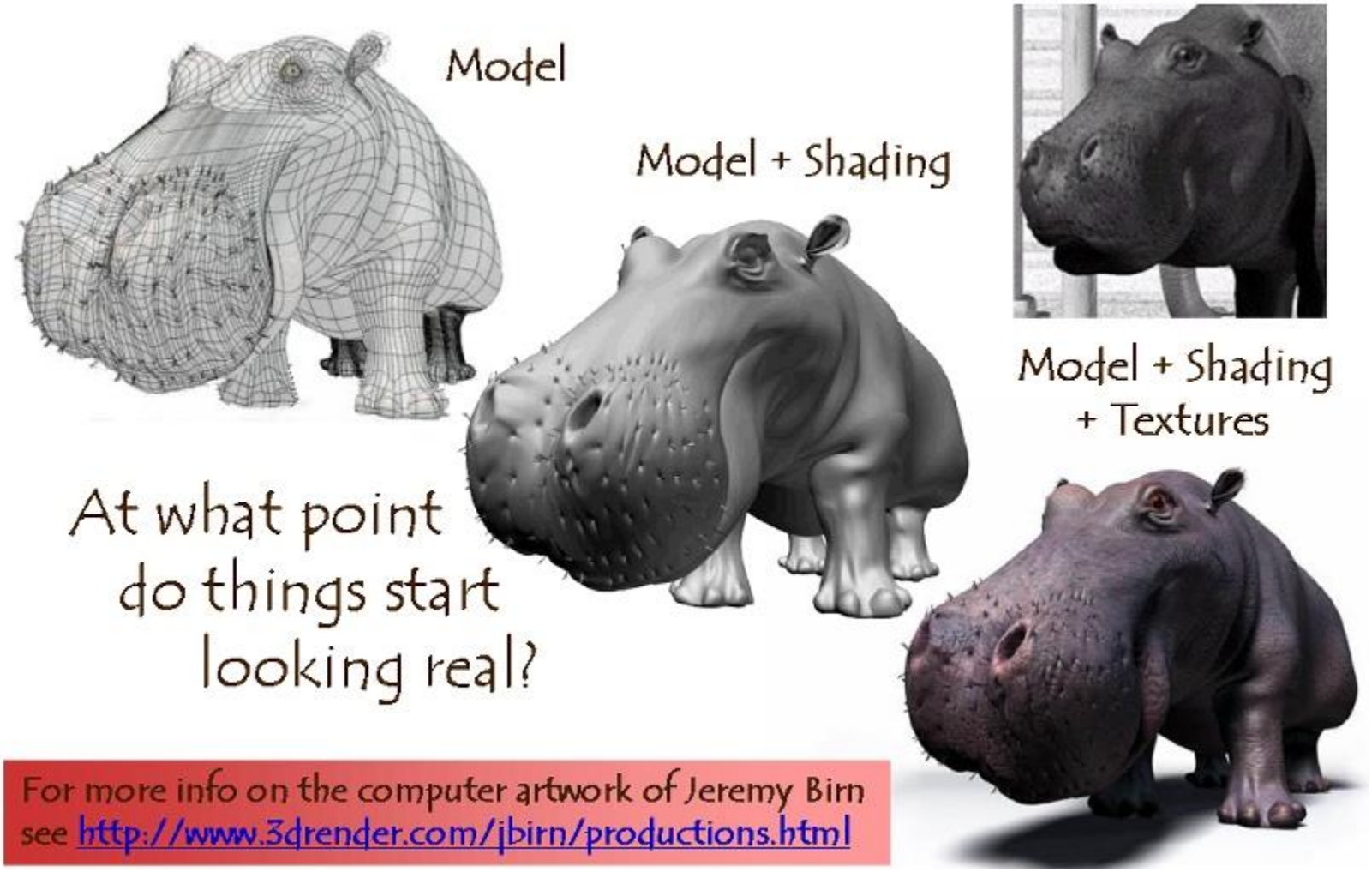
