CS559: Computer Graphics

Lecture 27: Texture Mapping

Li Zhang

Spring 2008

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

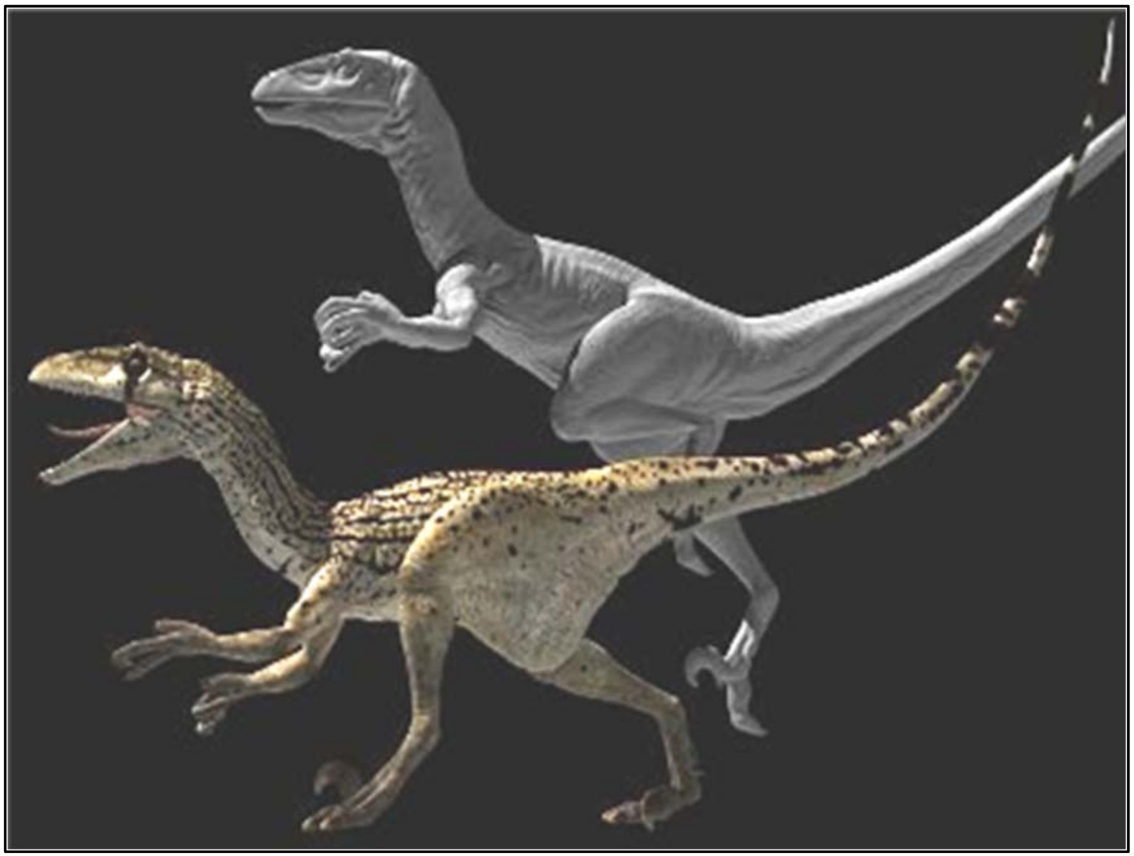
Today

- Continue on Texture mapping

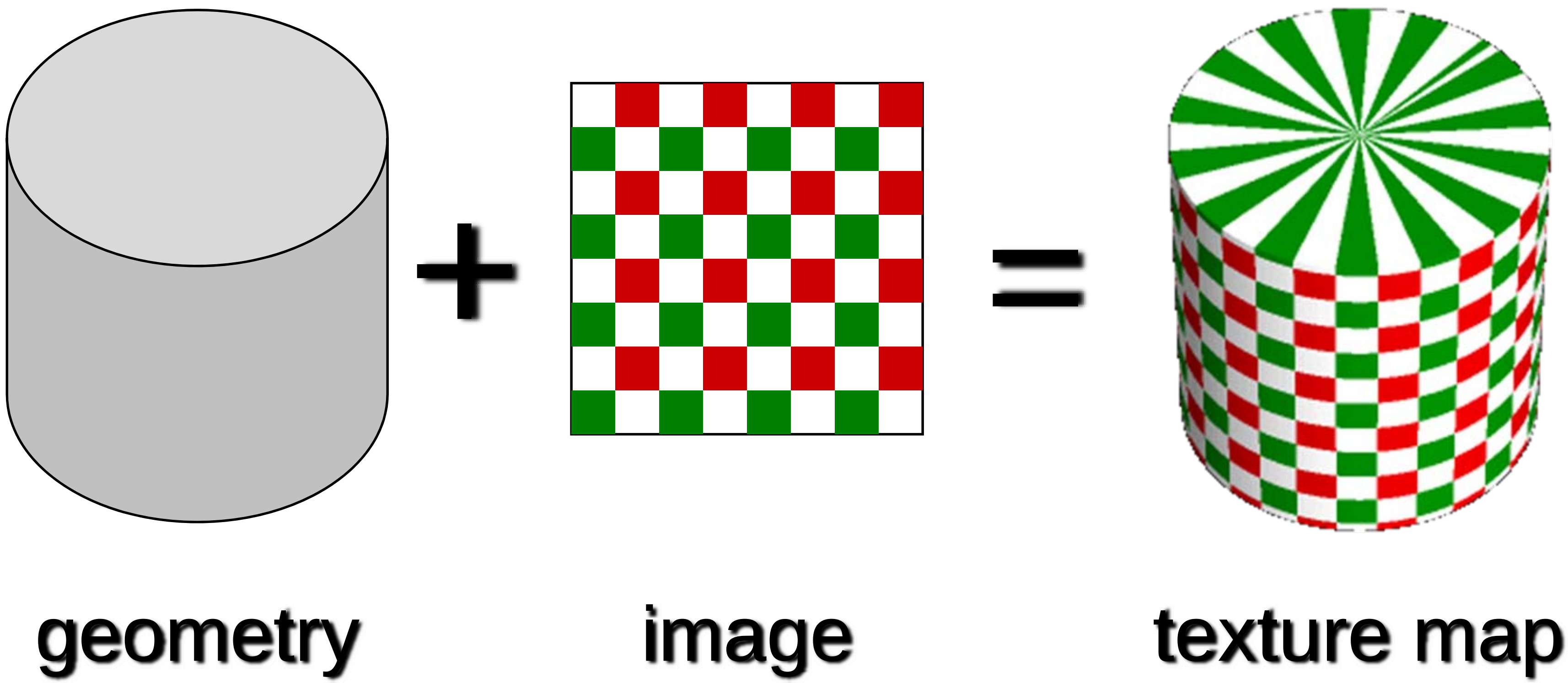
- Reading
  - Redbook: Ch 9
  - (highly recommended) Moller and Haines: *Real-Time Rendering, 3e*, Ch 6
    - Linux: /p/course/cs559-lizhang/public/readings/6_texture.pdf
    - Windows: P:\course\cs559-lizhang\public\readings\6_texture.pdf
  - (optional) Shirley: Ch 11.4 – 11.8

