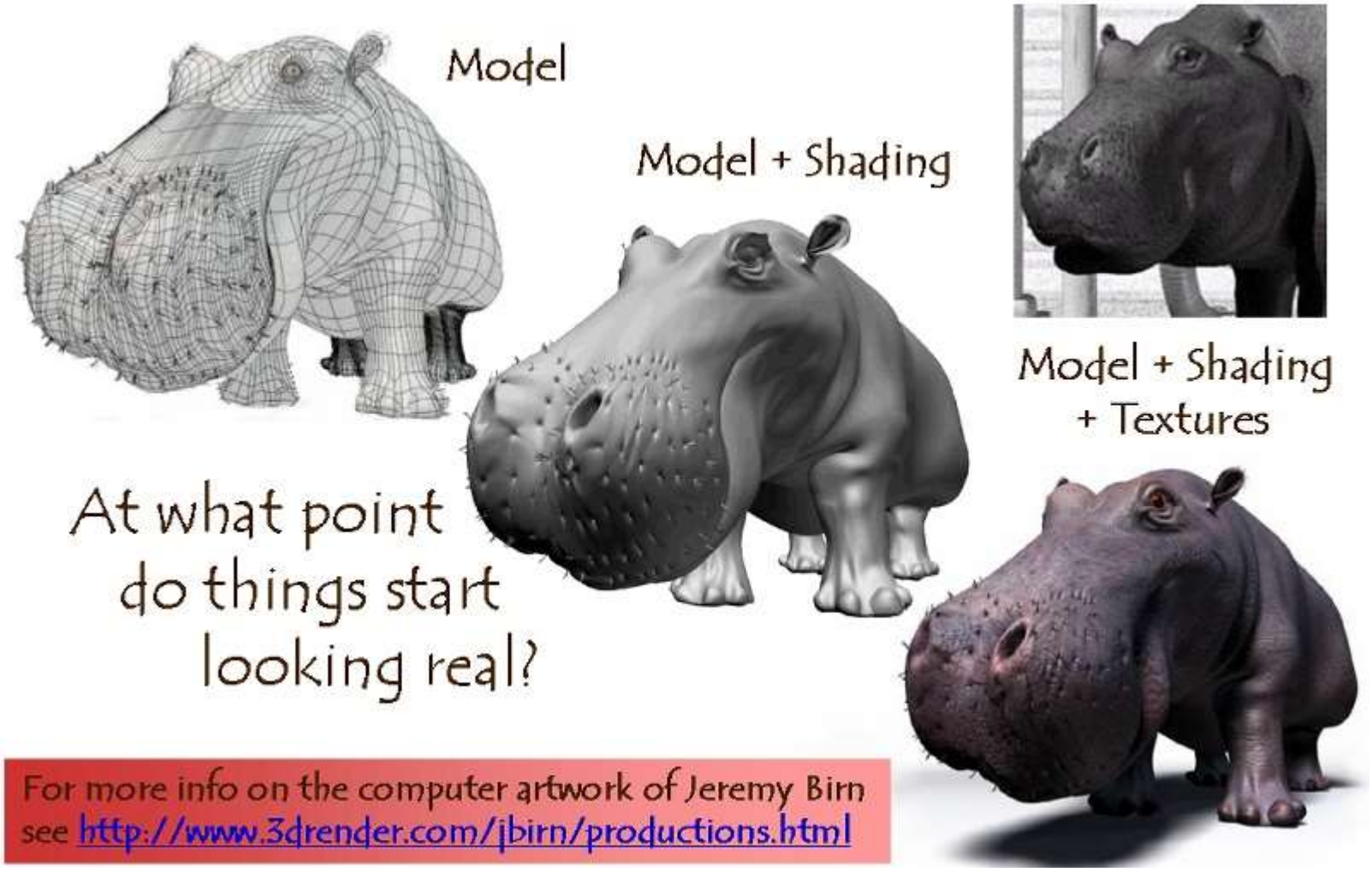
Texture Mapping

The Quest for Visual Realism

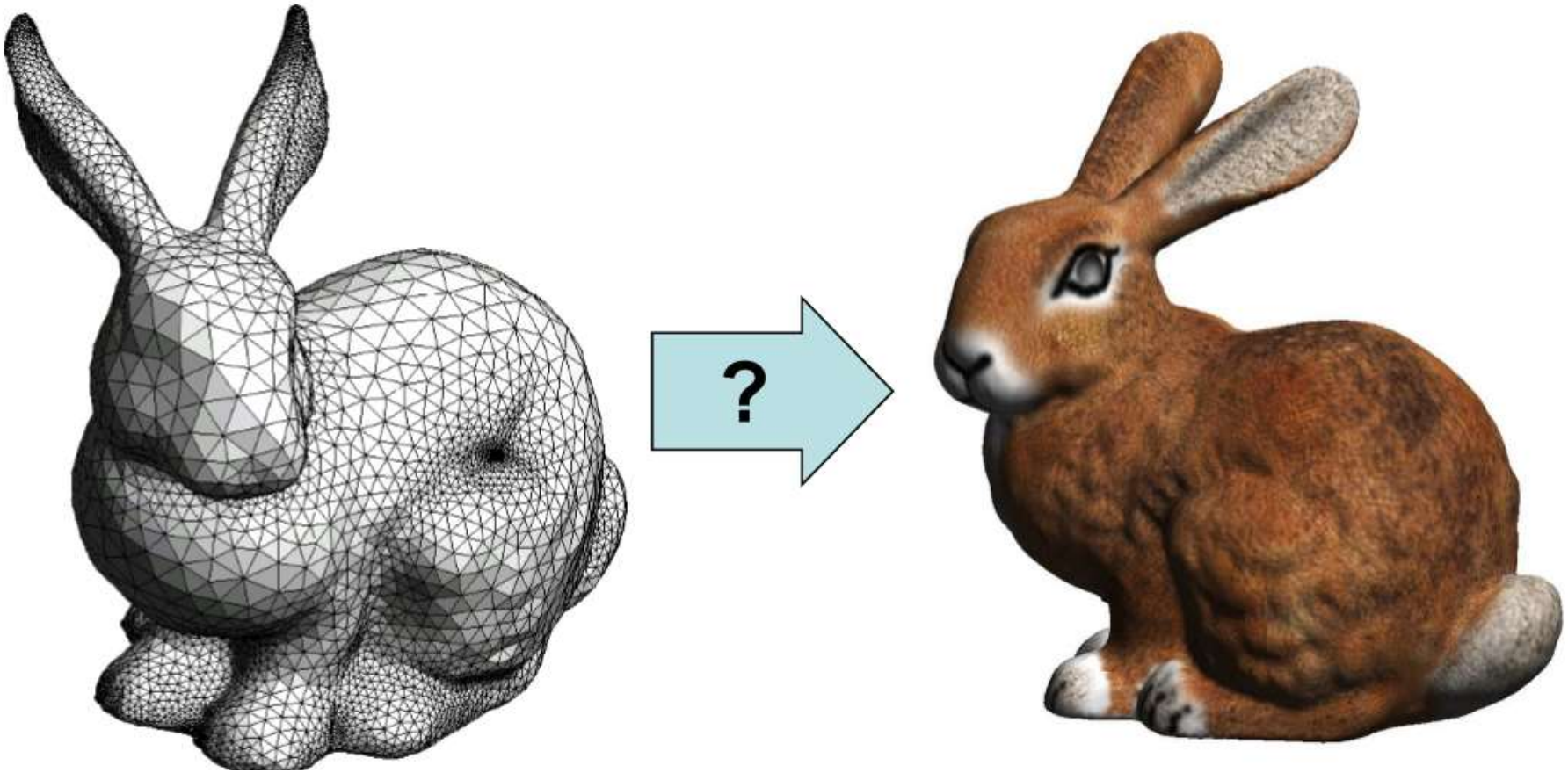


Some slides are from Leonard McMillan and others

Texture Mapping

3D model

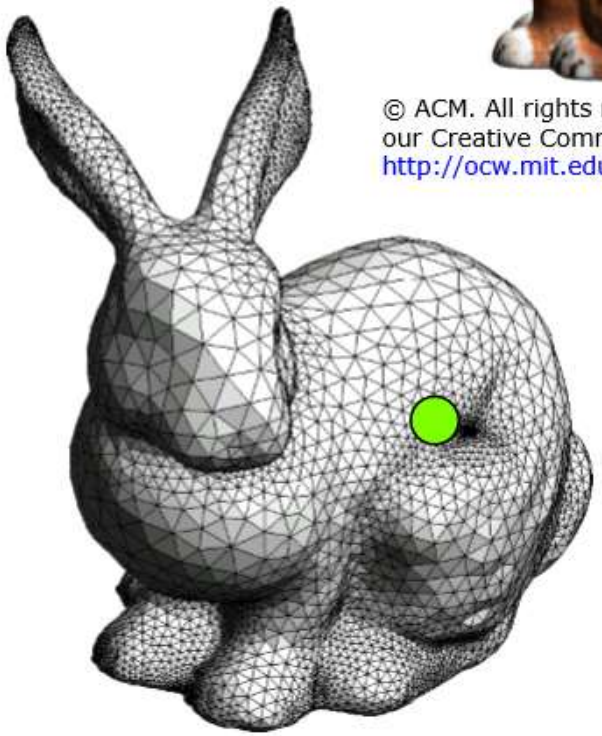
Texture mapped model



Texture Mapping

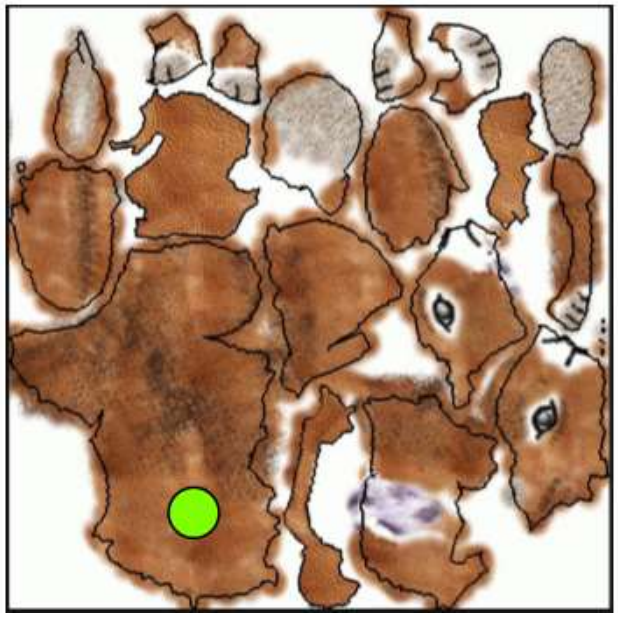
Texture mapped model

We need a function that associates each surface point with a 2D coordinate in the texture map



?