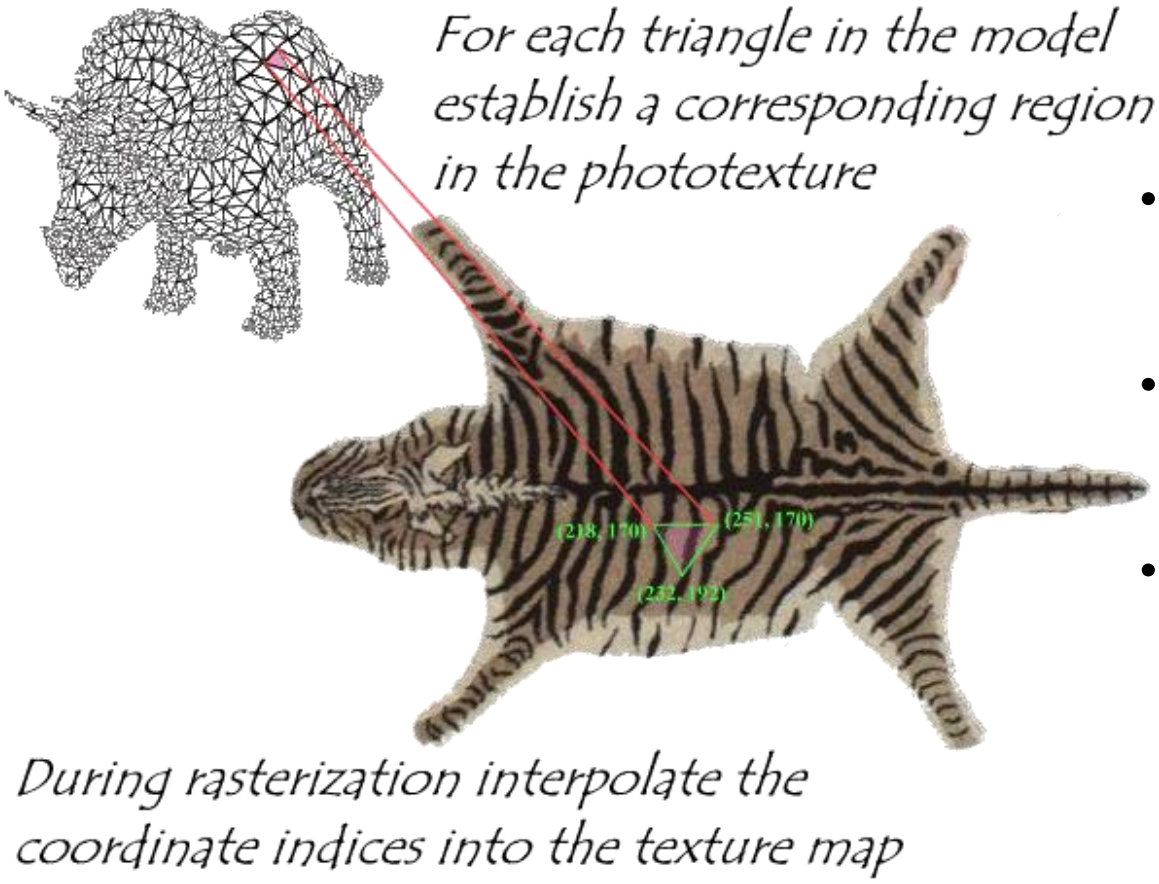
Texture Mapping

