CS380: Computer Graphics
Texture Mapping

Sung-Eui Yoon
(윤성의)

Course URL:
http://sgvr.kaist.ac.kr/~sungeui/CG

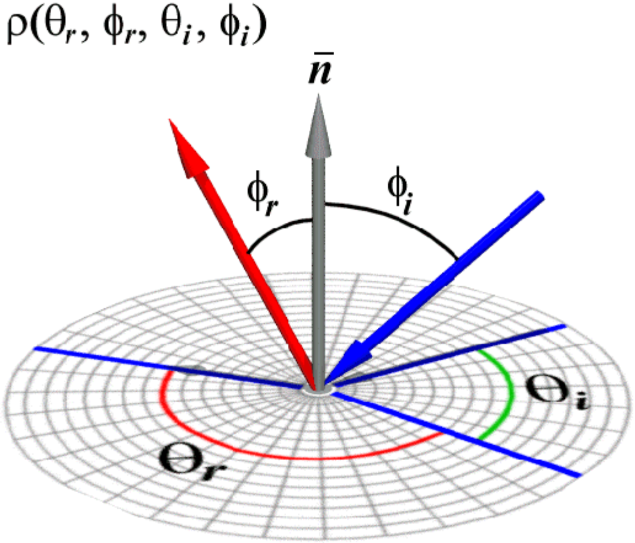
Class Objectives (Ch. 9)

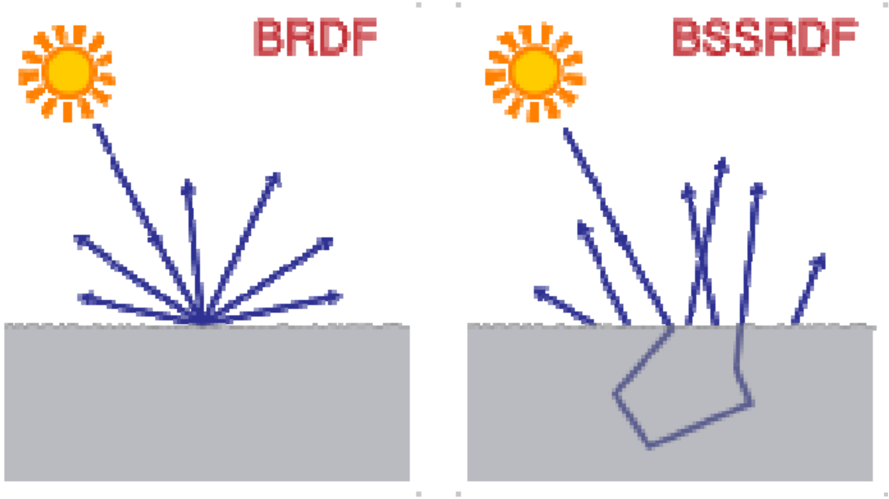
- Texture mapping overview
- Texture filtering

- At the last time
  - Phong illumination: ambient, diffuse, specular, and non-ideal reflectors
  - Shading methods

Questions

- In BRDF, if the material is transparent, can phi_r value could be over 90 degrees?