Adding Visual Detail

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

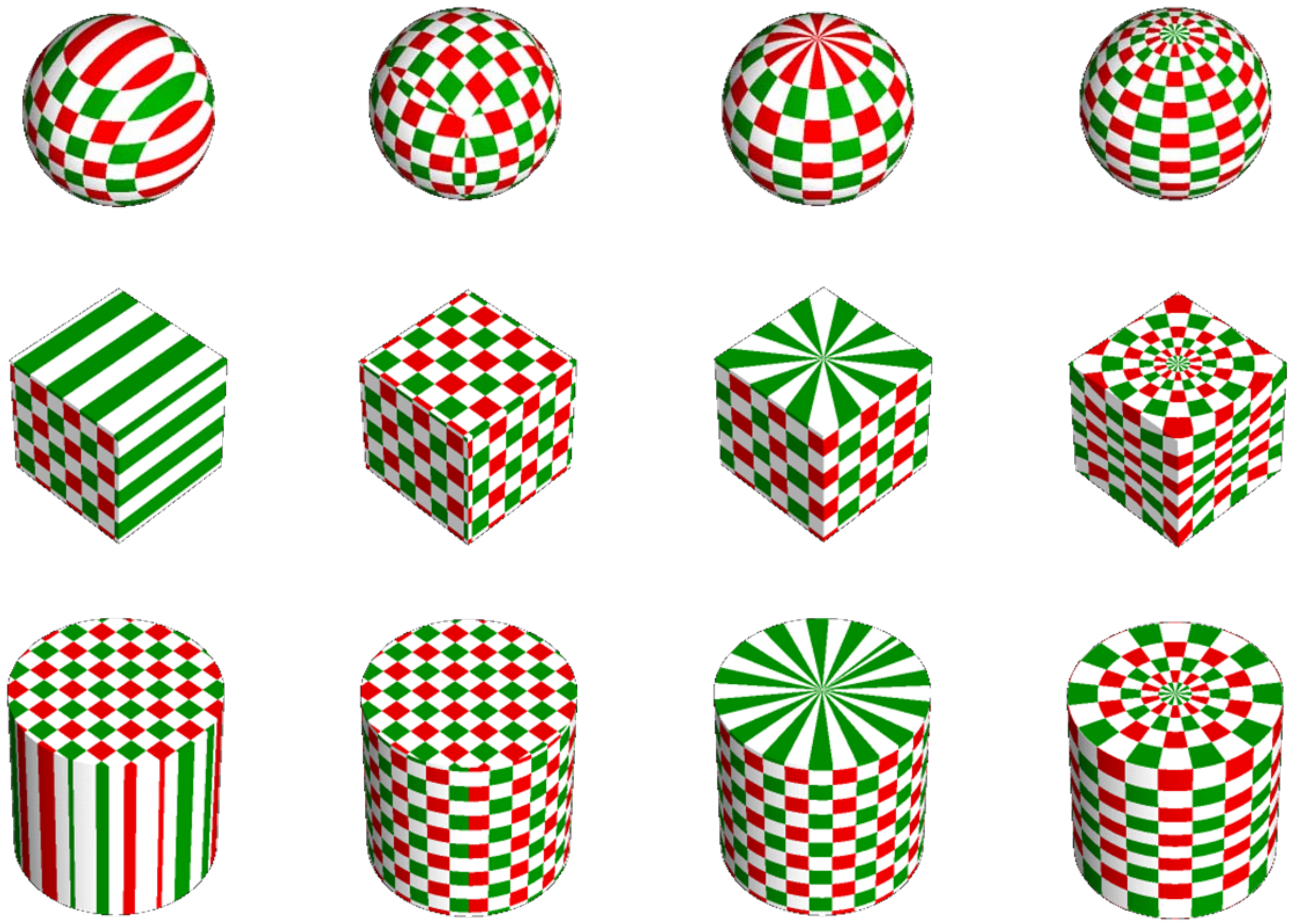


Parameterization



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

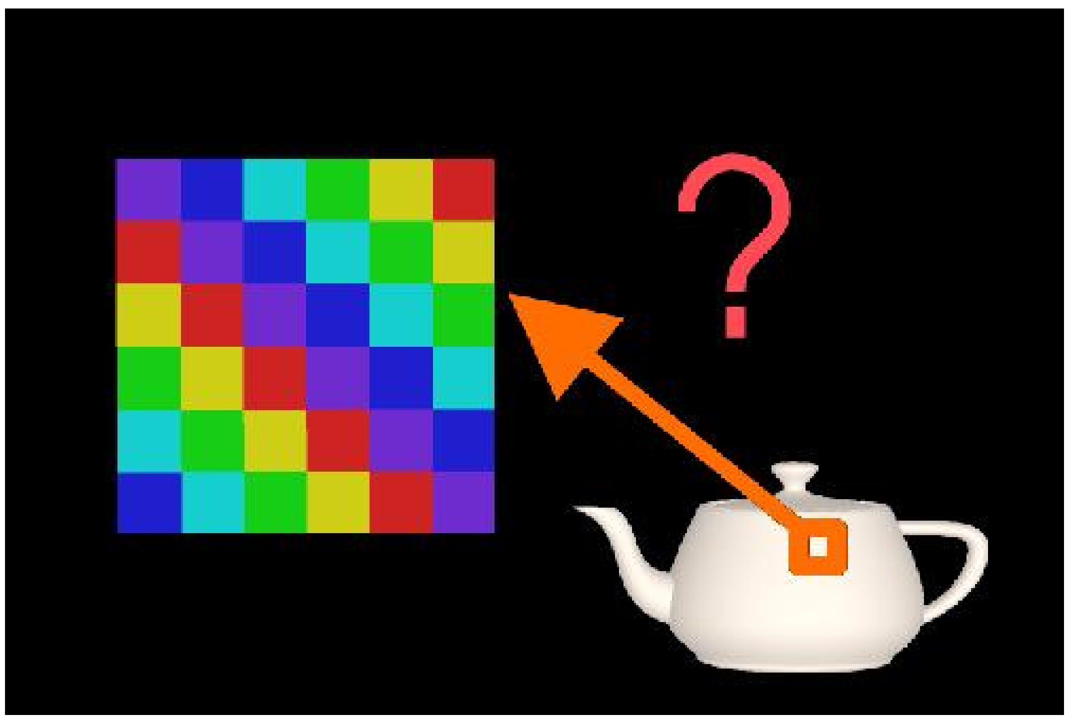
Option: Varieties of mappings



[Paul Bourke]

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

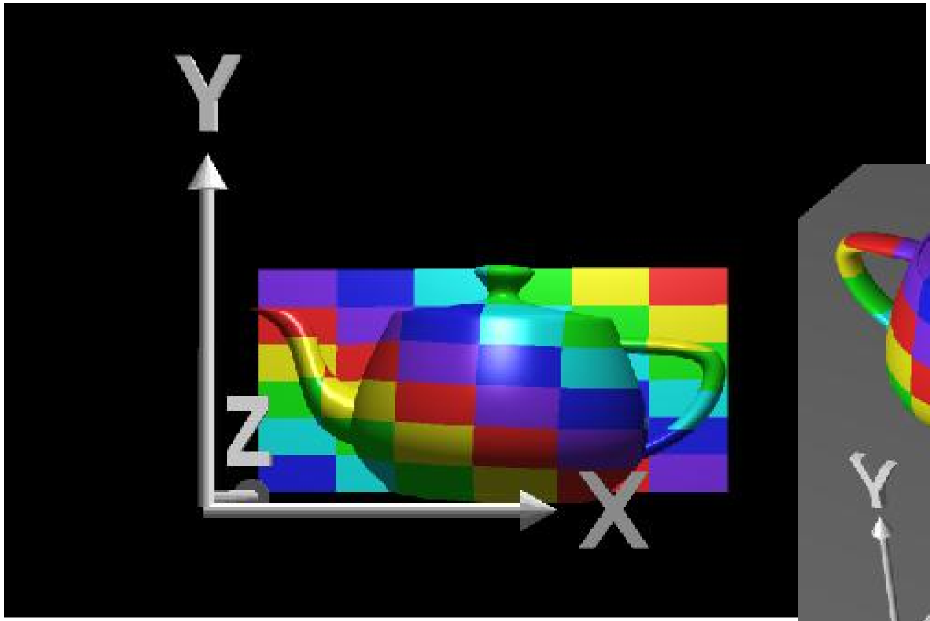


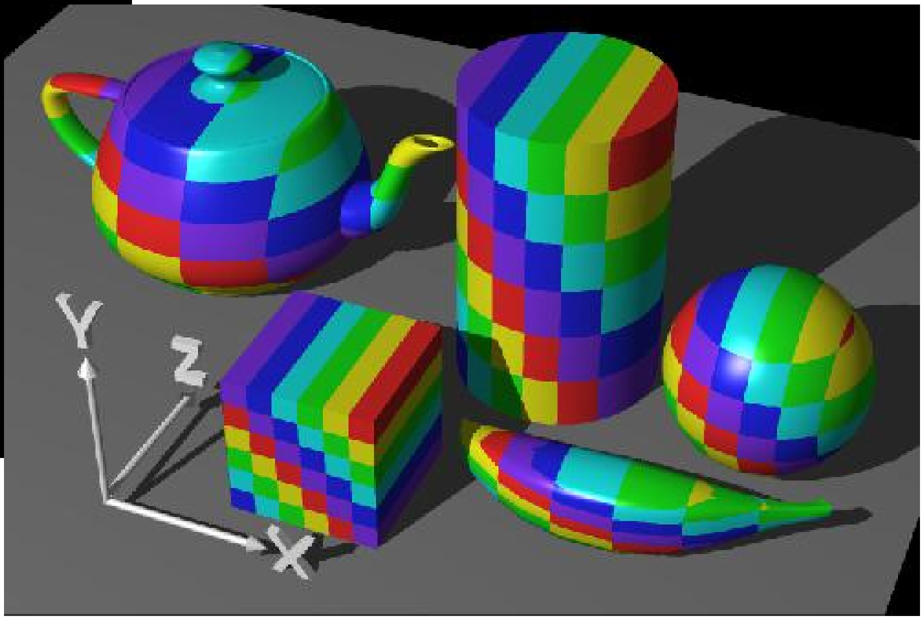
Outline

- *Types of mappings*
- Interpolating texture coordinates
- Texture Resampling
- Texture maping OpenGL
- Broader use of textures