The Quest for Visual Realism



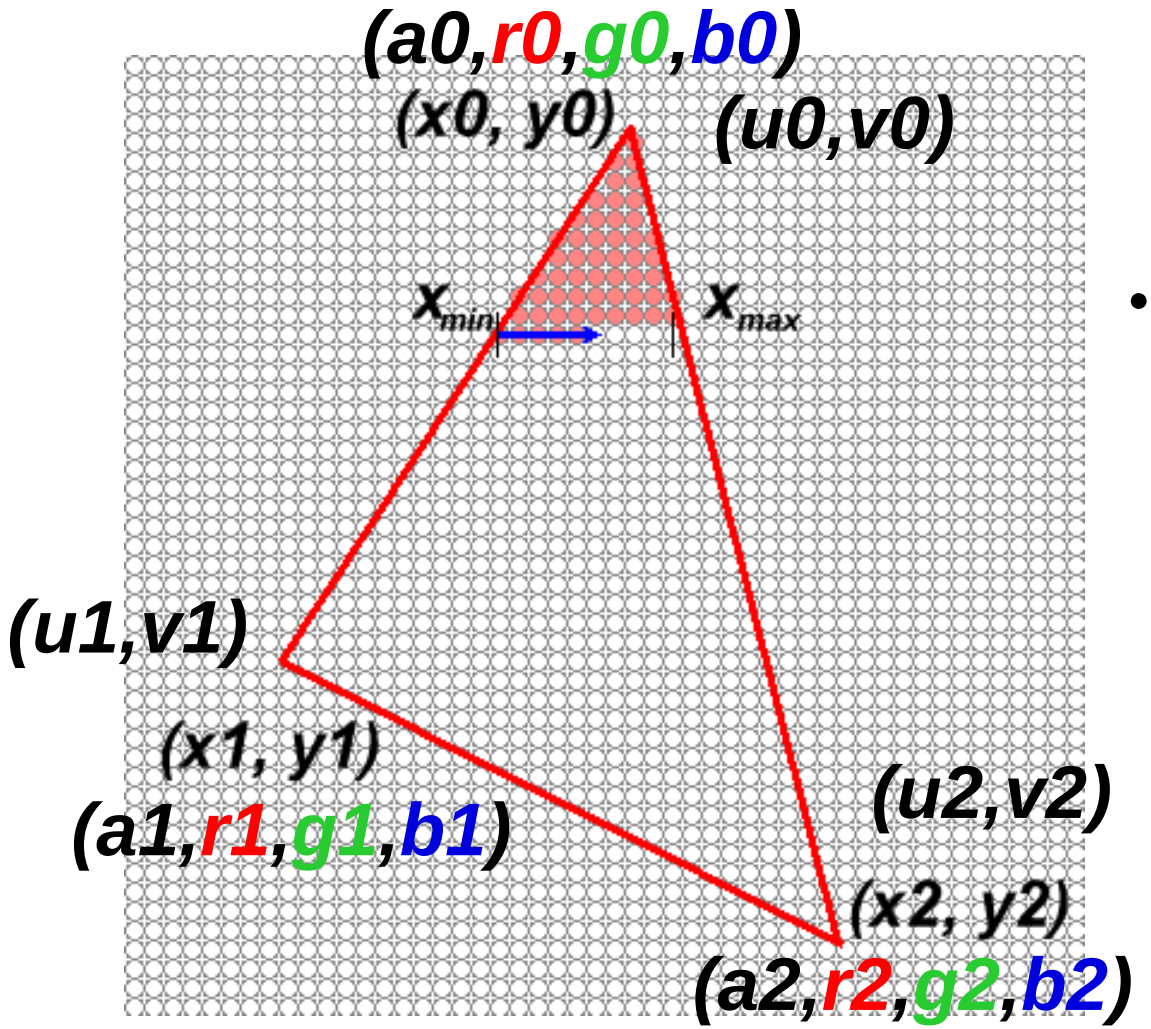
Some slides are from Leonard McMillan and others

