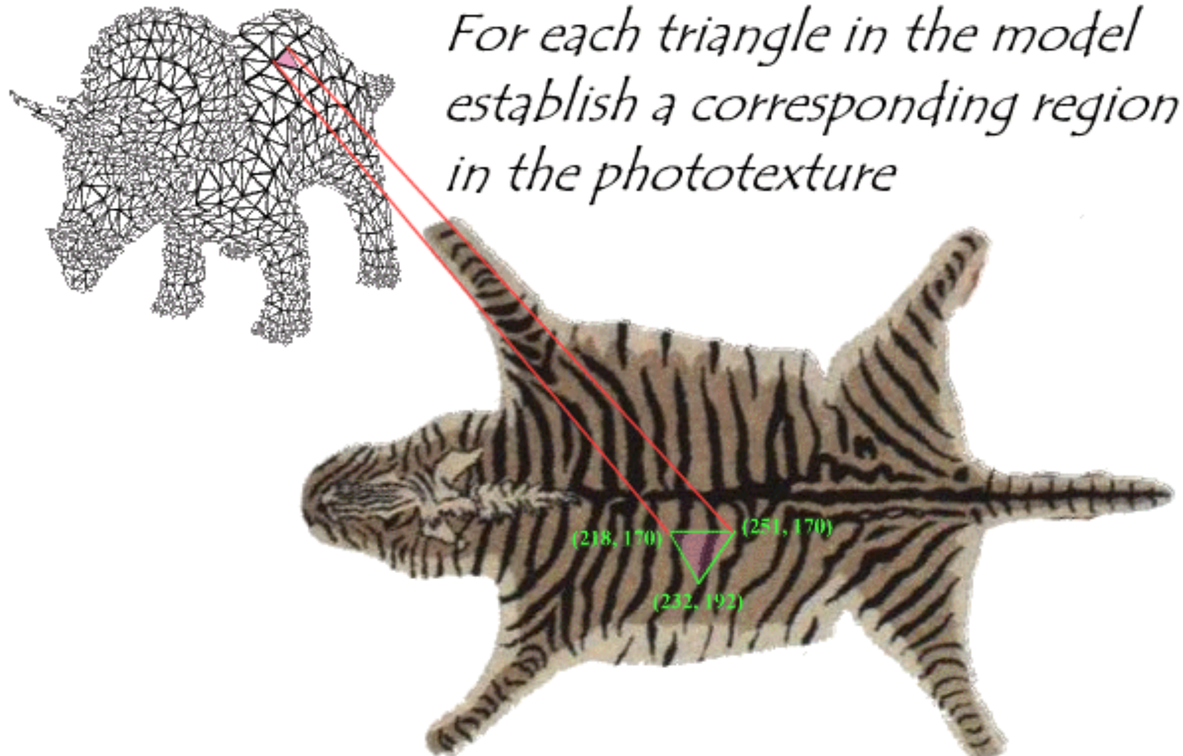


From Steve Marschner

Photo-textures

The concept is very simple!

For each triangle in the model establish a corresponding region in the phototexture