wiki

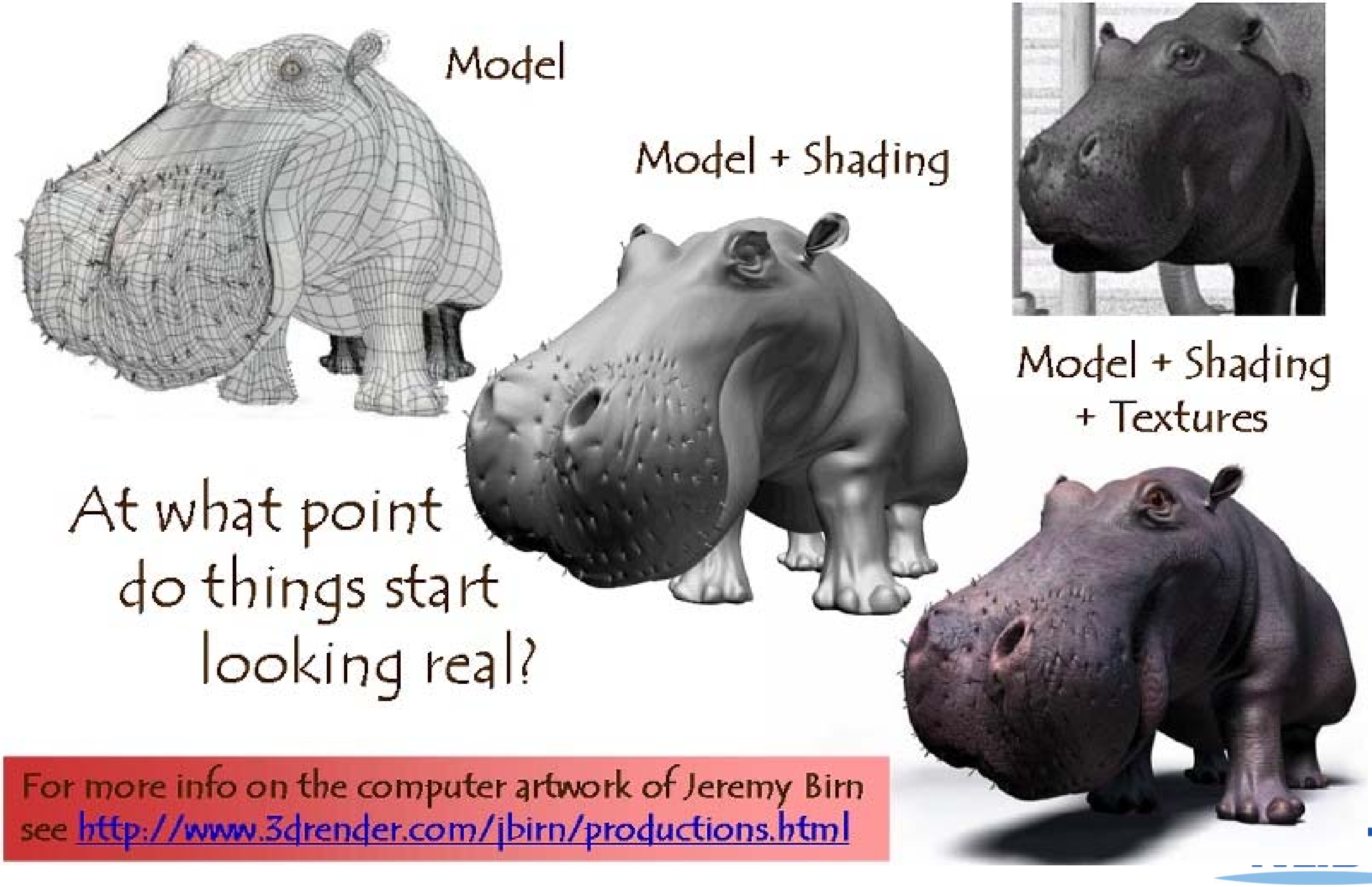
Texture Mapping

- Requires lots of geometry to fully represent complex shapes of models
- Add details with image representations