Photo-textures



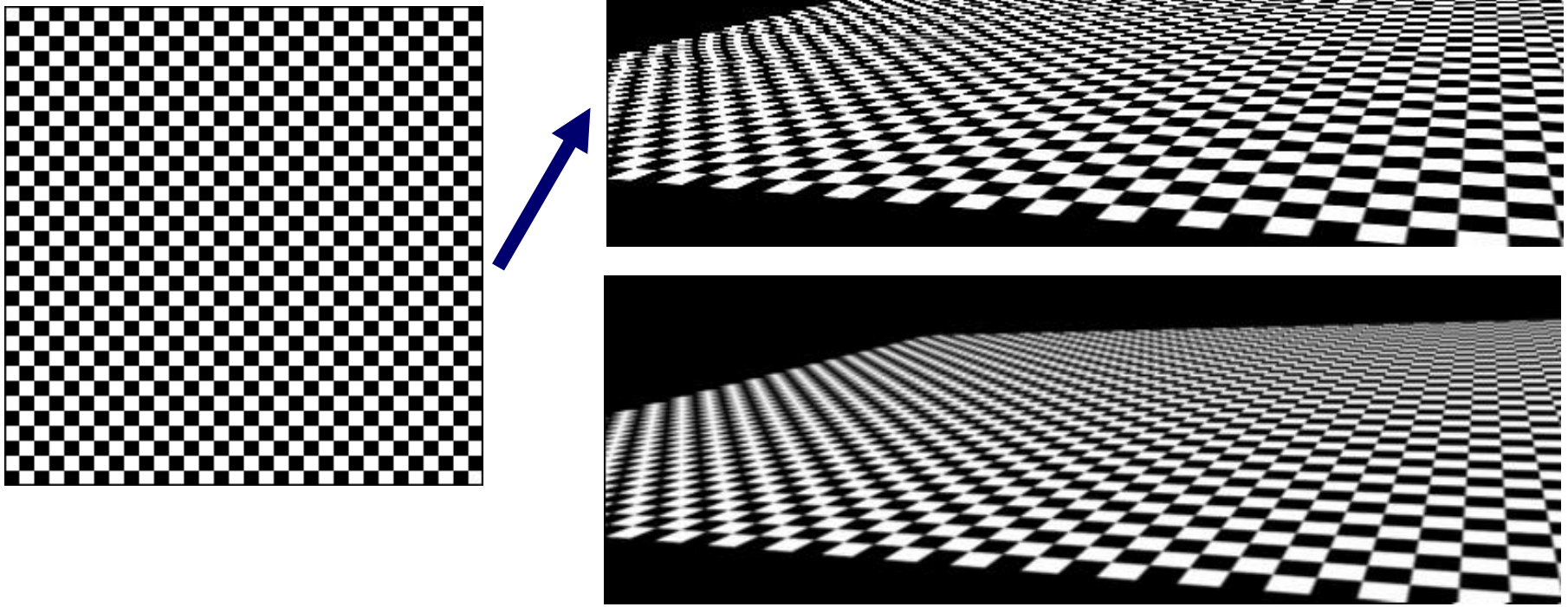
- Specify a texture coordinate at each vertex (s, t) or (u, v)
- Canonical coordinates where u and v are between 0 and 1
- Simple modifications to triangle rasterizer