Planar mapping

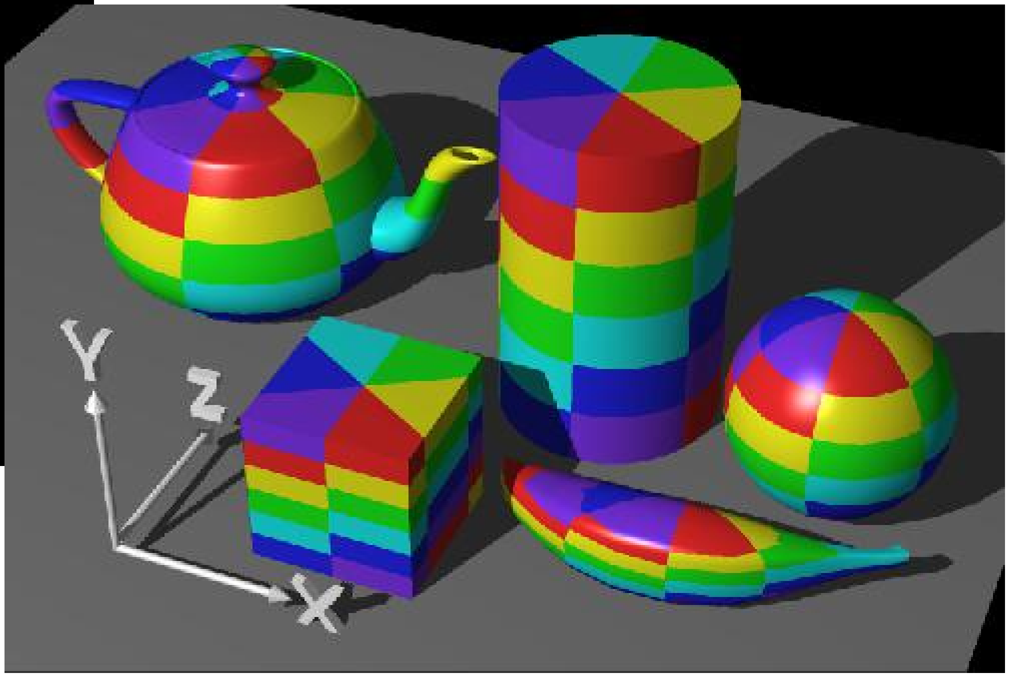
- Like projections, drop z coord (u,v) = (x/W,y/H)
- Problems: what happens near silhouettes?