SHREK
© 2001 PDI/DreamWorks

Excerpted from MIT EECS 6.837,
Durand and Cutler

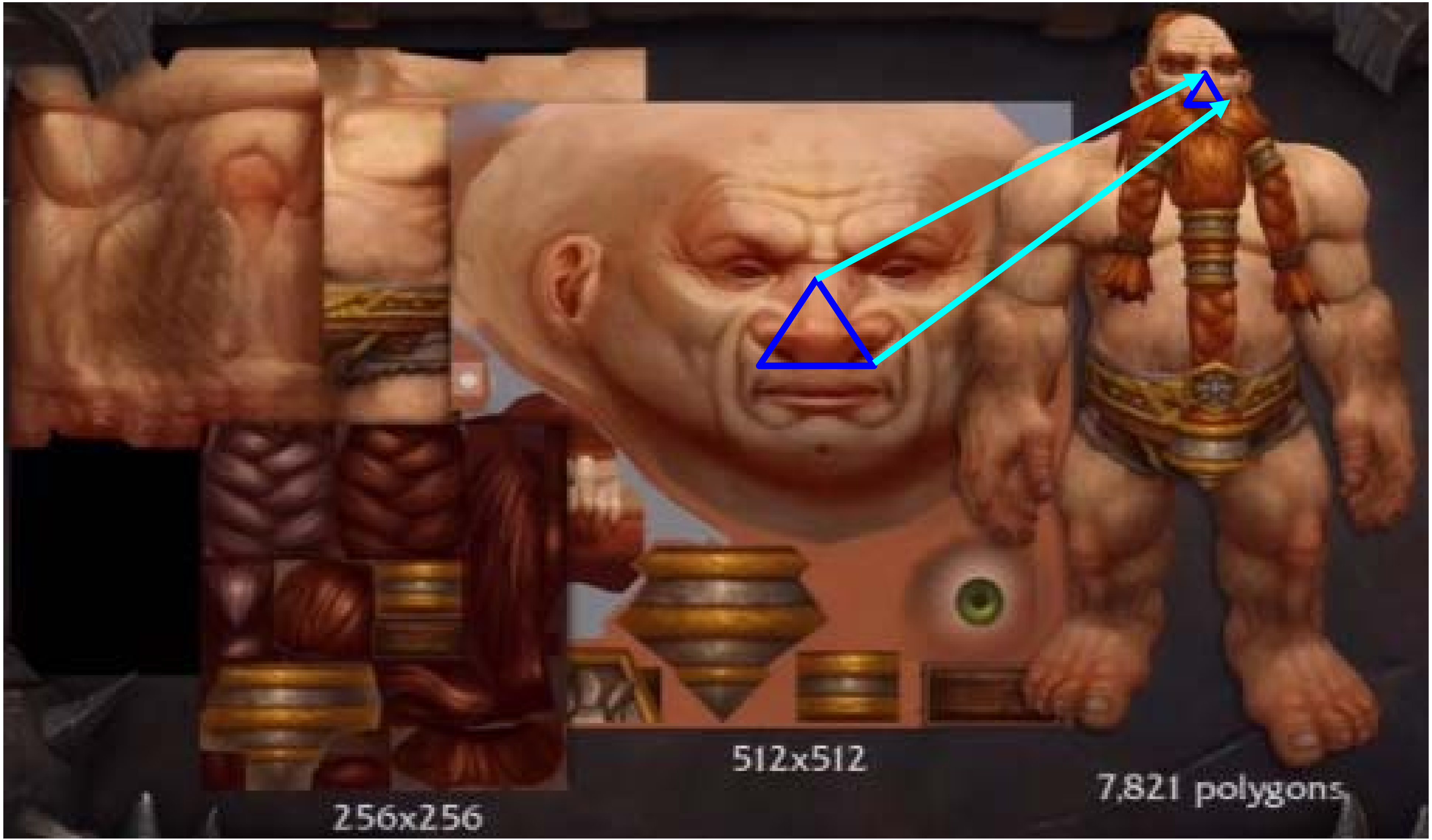
The Quest for Visual Realism



At what point do things start looking real?