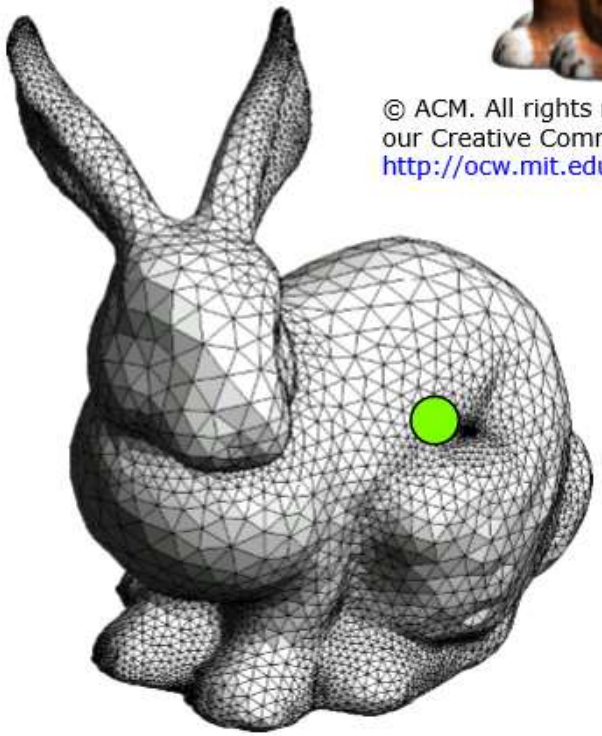
Texture map (2D image)



Texture Mapping

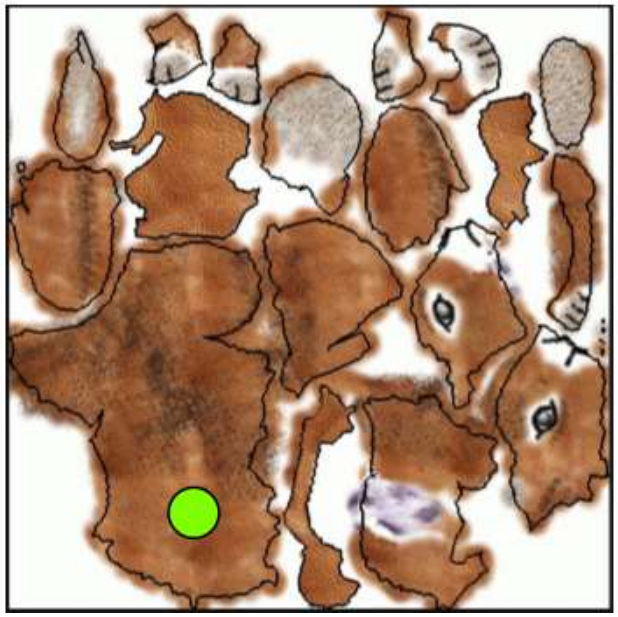
Texture mapped model

For each point rendered, look up color in texture map



?

Texture map (2D image)



UV Coordinates

- Each vertex P stores 2D (u, v) "texture coordinates"
  - UVs determine the 2D location in the texture for the vertex
  - We will see how to specify them later
- Then we interpolate using barycentrics