Cylindrical Mapping

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

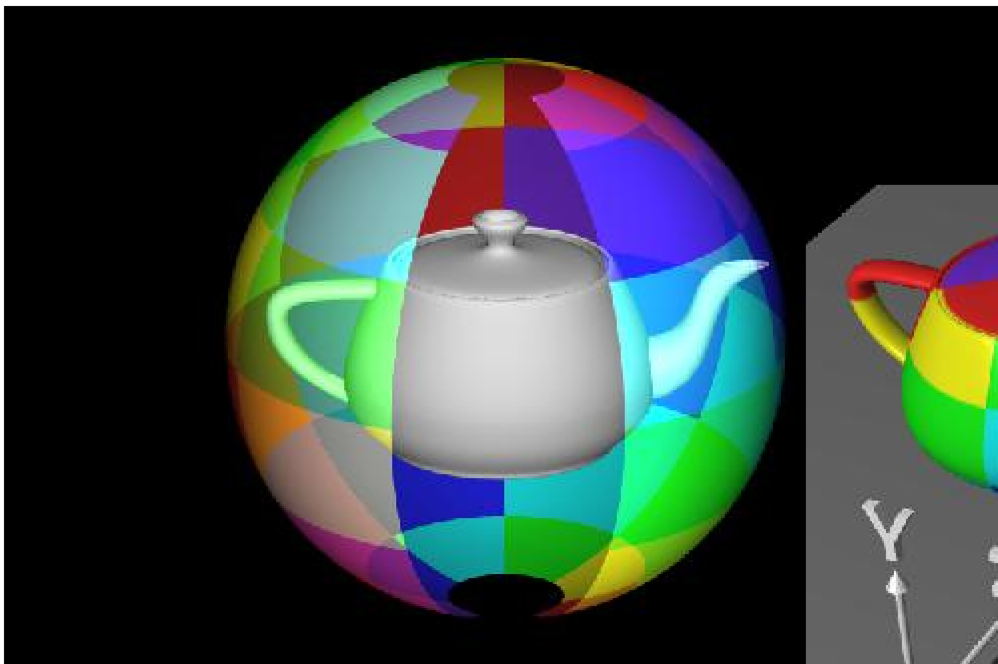


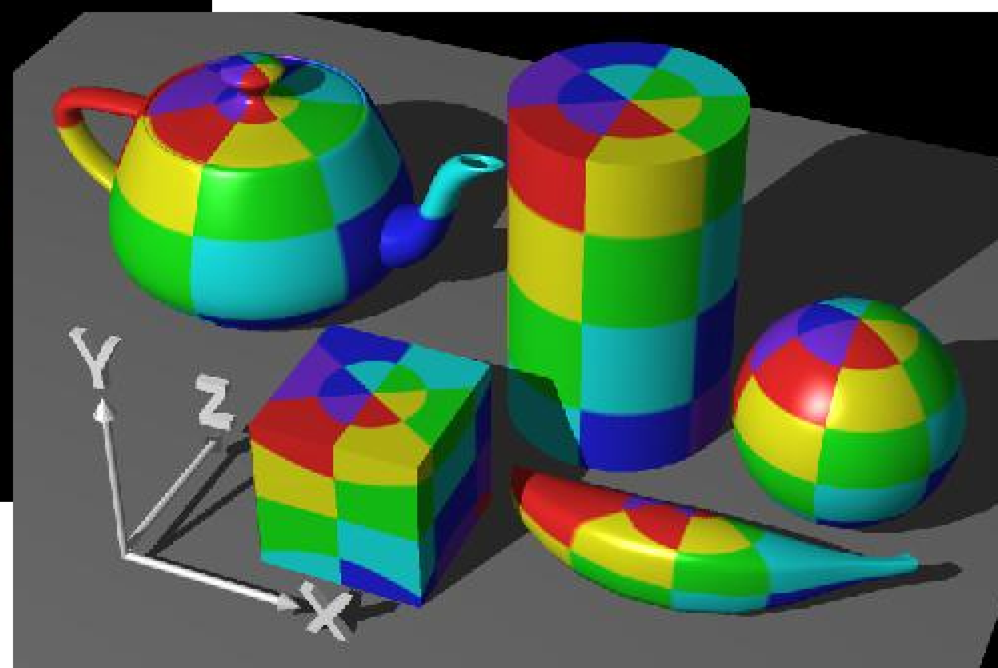
Basic procedure for simple mapping

- 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

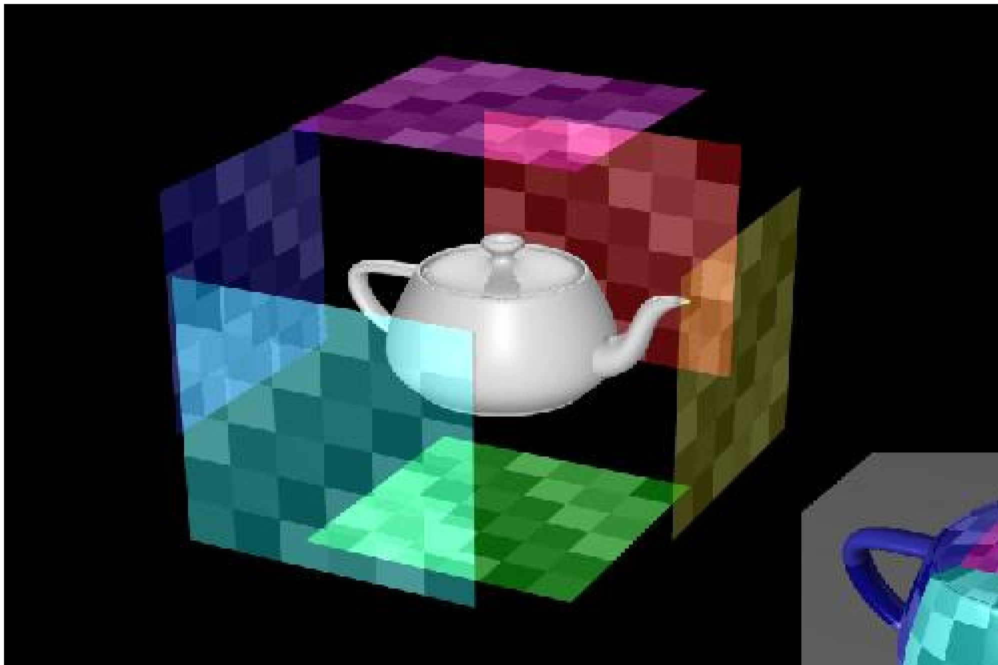
Spherical Mapping

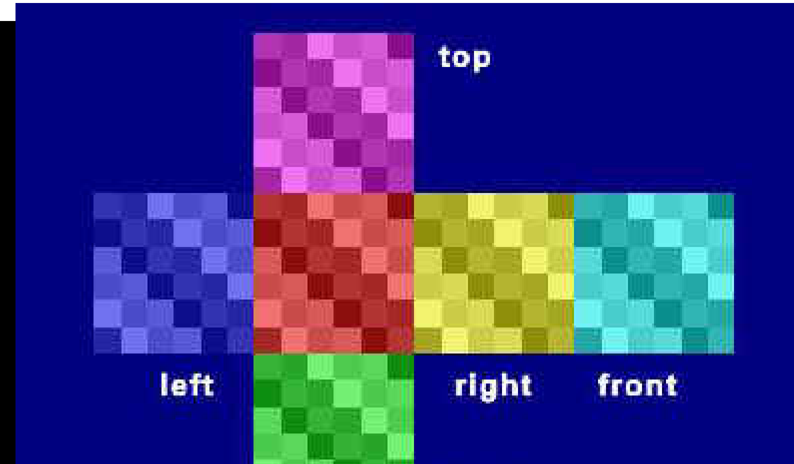
- Convert to spherical coordinates: use latitude/long.
  - Singularities at north and south poles

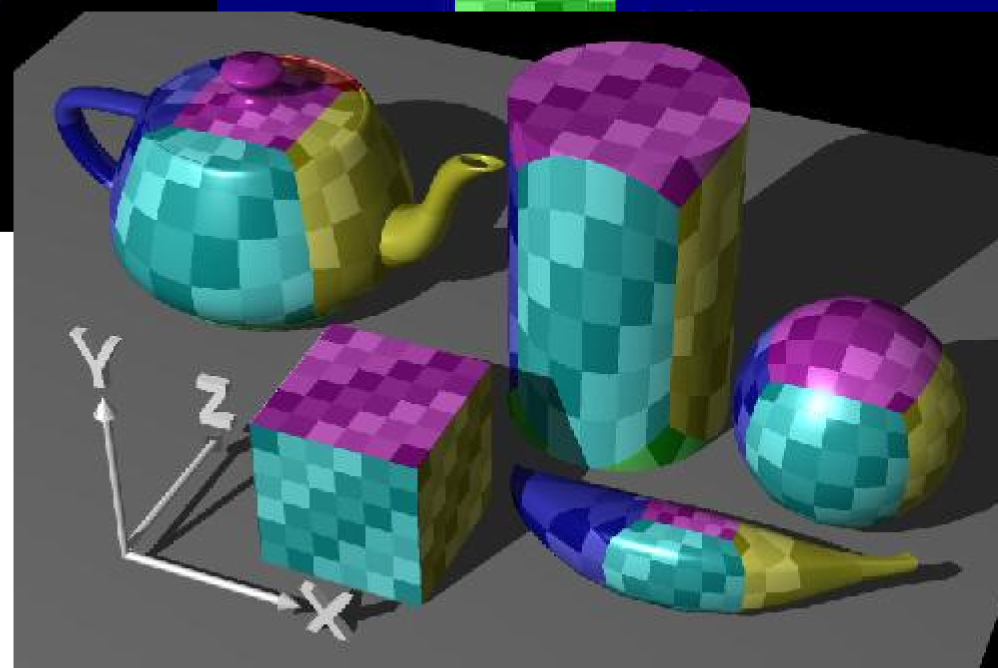




Cube Mapping







Cube Mapping

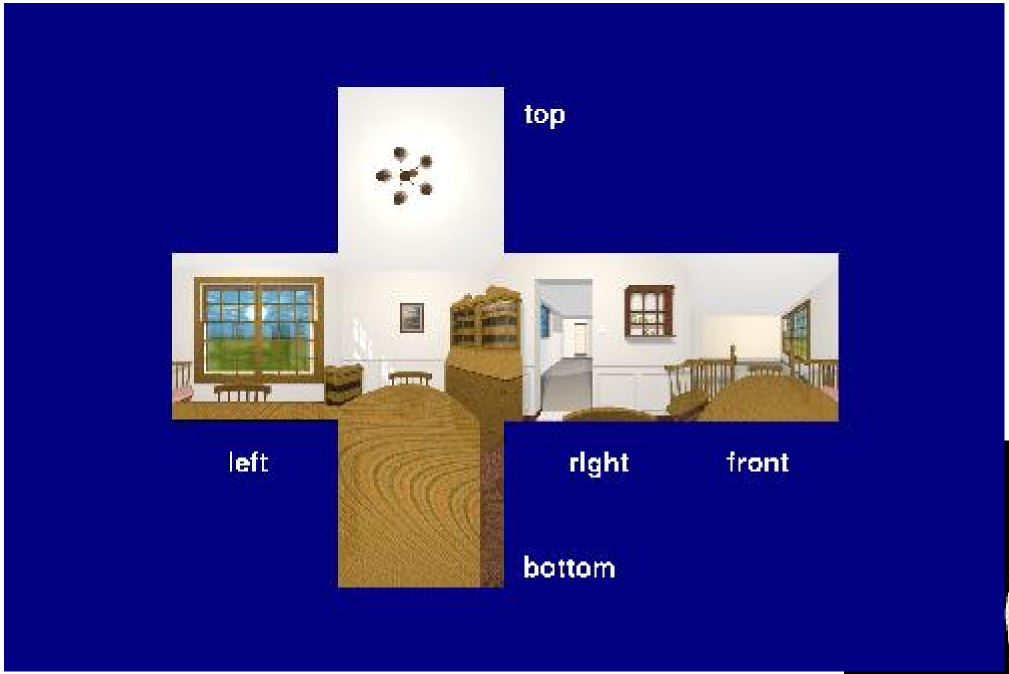
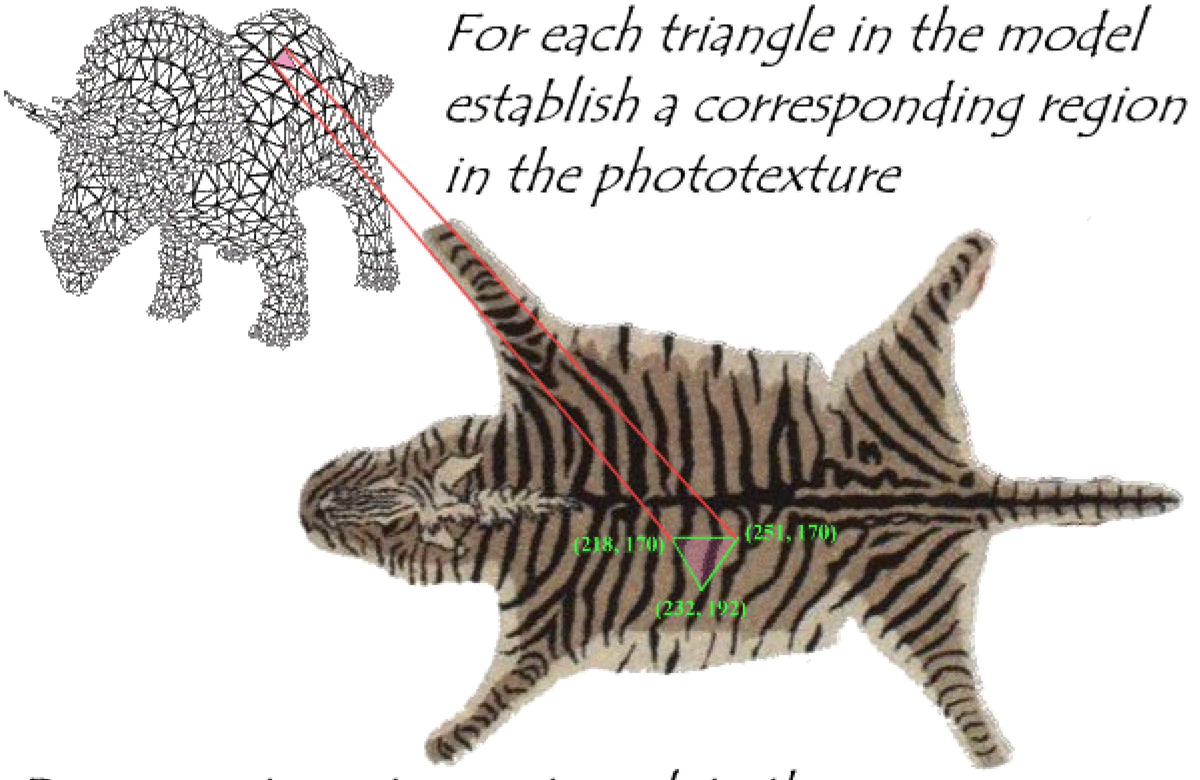