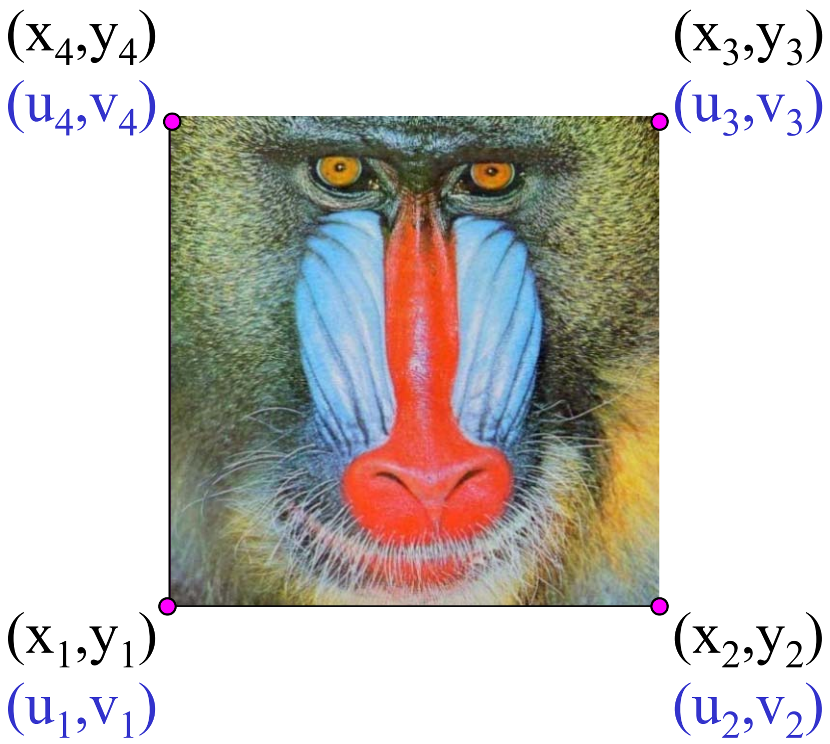
For more info on the computer artwork of Jeremy Birn see http://www.3drender.com/jbirn/productions.html

Texture Mapping



From MMO-champion

Texture Maps in OpenGL



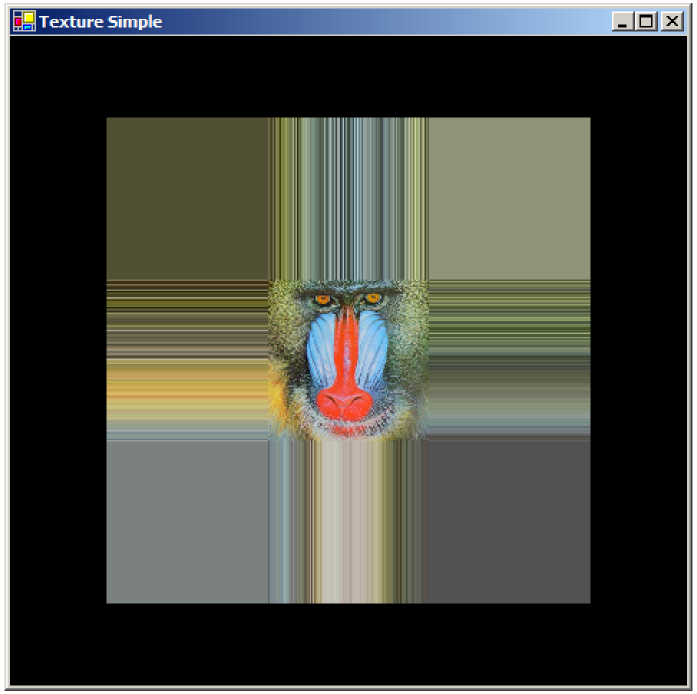
- **Specify normalized texture coordinates at each of the vertices**
- **Texel indices**
  $(s,t) = (u, v) \cdot (width, height)$

```
glBindTexture(GL_TEXTURE_2D, texID)
glBegin(GL_POLYGON)
  glTexCoord2d(0,1); glVertex2d(-1,-1);
  glTexCoord2d(1,1); glVertex2d( 1,-1);
  glTexCoord2d(1,0); glVertex2d( 1, 1);
  glTexCoord2d(0,0); glVertex2d(-1, 1);
glEnd()
```