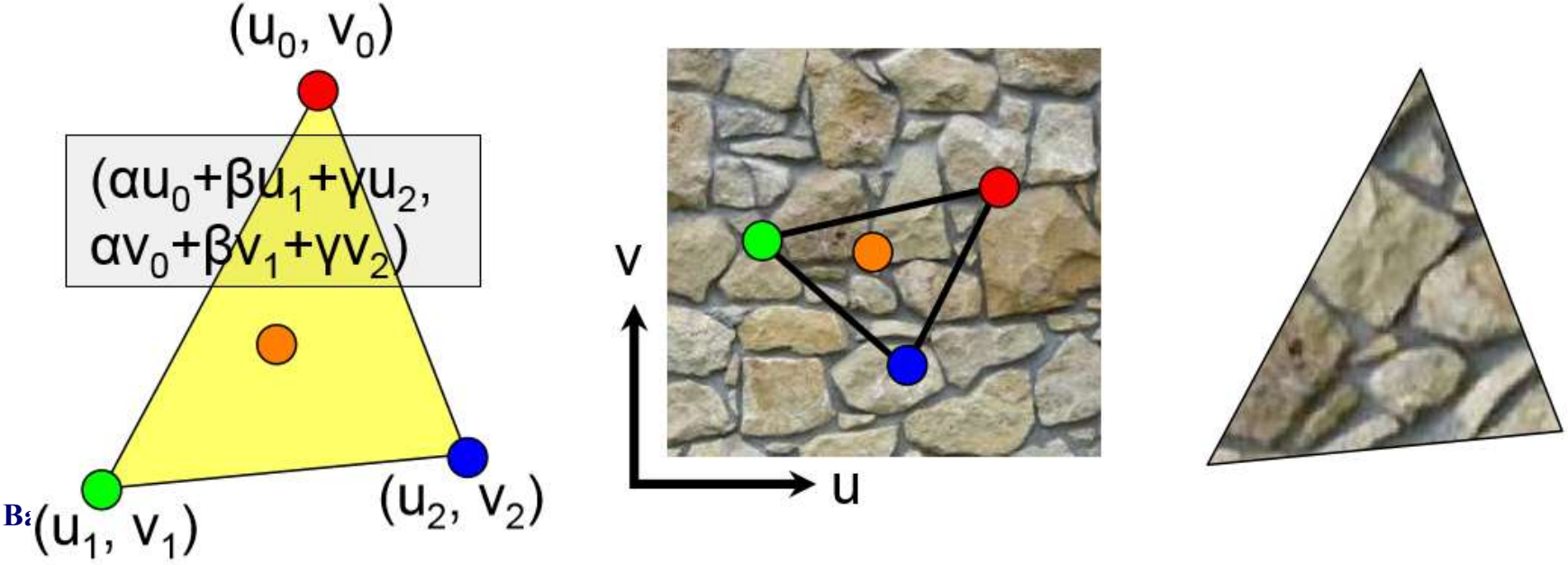
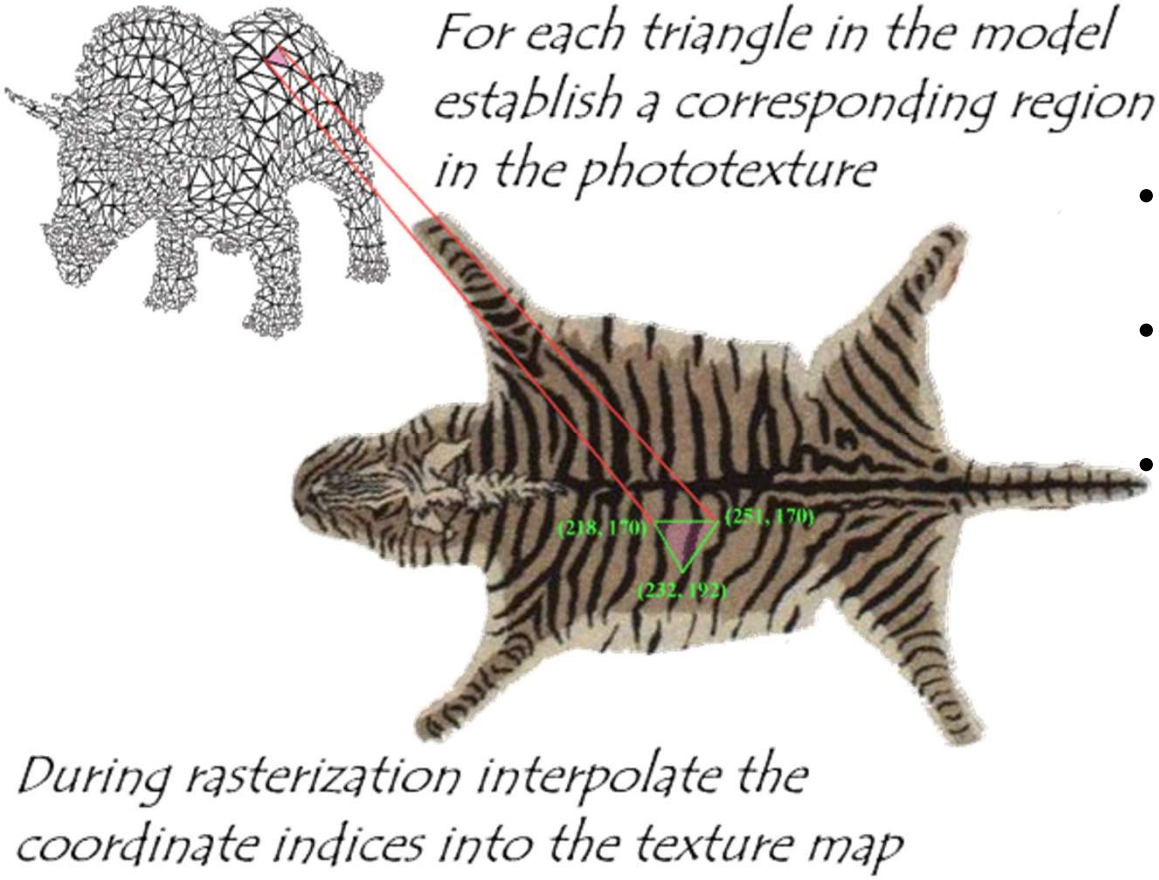
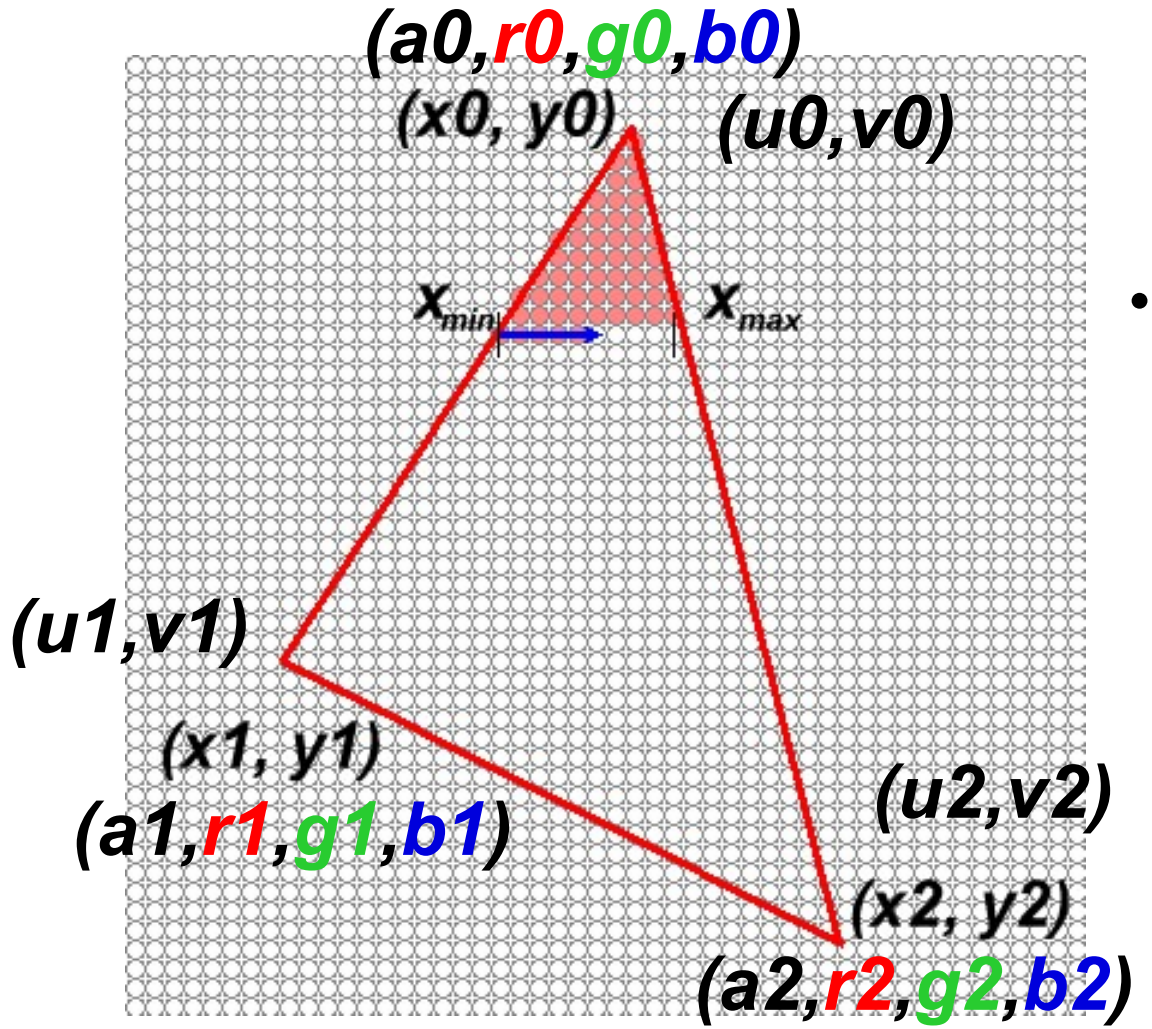


Photo-textures



- Specify a texture coordinate at each vertex (s, t) or (u, v)
- Canonical coordinates where u and v are between 0 and 1
- Simple modifications to triangle rasterizer