Texture Interpolation

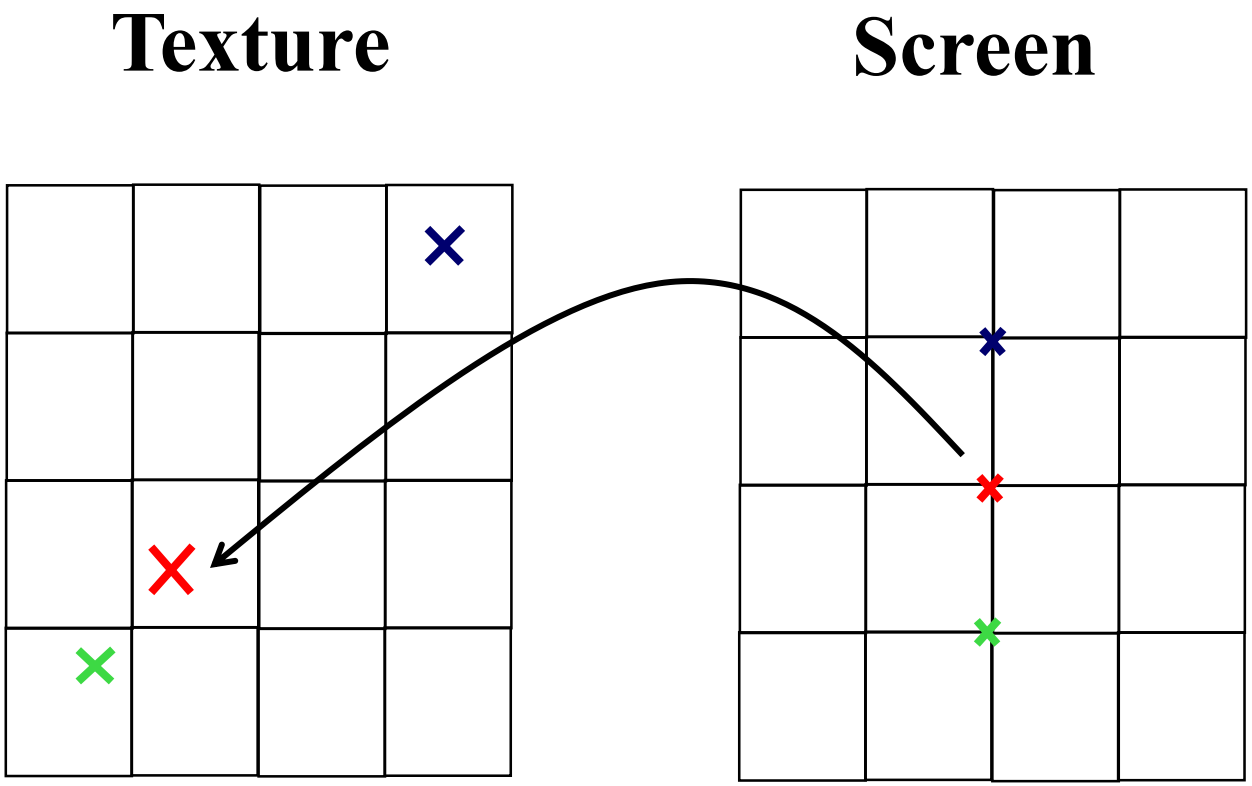


- Linear Interpolation

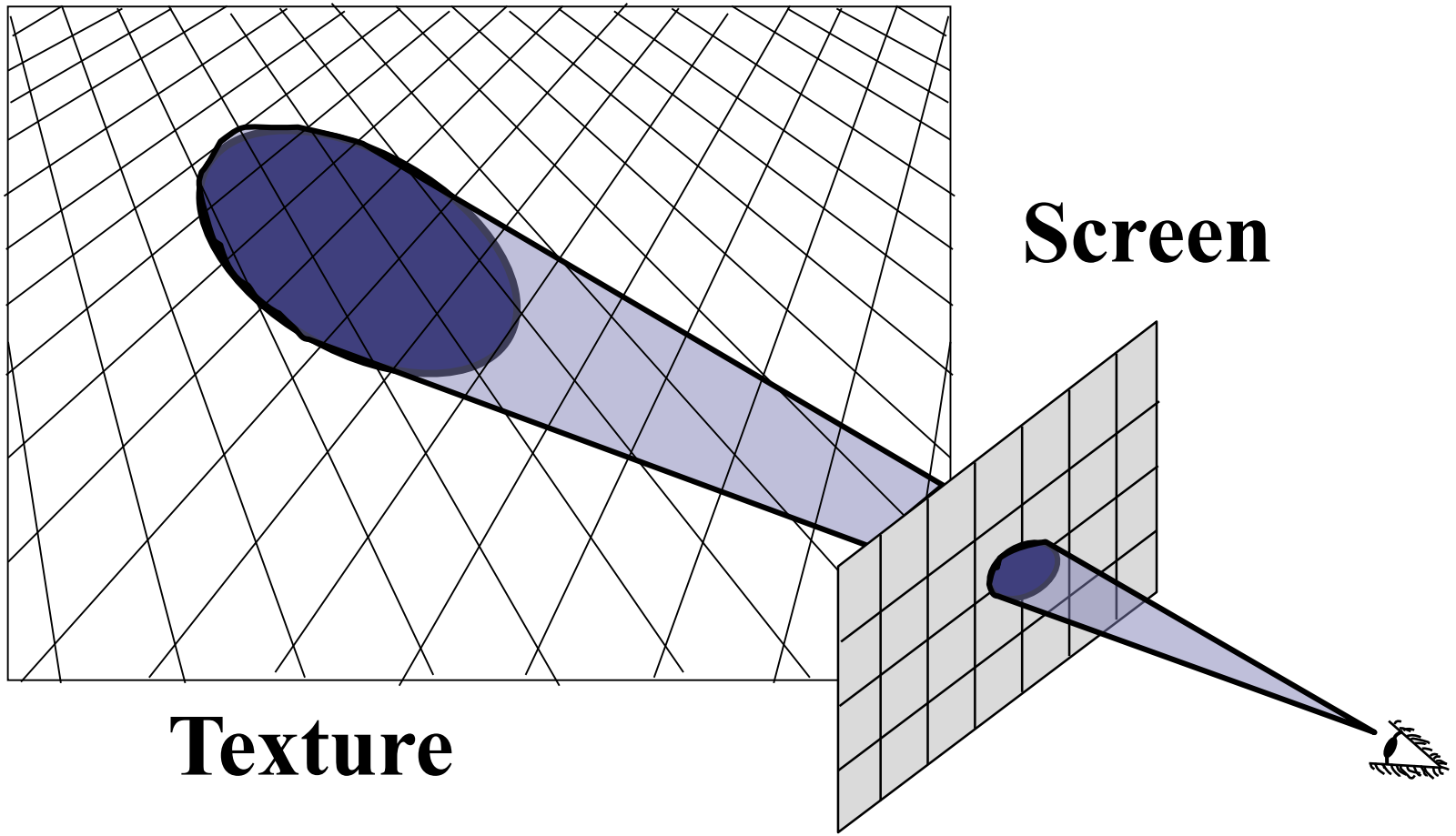
Texture Mapping Artifacts



Texture Resampling



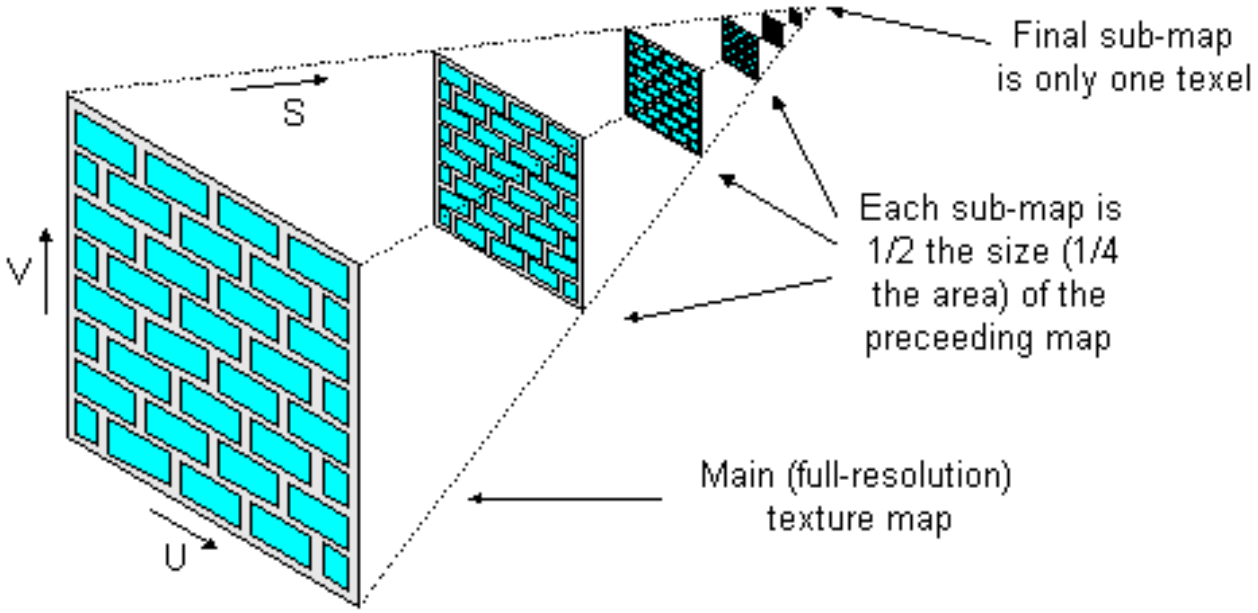
High Quality Texture Mapping



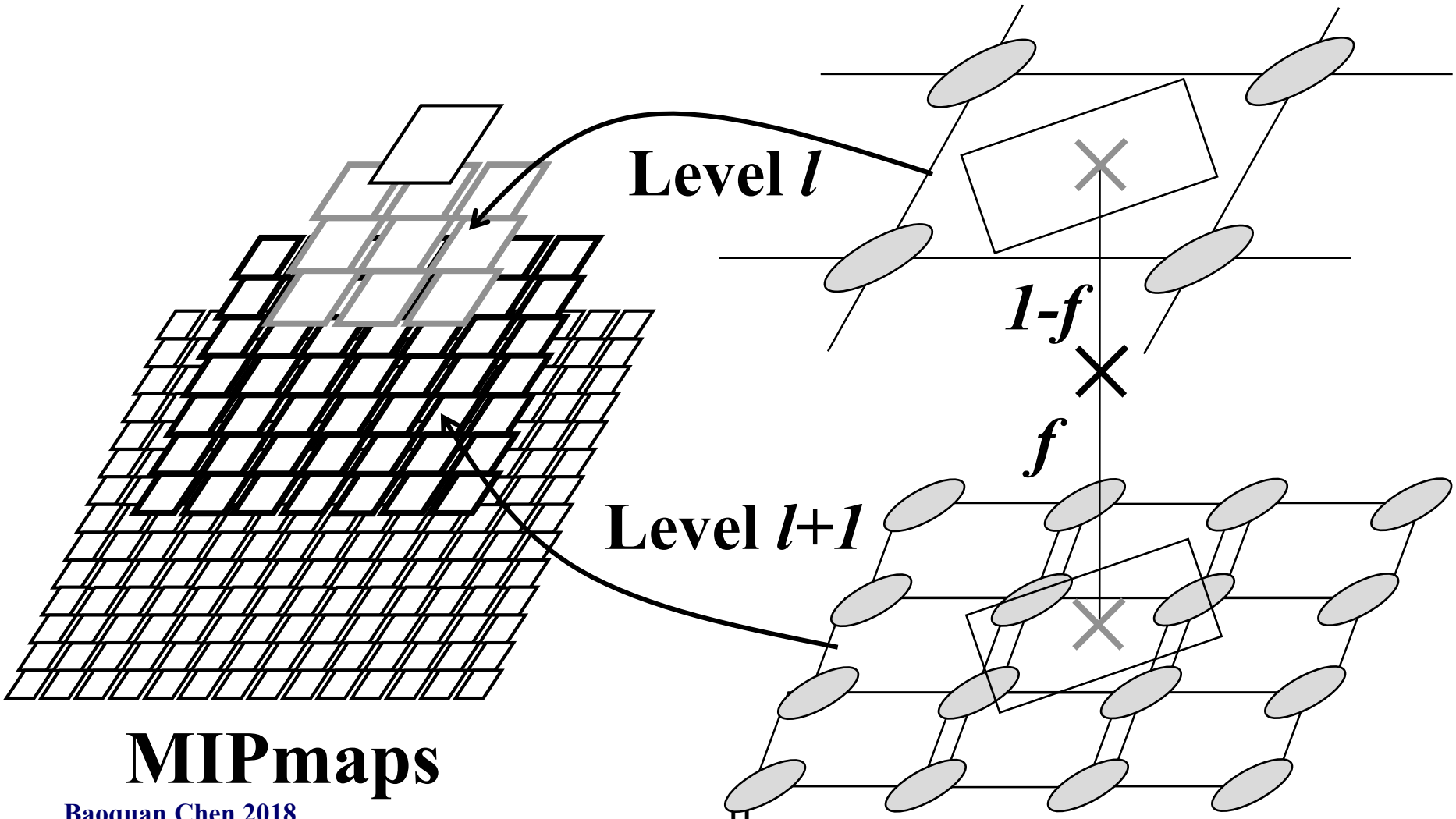
MIP Mapping

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

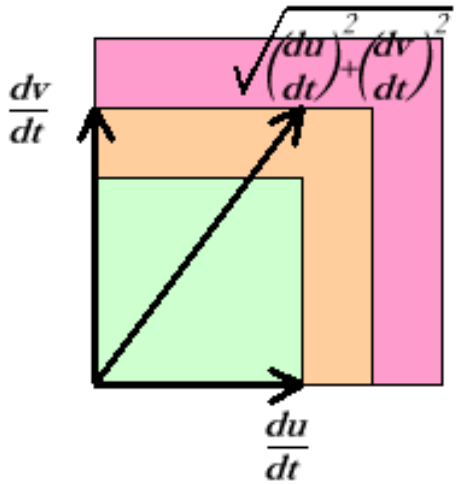
MIP-mapping

MIP Indices

Actually, you have a choice of ways to translate this gradient value into a MIP level. This also brings up one of the shortcomings of MIP mapping. MIP mapping assumes that both the u and v components of the texture index are undergoing a uniform