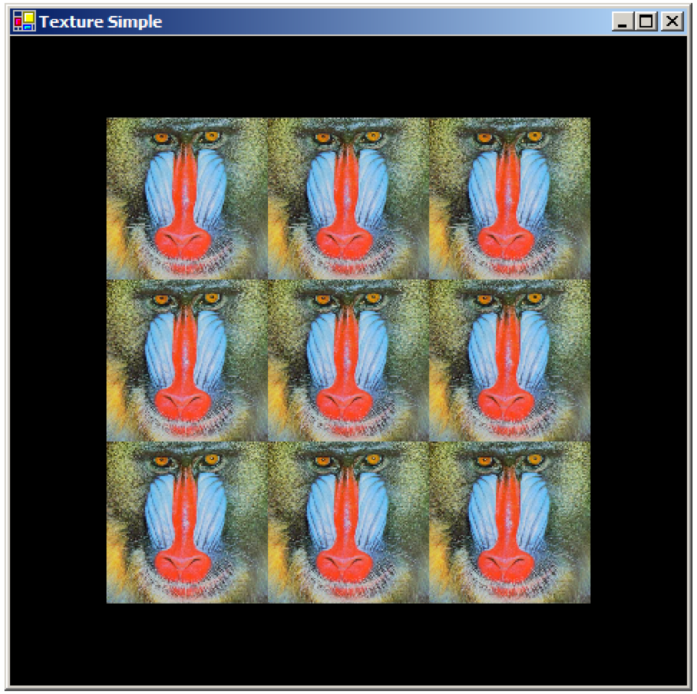
Wrapping

- The behavior of texture coordinates outside of the range [0,1) is determined by the texture wrap options.