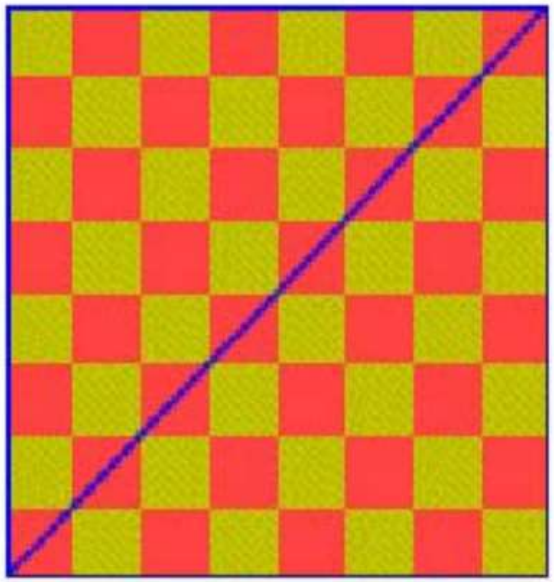
Texture Interpolation

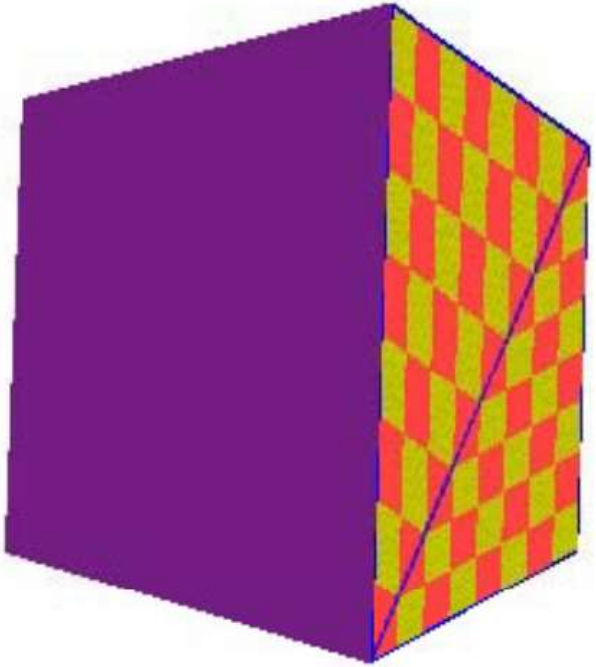


- Linear Interpolation

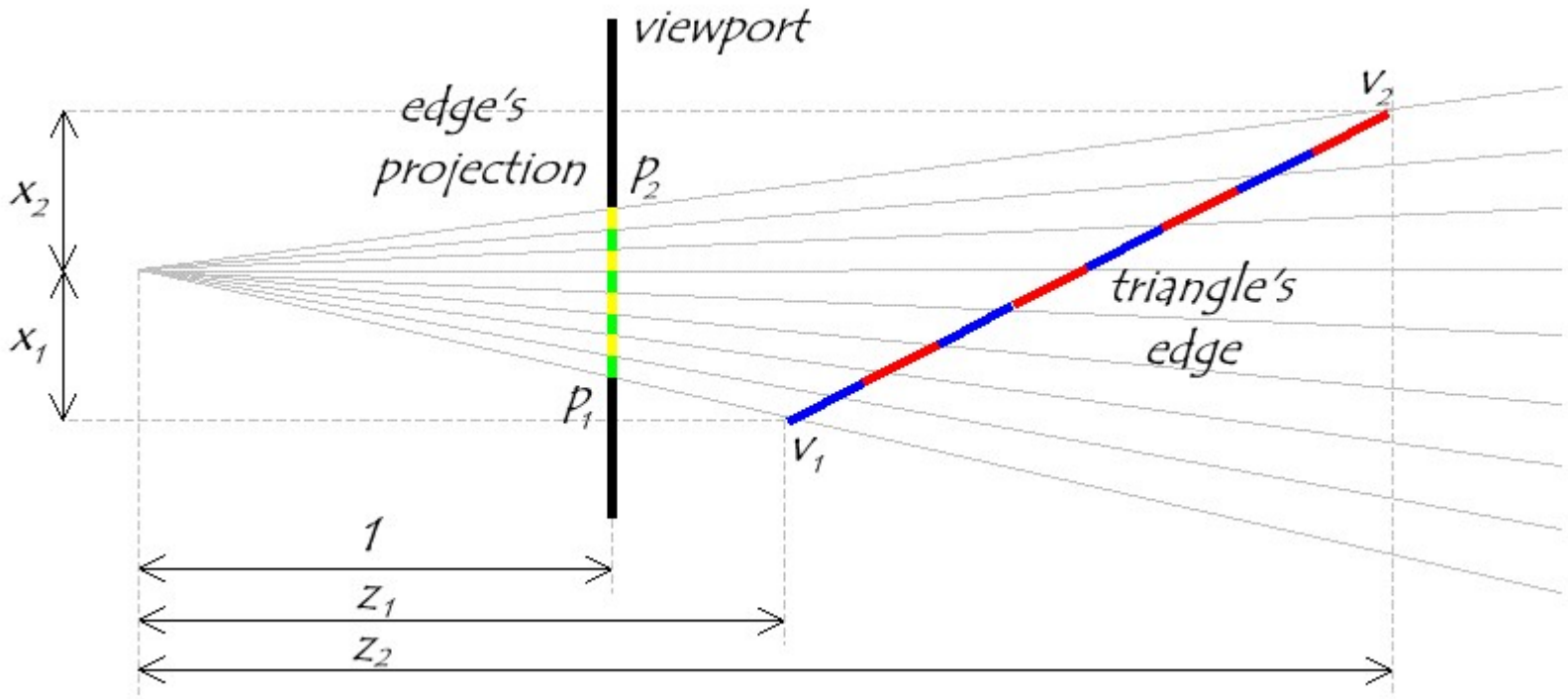
Texture Mapping Artifacts I

- **Simple linear interpolation of u and v over a triangle in a screen space leads to unexpected results unexpected results**
  - Distorted when the triangle's vertices do not have the same depth