Photo-textures

The concept is very simple!



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

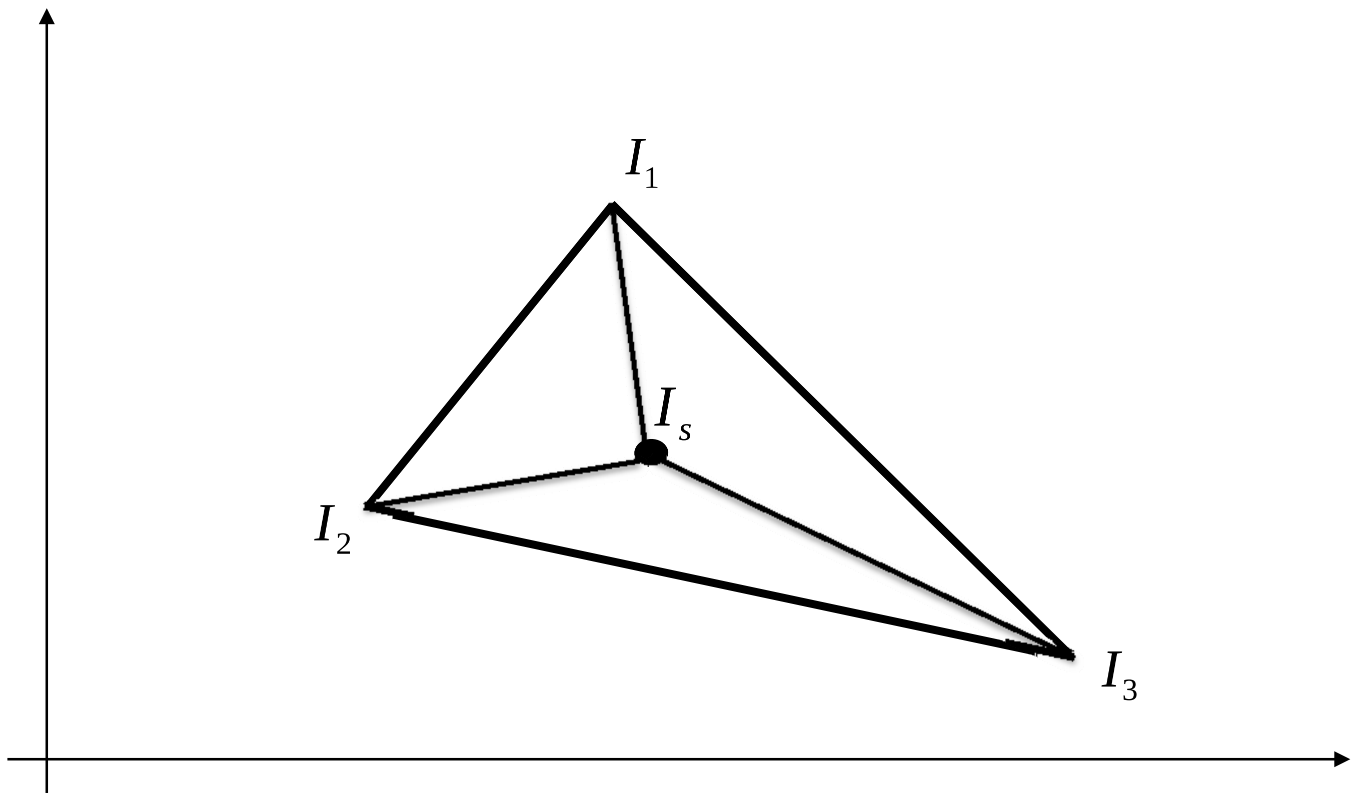
During rasterization interpolate the coordinate indices into the texture map

Outline

- Types of projections
- *Interpolating texture coordinates*
- Texture Resampling
- Texture mapping in OpenGL
- Broader use of textures