scaling, while in fact the terms du/dt and dv/dt are relatively independent. Thus, we must make some sort of compromise. Two of the most common approaches are given below:

$$level = \log_2\left(\sqrt{\left(\frac{du}{dt}\right)^2 + \left(\frac{dv}{dt}\right)^2}\right)$$

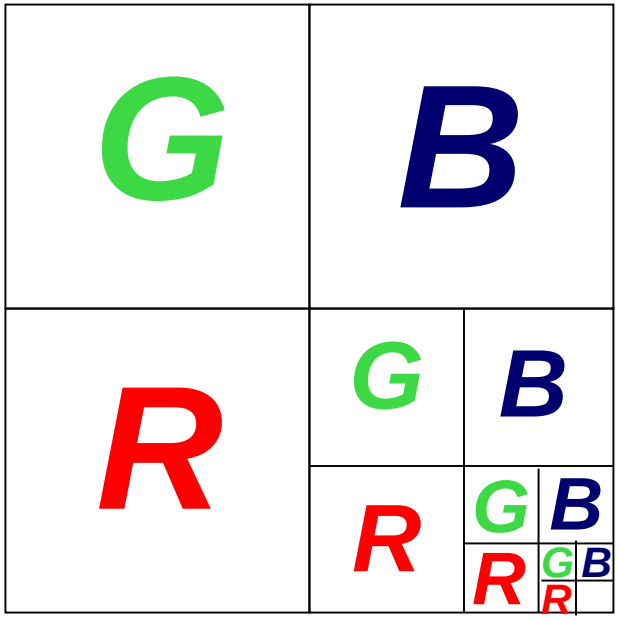
$$level = \log_2\left(Max\left(\left|\frac{du}{dt}\right|, \left|\frac{dv}{dt}\right|\right)\right)$$



The differences between these level selection methods is illustrated by the accompanying figure.

Storing MIP Maps

One convenient method of storing a MIP map is shown below (It also nicely illustrates the 1/3 overhead of maintaining the MIP map).