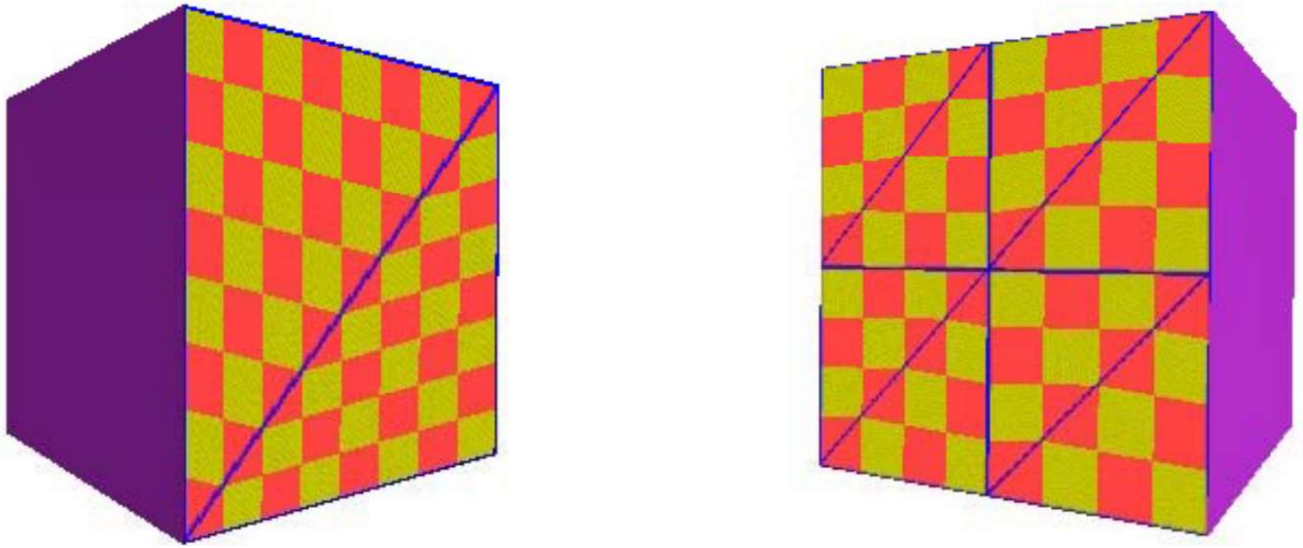
Visualizing the Problem



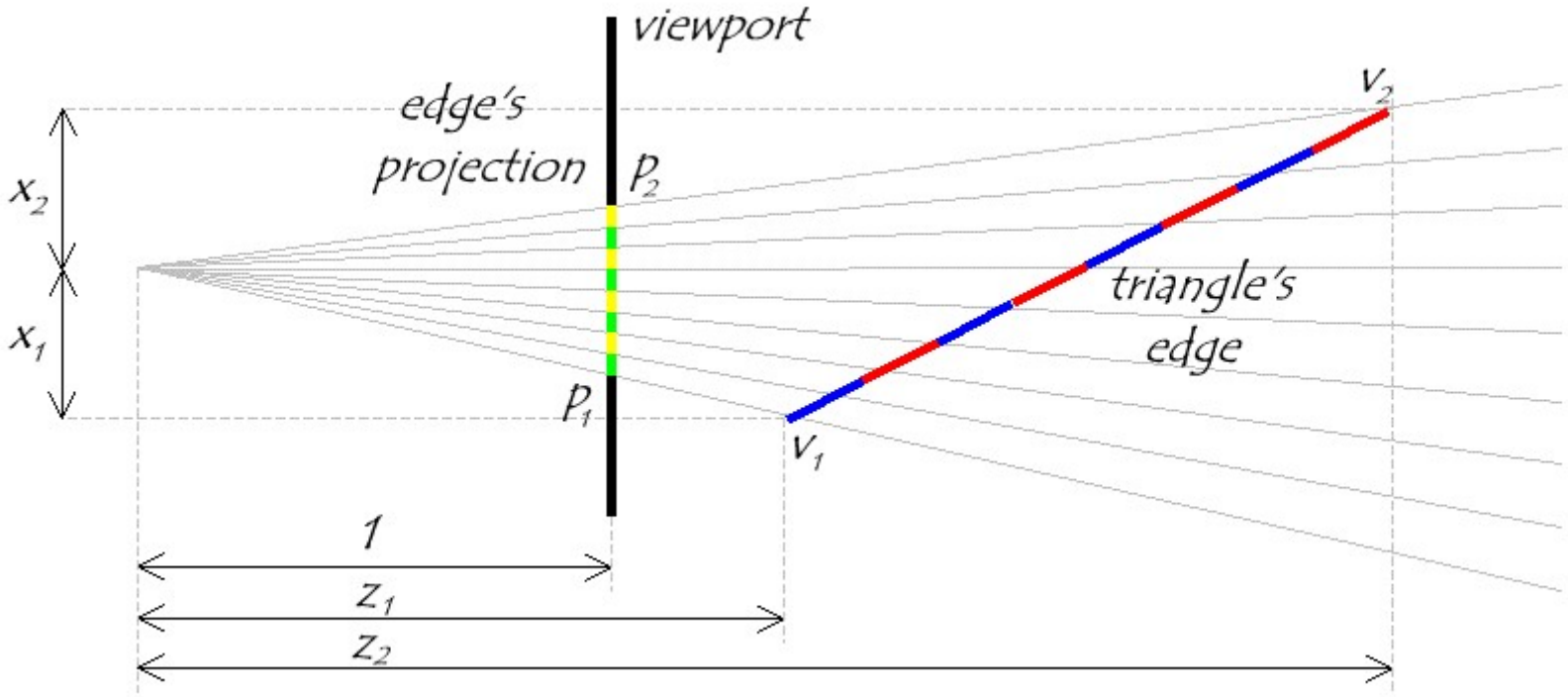
Notice that uniform steps on the image plane do not correspond to uniform steps along the edge.

An Approximation Approach

You can reduce the perceived artifacts of non-perspective correct interpolation by subdividing the texture-mapped triangles into smaller triangles (why does this work?). But, fundamentally the screen-space interpolation of projected parameters is inherently flawed.



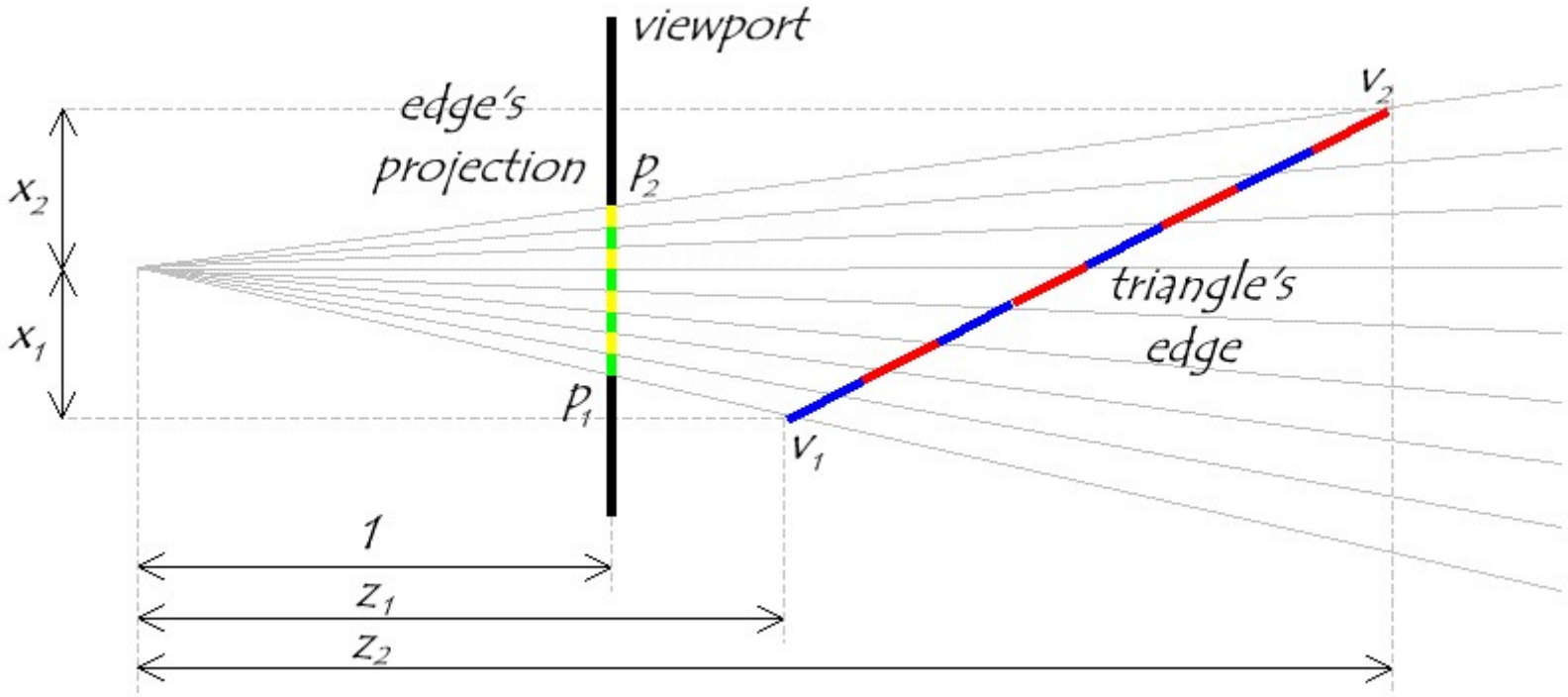
Linear Interpolation in Screen Space



Compare interpolation in screen space

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

Linear Interpolation in 3-Space



to interpolation in 3-space

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

Come Back to Screen Interpolation

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 follow 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?). However, we do have* $w_1 = 1/z_1$ *and* $w_2 = 1/z_2$.

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

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 substituting our solution for s given t into the parameter interpolation.

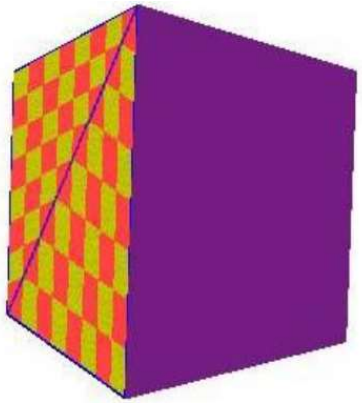
$$u = u_1 + s(u_2 - u_1)$$

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

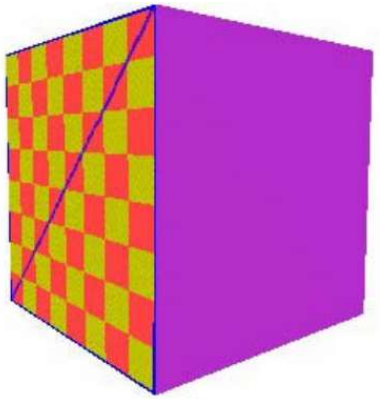
Therefore, if we **pre-multiply all parameters that we wish to interpolate in 3-space by their corresponding *w value* and add a new plane equation to interpolate the** w values themselves, we can interpolate the numerators and denominator in screen-space. We then need to perform a divide at each step to get to map the screen-space interpolants to their corresponding 3-space values.