1st idea: Gouraud interp. of texcoords

Using barycentric Coordinates

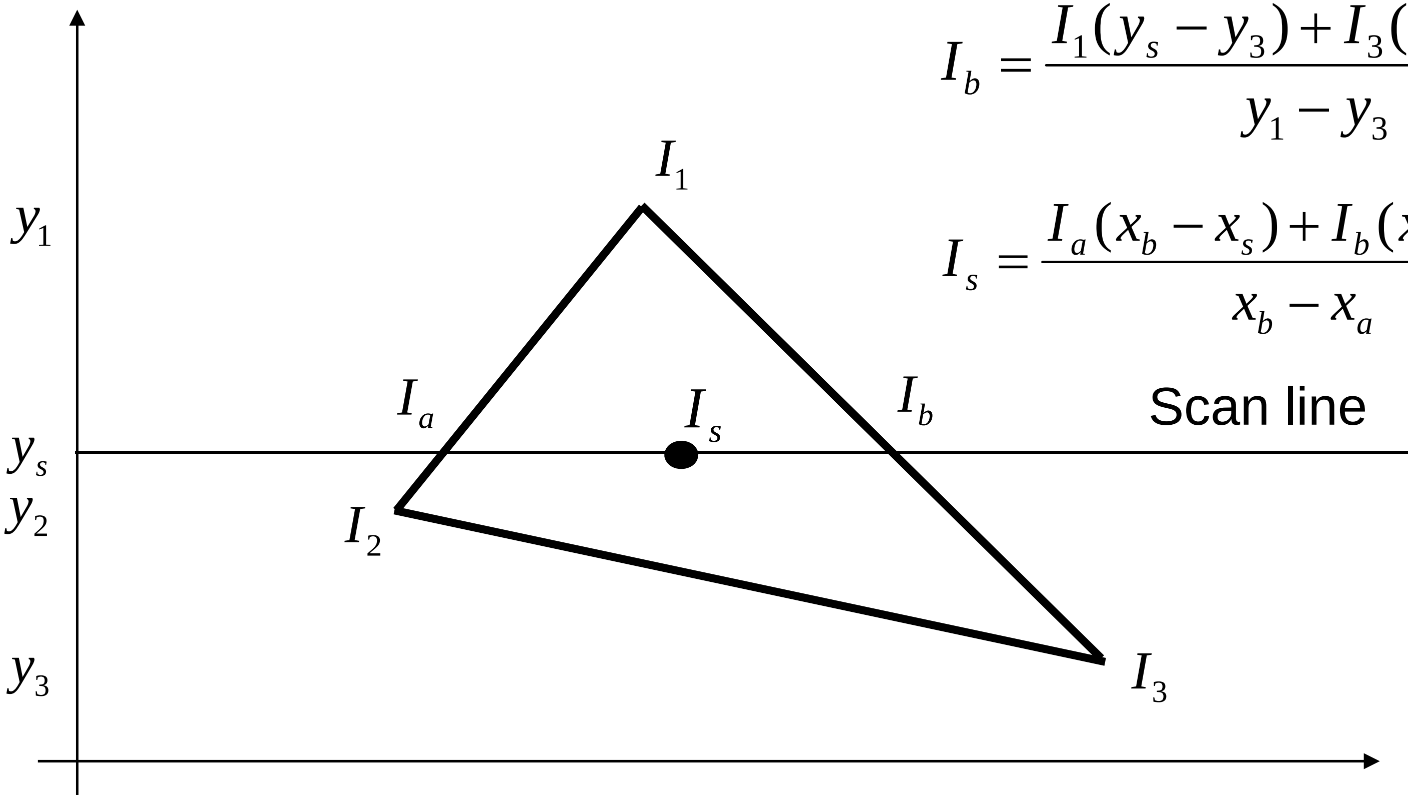


1st idea: Gouraud interp. of texcoords

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

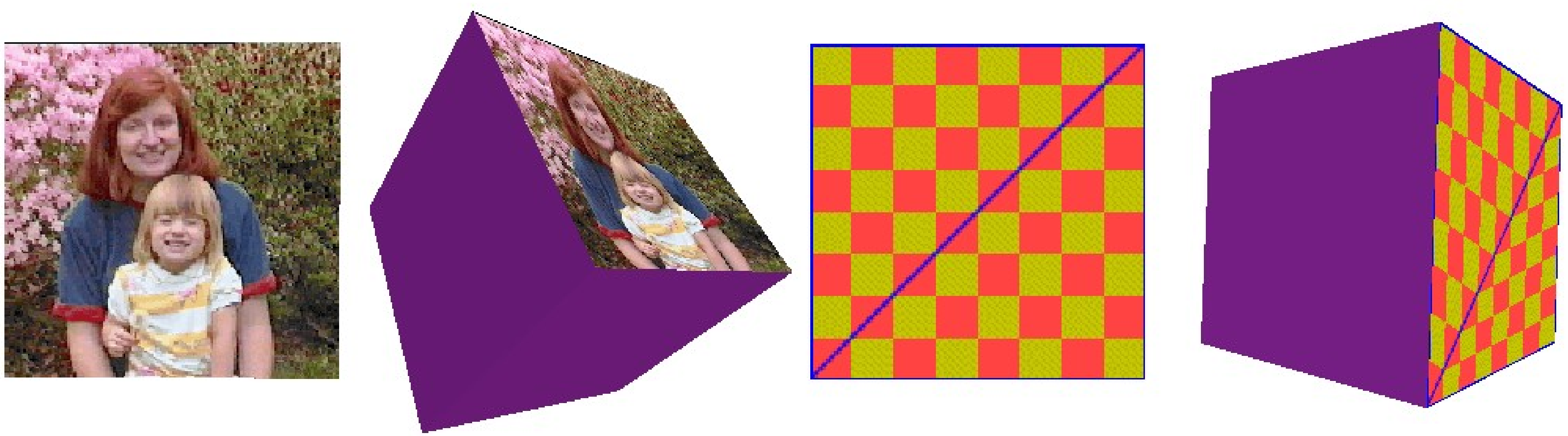
$$I_b = \frac{I_1(y_s - y_3) + I_3(y_1 - y_s)}{y_1 - y_3}$$