During rasterization interpolate the coordinate indices into the texture map

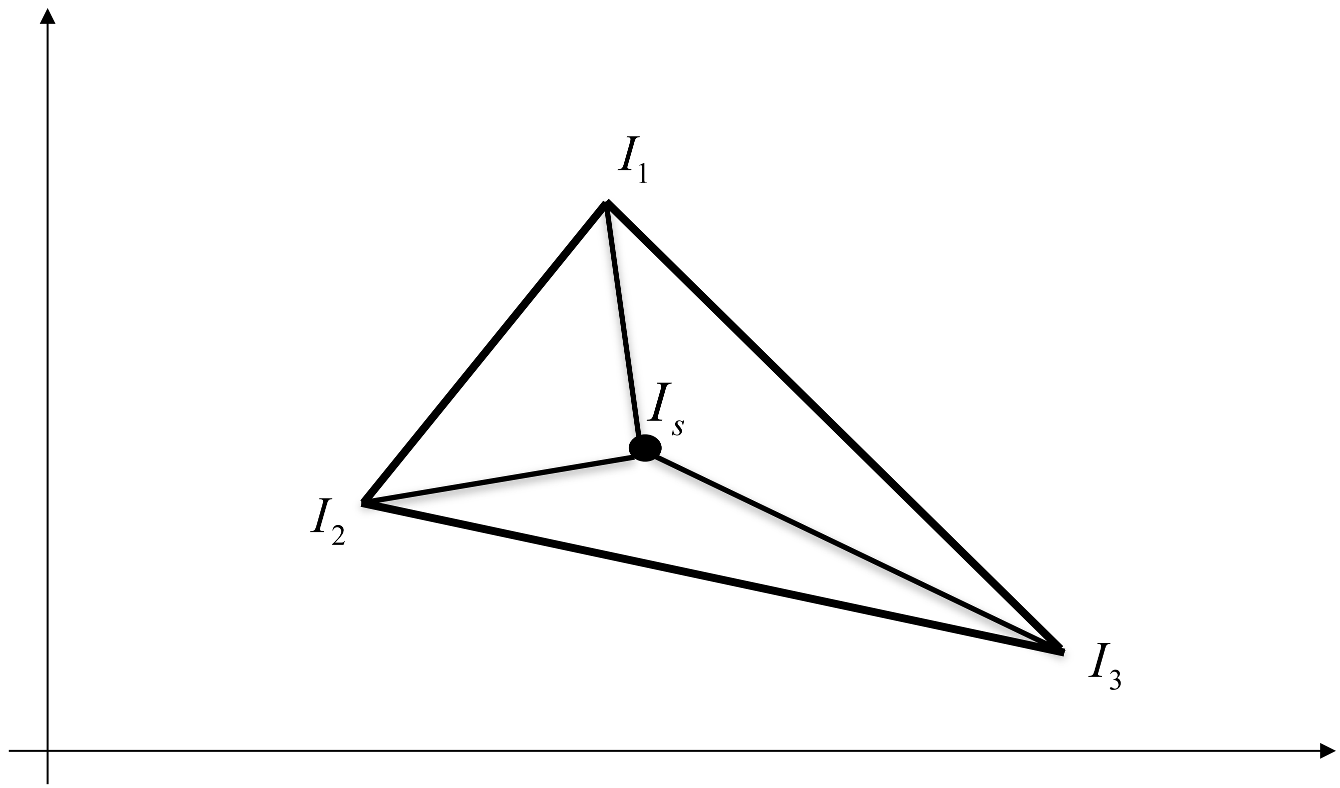
Slides from Leonard Mcmillan

Outline

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

1st idea: Gouraud interp. of texcoords

Using barycentric Coordinates

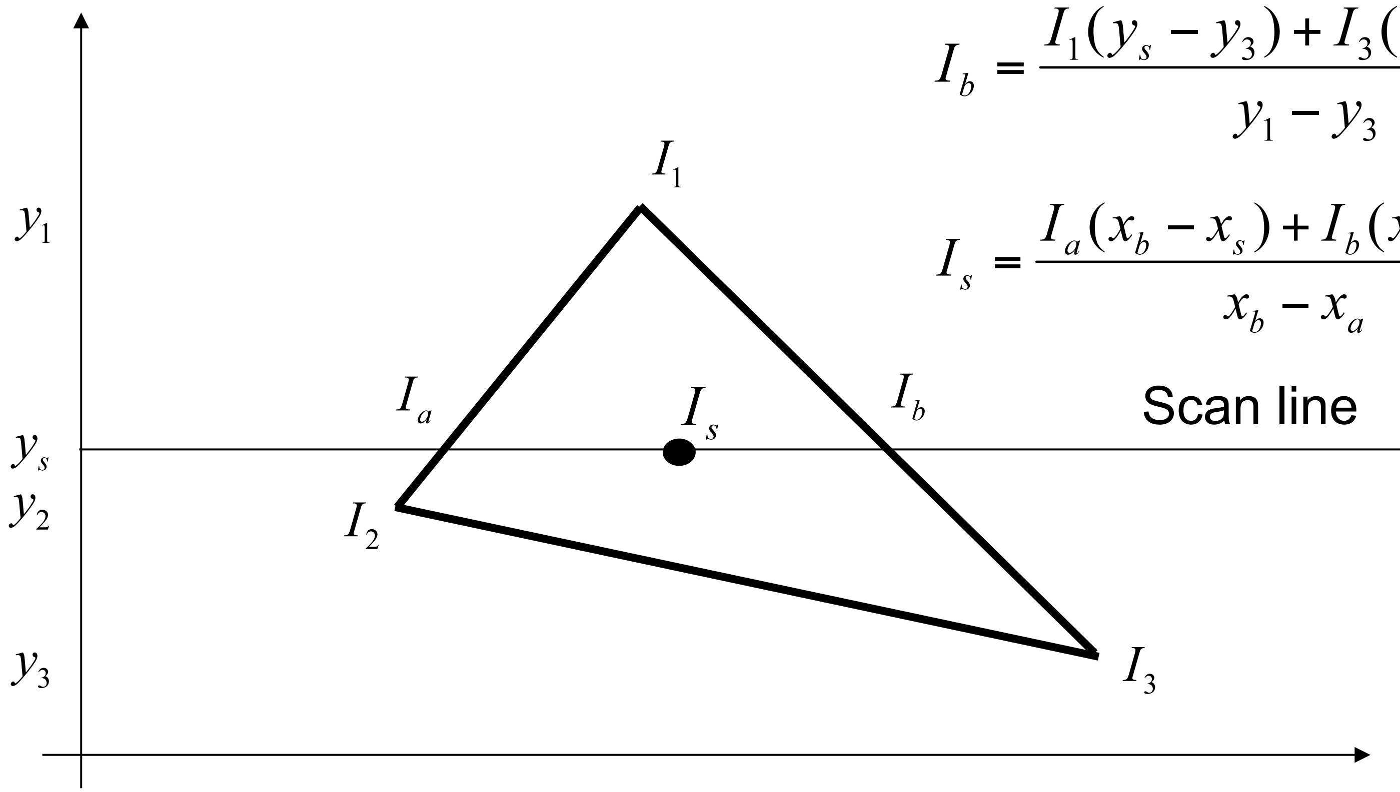


1^{st} idea: Gouraud interp. of texcoords

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

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