Perspective-Correct Interpolation

- This method of interpolation is called perspective-correct interpolation
  - Actually it is simply correct interpolation correct interpolation
  - Not all 3D graphics APIs implement perspective-correct interpolation

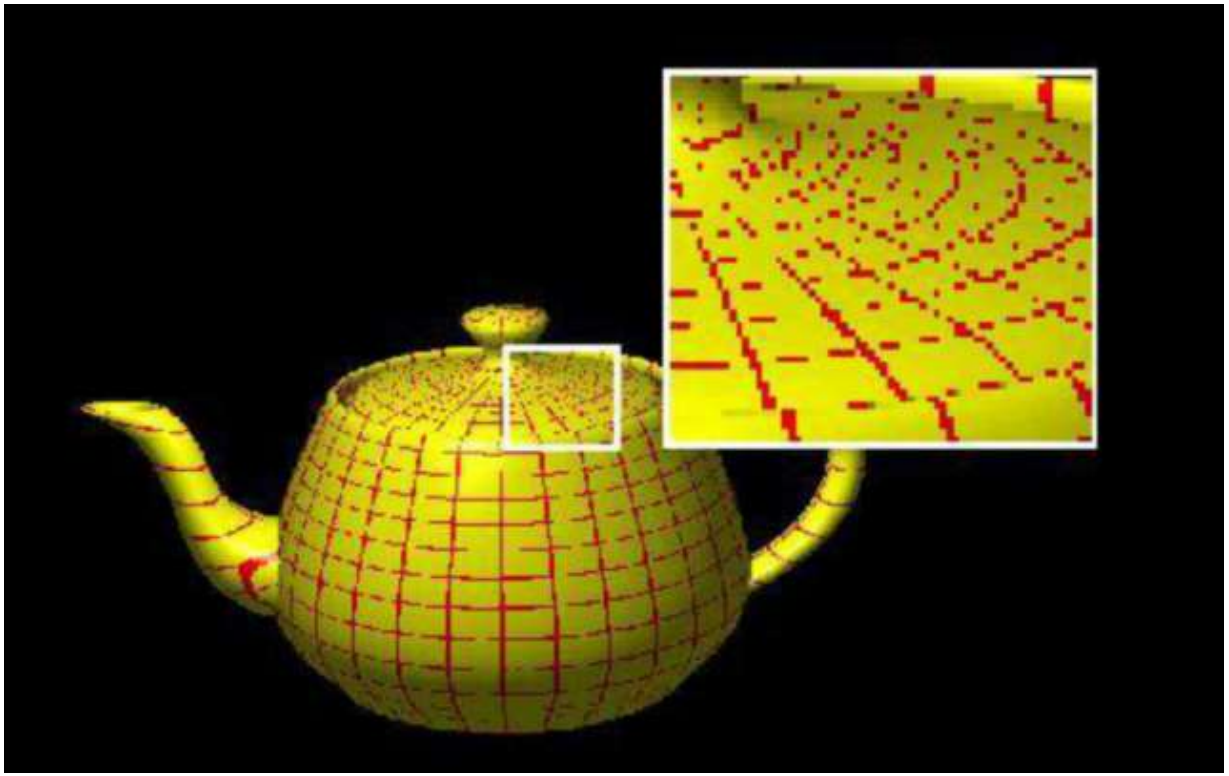


Linear interpolation

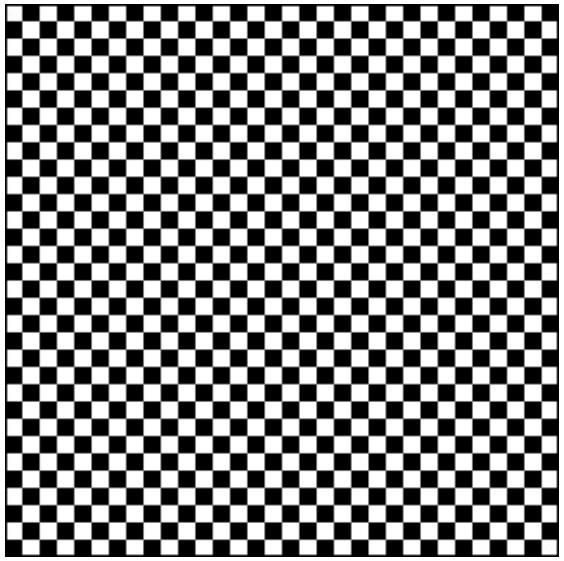


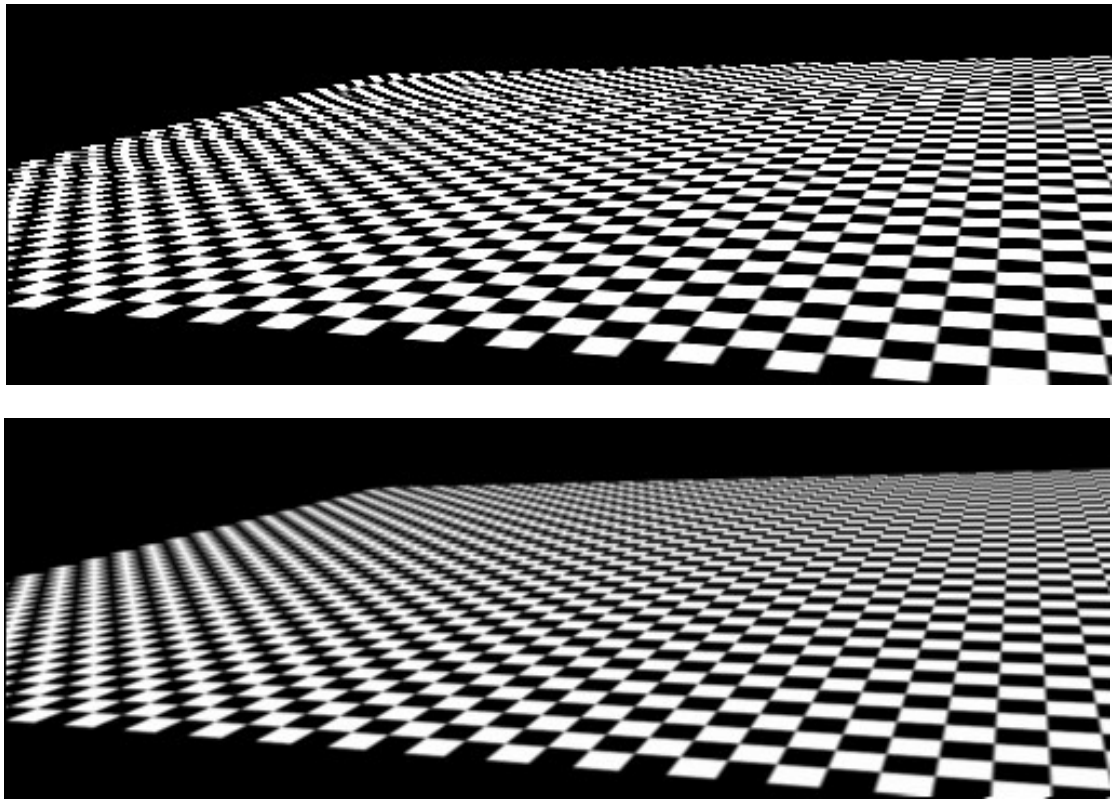
Perspective-correct interpolation

Texture Mapping Artifacts II



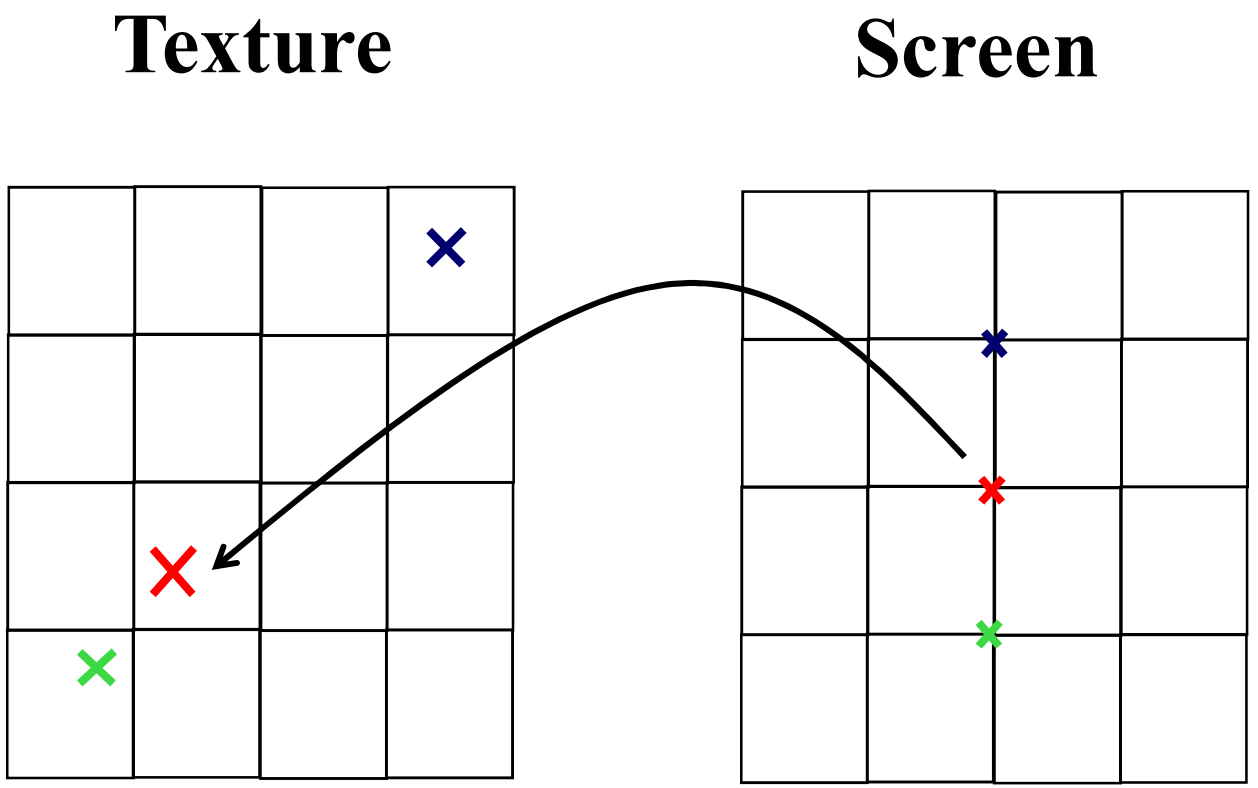
Texture Mapping Artifacts II



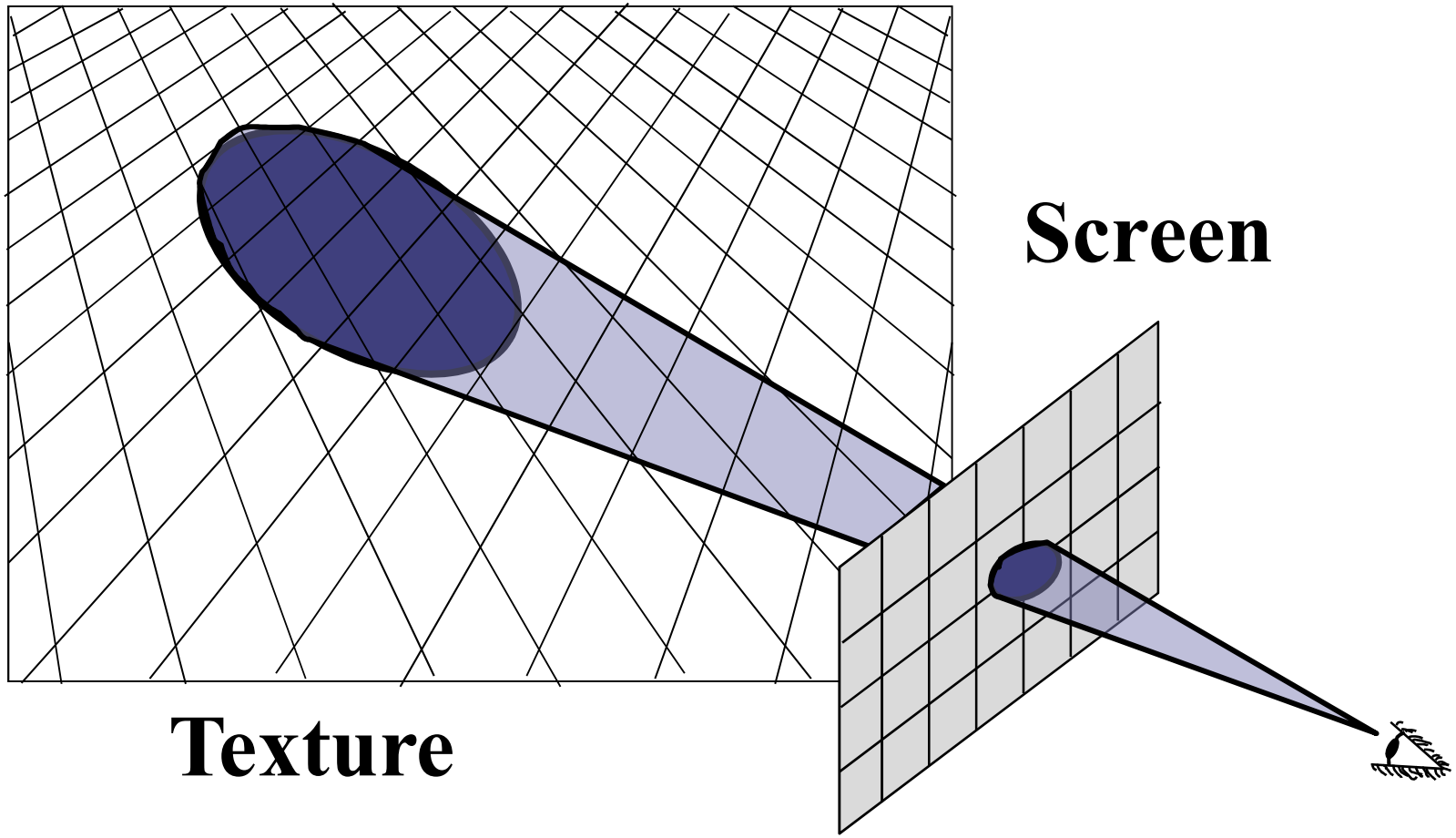


CLICK

Texture Resampling



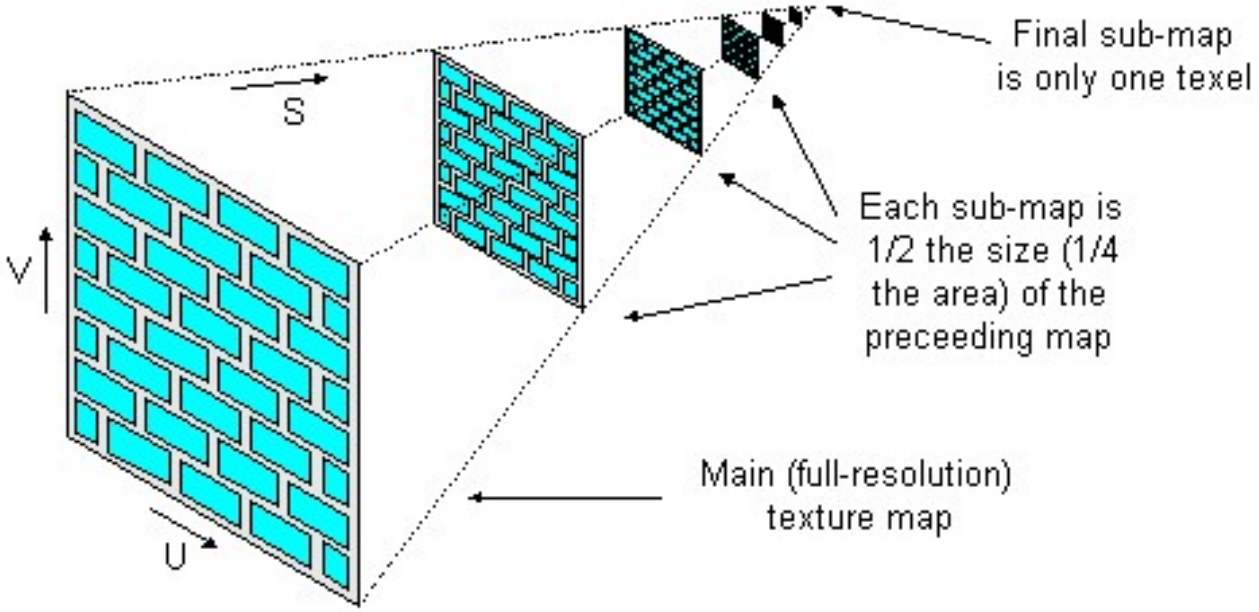
High Quality Texture Mapping



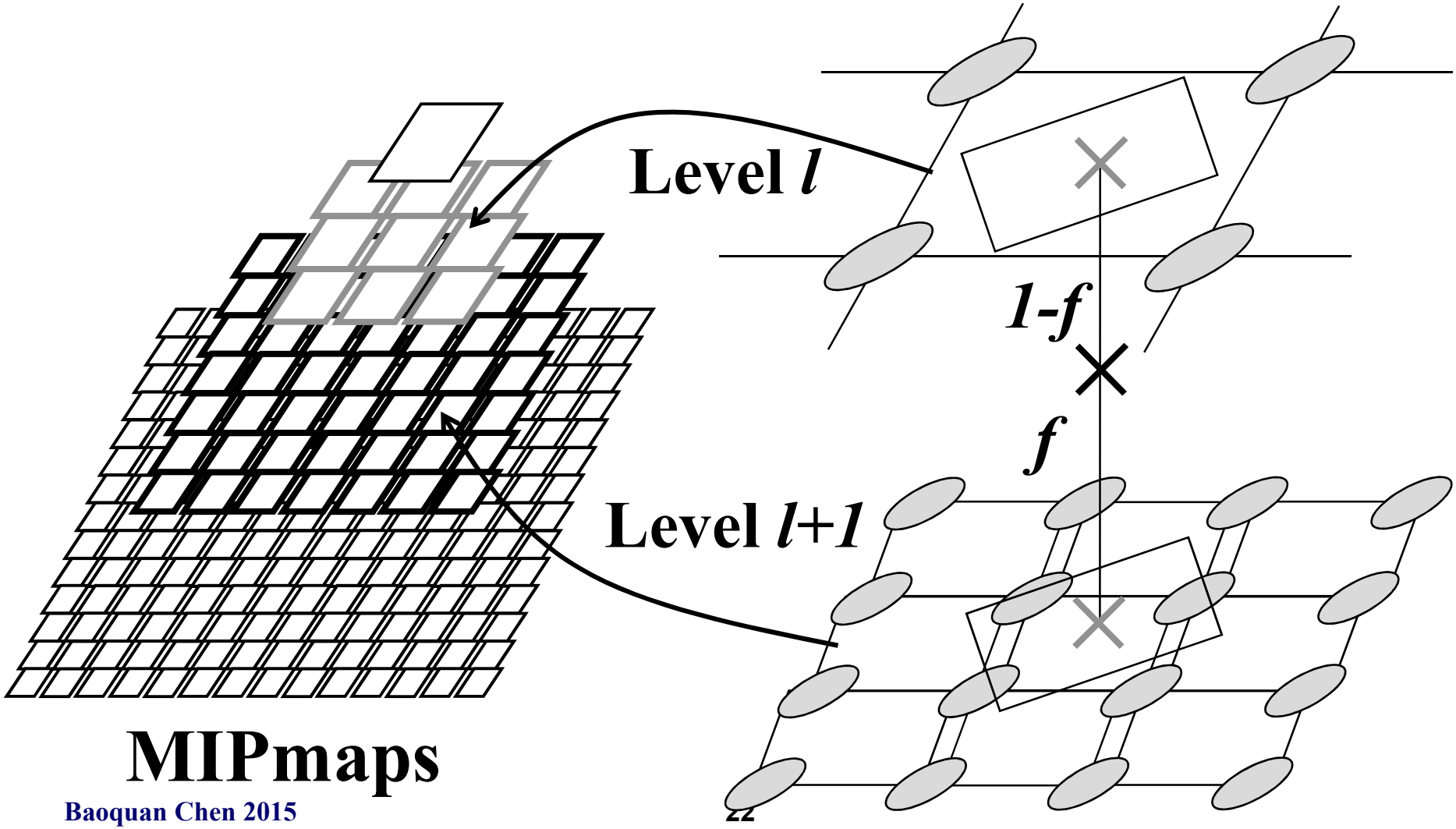
MIP Mapping

MIP Mapping is one popular technique for antialiasing in texture mapping. MIP is an acronym for the latin phrase multium in parvo, which