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



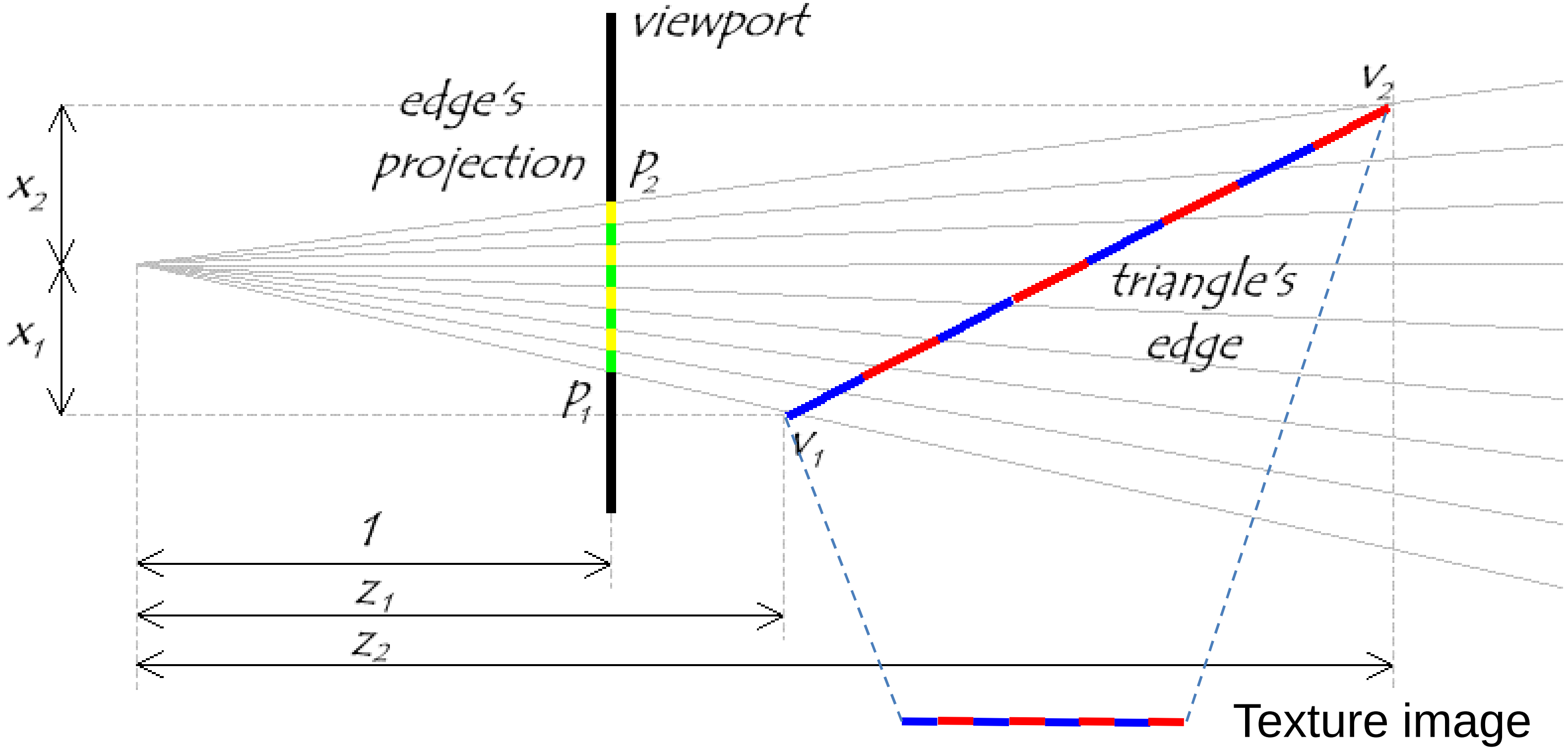
Artifacts

- 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

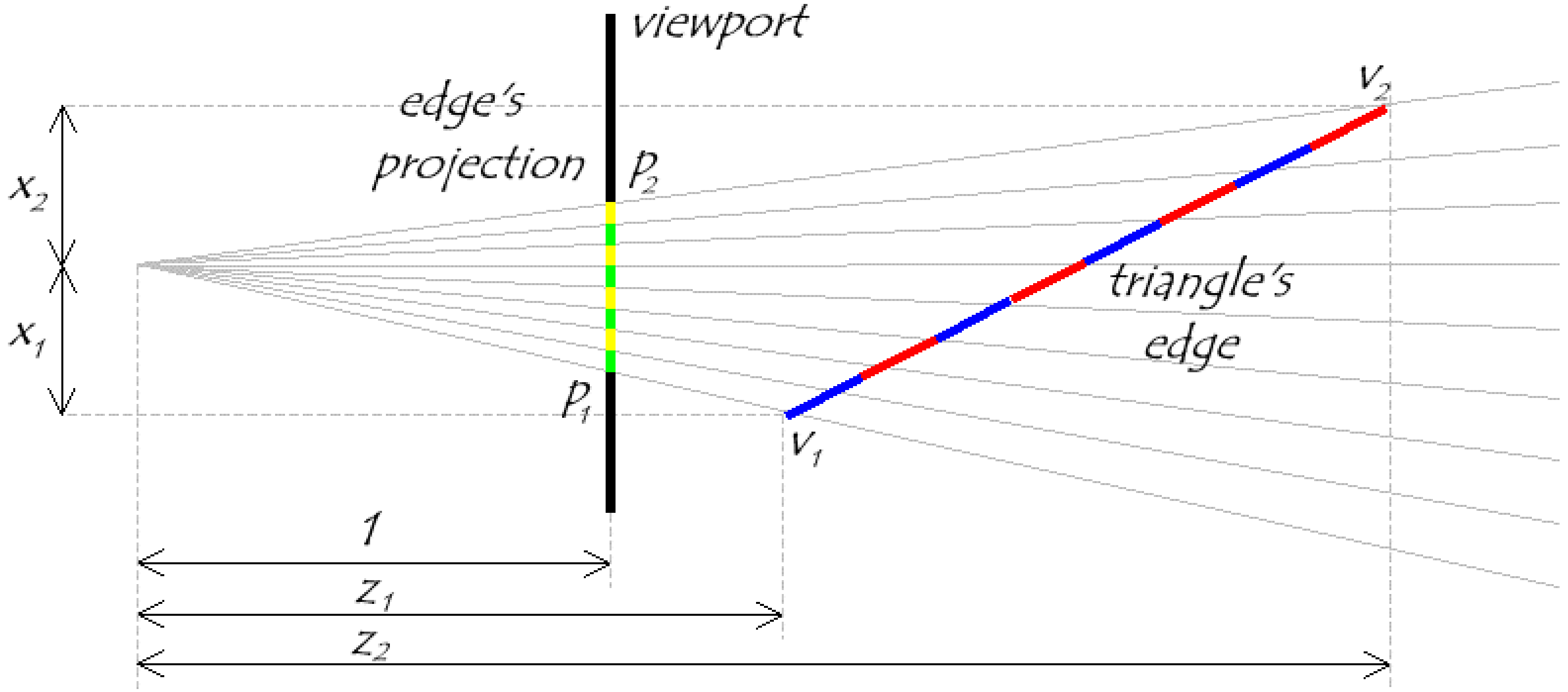


Interpolating Parameters

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



Linear Interpolation in Screen Space



Compare linear interpolation in screen space

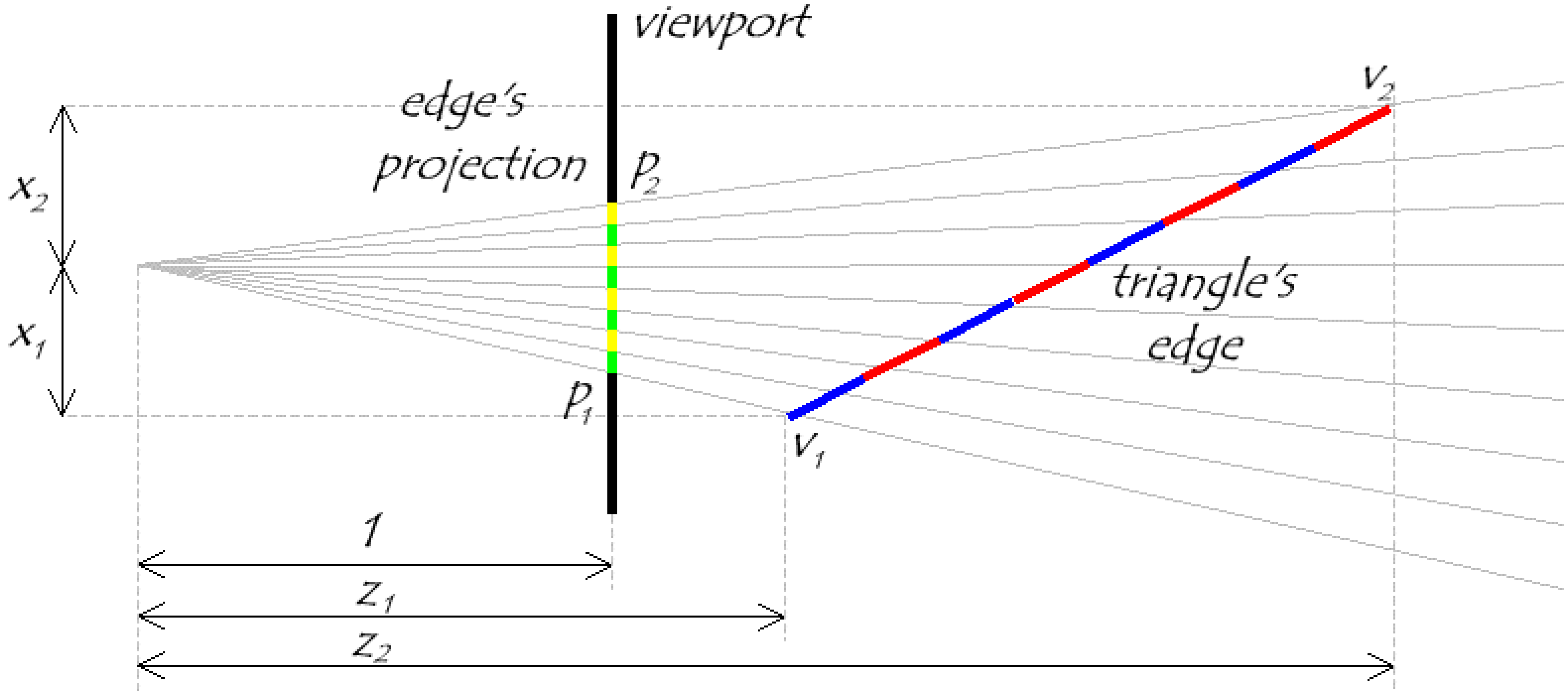
$$p(t) = p_1 + t(p_2 - p_1) = \frac{x_1}{z_1} + t\left(\frac{x_2}{z_2} - \frac{x_1}{z_1}\right)$$

Without loss of generality, let's assume that the image is located 1 unit away from the center of projection. That is

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Slides from Jingyi Yu

Linear Interpolation in 3-Space



to interpolation in 3-space:

$$\begin{bmatrix} x \\ z \end{bmatrix} = \begin{bmatrix} x_1 \\ z_1 \end{bmatrix} + s\left(\begin{bmatrix} x_2 \\ z_2 \end{bmatrix} - \begin{bmatrix} x_1 \\ z_1 \end{bmatrix}\right) \qquad P\left(\begin{bmatrix} x \\ z \end{bmatrix}\right) = \frac{x_1 + s(x_2 - x_1)}{z_1 + s(z_2 - z_1)}$$

Slides from Jingyi Yu

How to make them Mesh

Still need to scan convert in screen space... so we need a mapping from *t* values to *s* values. We know that the all points on the 3-space edge project onto our screen-space line. Thus we can set up the following equality:

$$\frac{x_1}{z_1} + t\left(\frac{x_2}{z_2} - \frac{x_1}{z_1}\right) = \frac{x_1 + s(x_2 - x_1)}{z_1 + s(z_2 - z_1)}$$

and solve for *s* in terms of *t* giving:

$$s = \frac{t\ z_1}{z_2 + t\ (z_1 - z_2)}$$

Unfortunately, at this point in the pipeline (after projection) we no longer have z_1 and z_2 lingering around (Why? Efficiency, don't need to compute 1/z all the time). However, we do have $w_1 = 1/z_1$ and $w_2 = 1/z_2$.

$$s = \frac{t\ \frac{1}{w_1}}{\frac{1}{w_2} + t\ (\frac{1}{w_1} - \frac{1}{w_2})} = \frac{t\ w_2}{w_1 + t\ (w_2 - w_1)}$$

Slides from Jingyi Yu

Interpolating Parameters

We can now use this expression for *s* to interpolate arbitrary parameters, such as texture indices (*u, v)*, over our 3-space triangle. This is accomplished by substituting our solution for *s* given *t* into the parameter interpolation.

$$u = u_1 + s(u_2 - u_1)$$

$$u = u_1 + \frac{t\ w_2}{w_1 + t\ (w_2 - w_1)}(u_2 - u_1) = \frac{u_1 w_1 + t\ (u_2 w_2 - u_1 w_1)}{w_1 + t\ (w_2 - w_1)}$$

Therefore, if we **premultiply all parameters that we wish to interpolate in 3-space by their corresponding *w* value** and add a new plane equation to interpolate the *w* values themselves, we can interpolate the numerators and denominator in screen-space. We then need to perform a divide a each step to get to map the screen-space interpolants to their corresponding 3-space values. This is a simple modification to the triangle rasterizer that we developed in class.