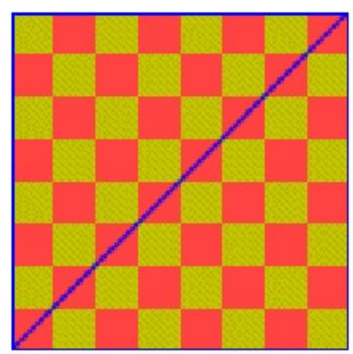
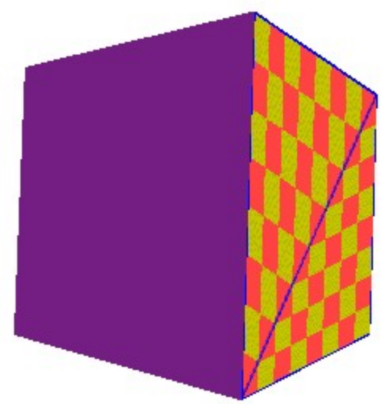
$$I_s = \frac{I_a(x_b - x_s) + I_b(x_s - x_a)}{x_b - x_a}$$



Artifacts

- McMillan's demo of this is at
  http://graphics.lcs.mit.edu/classes/6.837/F98/Lecture21/Slide05.html
- Another example
  http://graphics.lcs.mit.edu/classes/6.837/F98/Lecture21/Slide06.html
- What artifacts do you see?
- Why?
- 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*