means "many in a small place". The technique was first described by Lance Williams. The basic idea is to construct a pyramid of images that are prefiltered and resampled at resolutions that are a binary fractions (1/2, 1/4, 1/8, etc) of the original image's resolution.

While rasterizing we compute the index of the image pyramid level that has resolution *closest* to that of our desired screen resolution; in practice, two closest levels, rather than only one, are picked up and an interpolation between the two levels is performed).

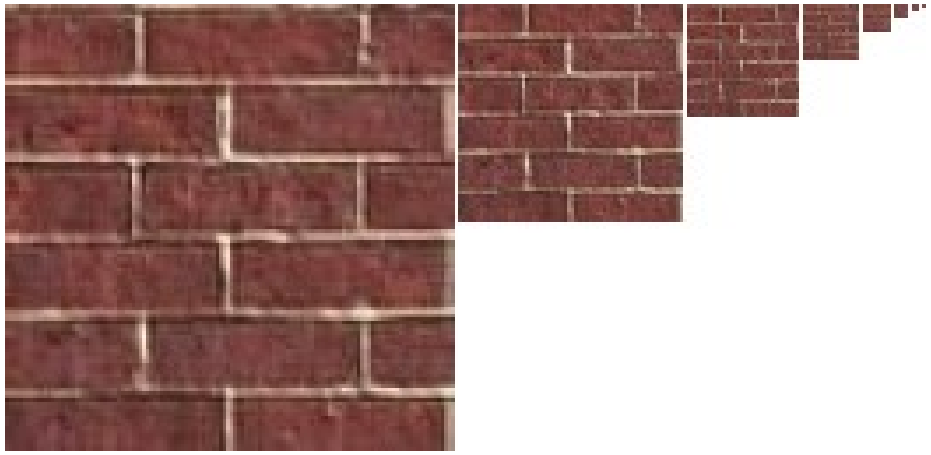


MIP Mapping



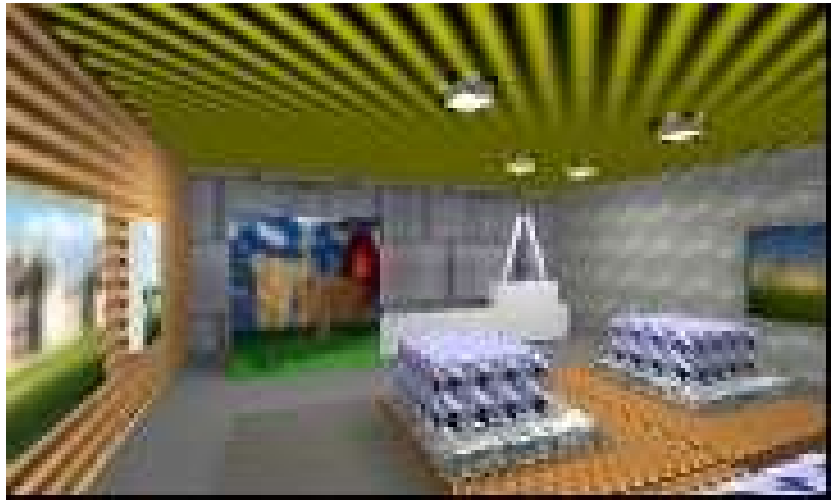
MIP Mapping

Computing this series of filtered images requires only a small fraction of additional storage over the original texture (How small of a fraction?).



Comparison

Nearest neighbor interpolation



MIP-mapping

Finding the MIP level

What we'd like to find is the step size that a uniform step in screen-space causes in three-space, or, in other words how a screen-space change relates to a 3-space change. This sounds like the derivatives, (du/dt, dv/dt). They can be computed simply using the chain rule:

$$\frac{du}{dt} = \frac{du}{ds}\frac{ds}{dt} = (u_2 - u_1)\frac{w_1 w_2}{(w_1 + t(w_2 - w_1))^2}$$

$$\frac{dv}{dt} = (v_2 - v_1)\frac{w_1 w_2}{(w_1 + t(w_2 - w_1))^2}$$

Notice that the term being squared under the numerator is just the w plane equation that we are already computing. The remaining terms are constant for a given rasterization. Thus all we need to do to compute the derivative is a square the w accumulator and multiply it by a couple of constants.

Now, we know how a step in screen-space relates to a step in 3-space. So how do we translate this to an index into our MIP table?