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, wrap_mode )
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, wrap_mode )
```



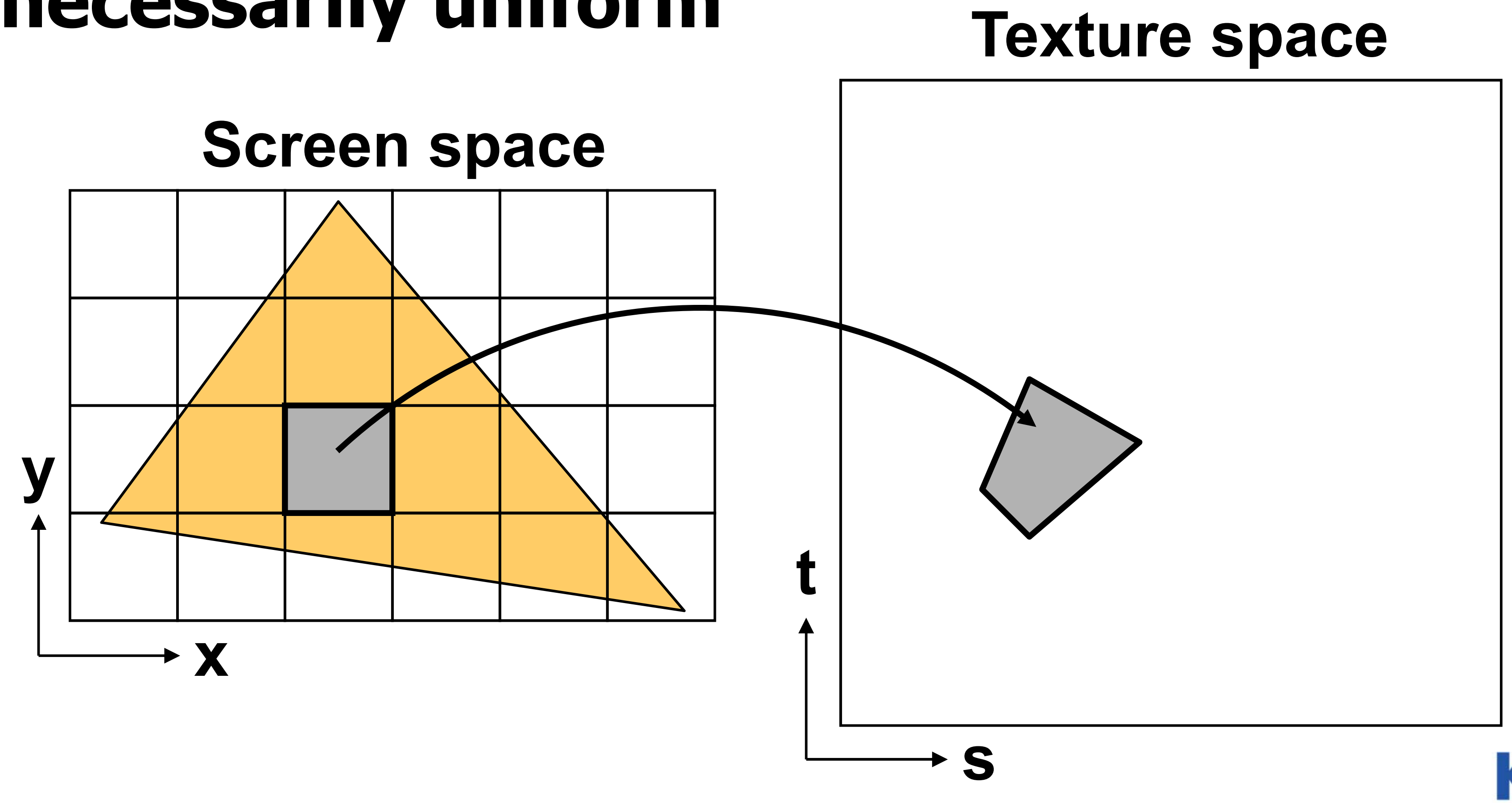
GL_CLAMP



GL_REPEAT

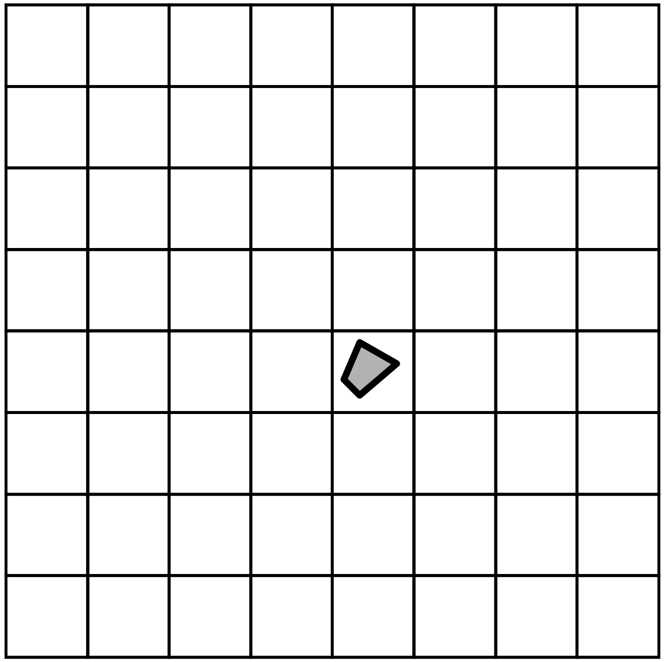
Sampling Texture Maps

- The uniform sampling pattern in screen space