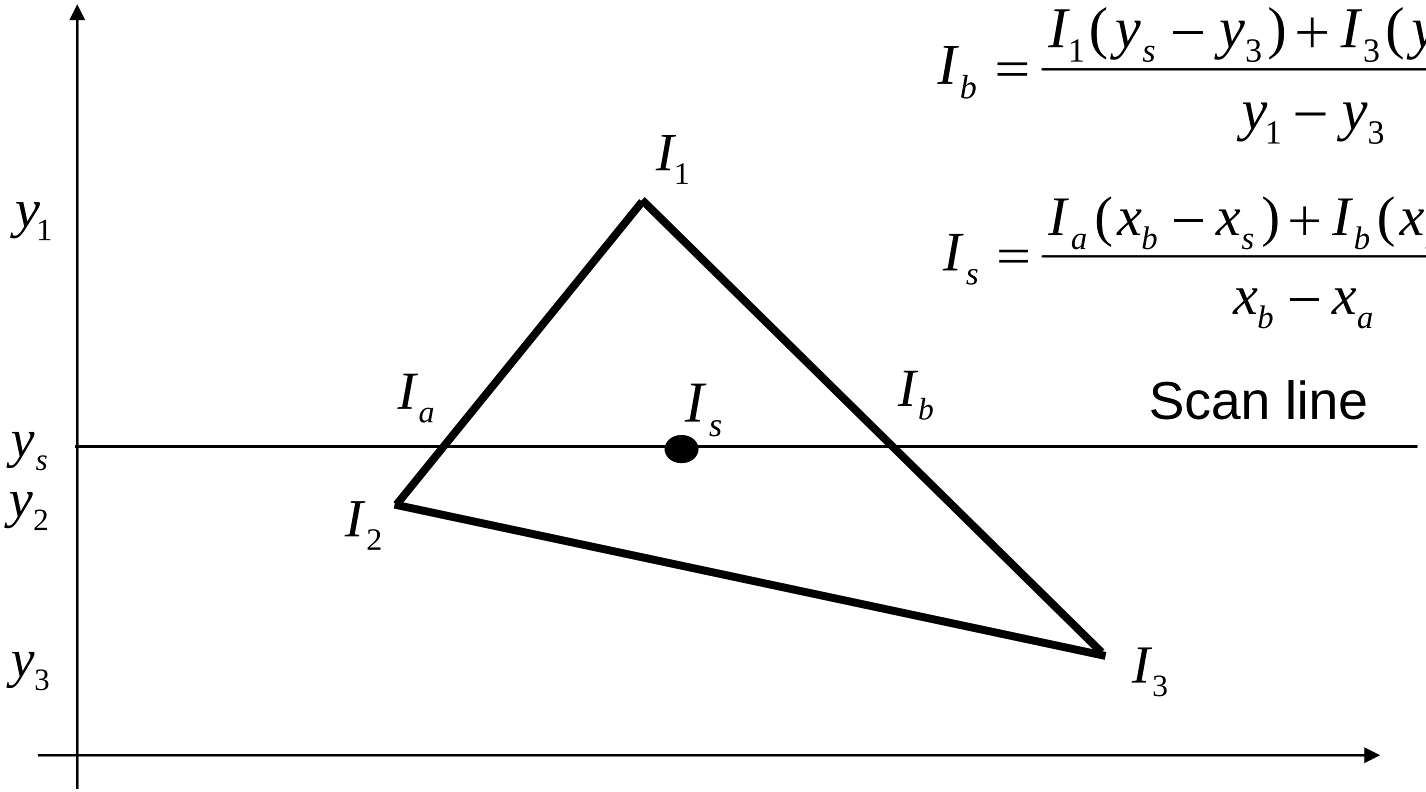
Slides from Jingyi Yu

1st idea: Gouraud interp. of texcoords

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

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

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

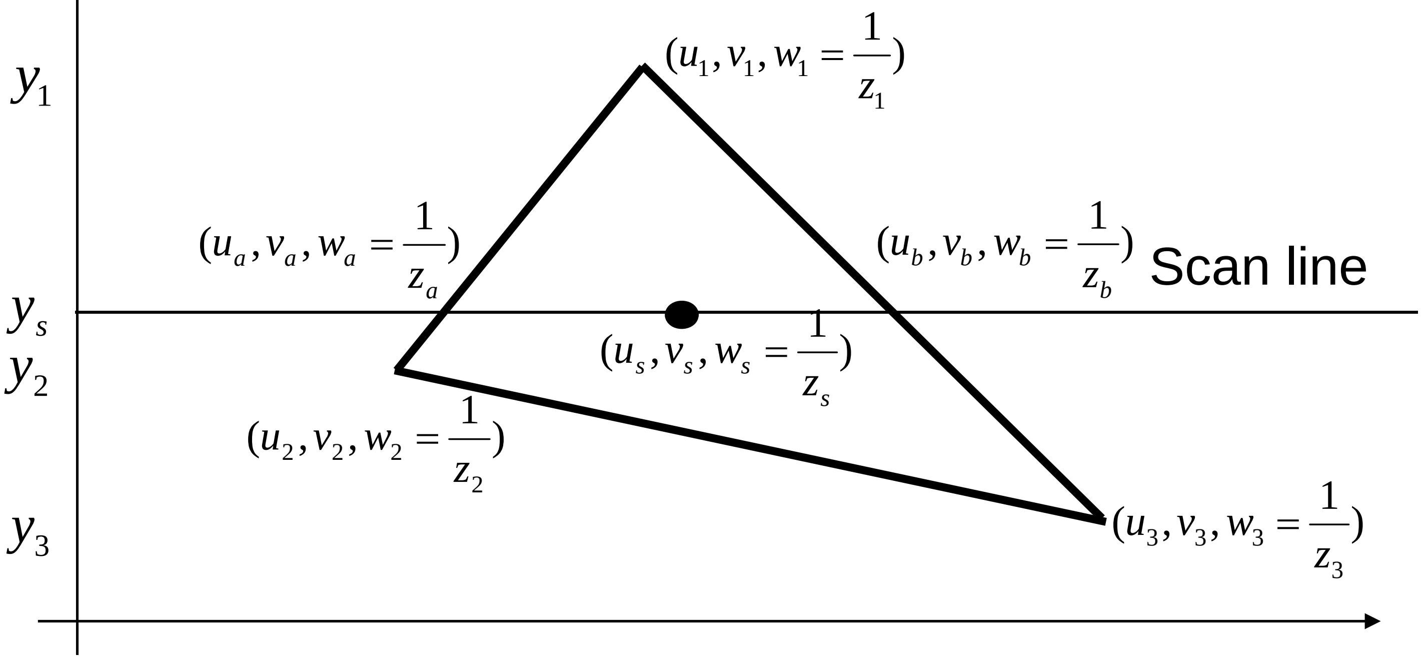


Replace I to uw, vw, and w, then compute (uw/w, and vw/w)

1st idea: Gouraud interp. of texcoords

$$(uw)_a = \frac{u_1 w_1 (y_s - y_2) + u_2 w_2 (y_1 - y_s)}{y_1 - y_2} \qquad (vw)_a = \frac{v_1 w_1 (y_s - y_2) + v_2 w_2 (y_1 - y_s)}{y_1 - y_2}$$

$$w_a = \frac{w_1 (y_s - y_2) + w_2 (y_1 - y_s)}{y_1 - y_2} \qquad u_a = \frac{(uw)_a}{w_a} \qquad v_a = \frac{(vw)_a}{w_a}$$



Do same thing for point b. From a and b, interpolate for s