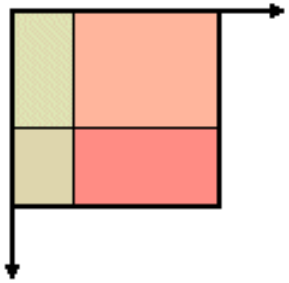
We must make a few small modifications to our rasterizer to compute the MIP map level. Remember the equations that we derived last lecture for mapping screen-space interpolants to their 3-space equivalent.

Summed-Area Tables

There are other approaches to computing this prefiltering integration on the fly. One, which was introduced by Frank Crow is called a summed-area table. Basically, a summed-area table is a tabularized two-dimensional cummulative distribution function. Imagine having a 2-D table of numbers the cummulative distribution function could be found as shown below.

1	6	8	3
0	0	3	7
4	7	8	8
5	0	9	9

1	7	15	18
1	7	18	28
5	18	37	55
10	23	51	78



To find the sum of region contained in a box bounded by (x_0, y_0) *and* (x_1, y_1)*:*
$\mathbf{T}(x_1, y_1) - \mathbf{T}(x_0, y_1) - \mathbf{T}(x_1, y_0) + \mathbf{T}(x_0, y_0)$

This approach can be used to compute the integration of texels that lie under a pixel by dividing the resulting sum by the area of the rectangle,
$(y_1 - y_0)(x_1 - x_0)$.

With a little more work you can compute the area under any four-sided polygon (How?).

Summed-Area Tables

- How much storage does a summed-area table require?
- Does it require more or less work per pixel than a MIP map?
- What sort of low-pass filter does a summed-area table represent?