cooresponds to some sampling pattern in texture space that is not necessarily uniform

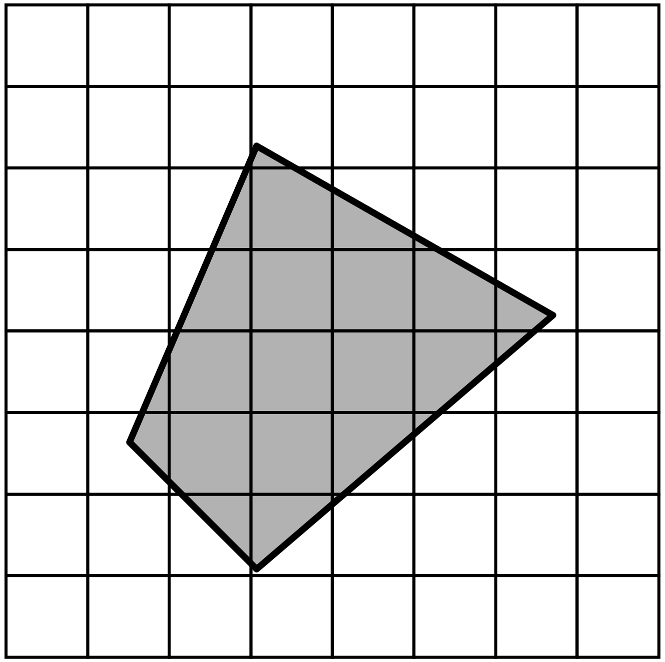


Sampling Density Mismatch

- Sampling density in texture space rarely matches the sample density of the texture itself

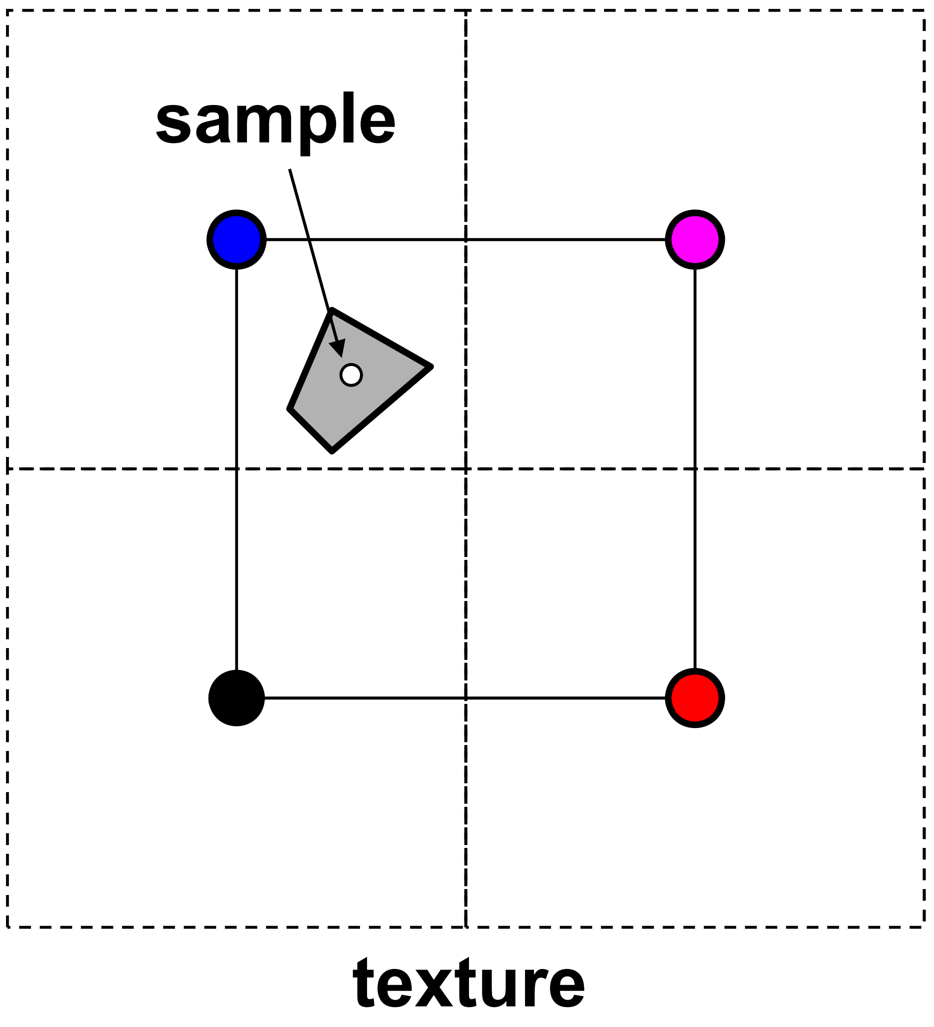


Oversampling
(Magnification)



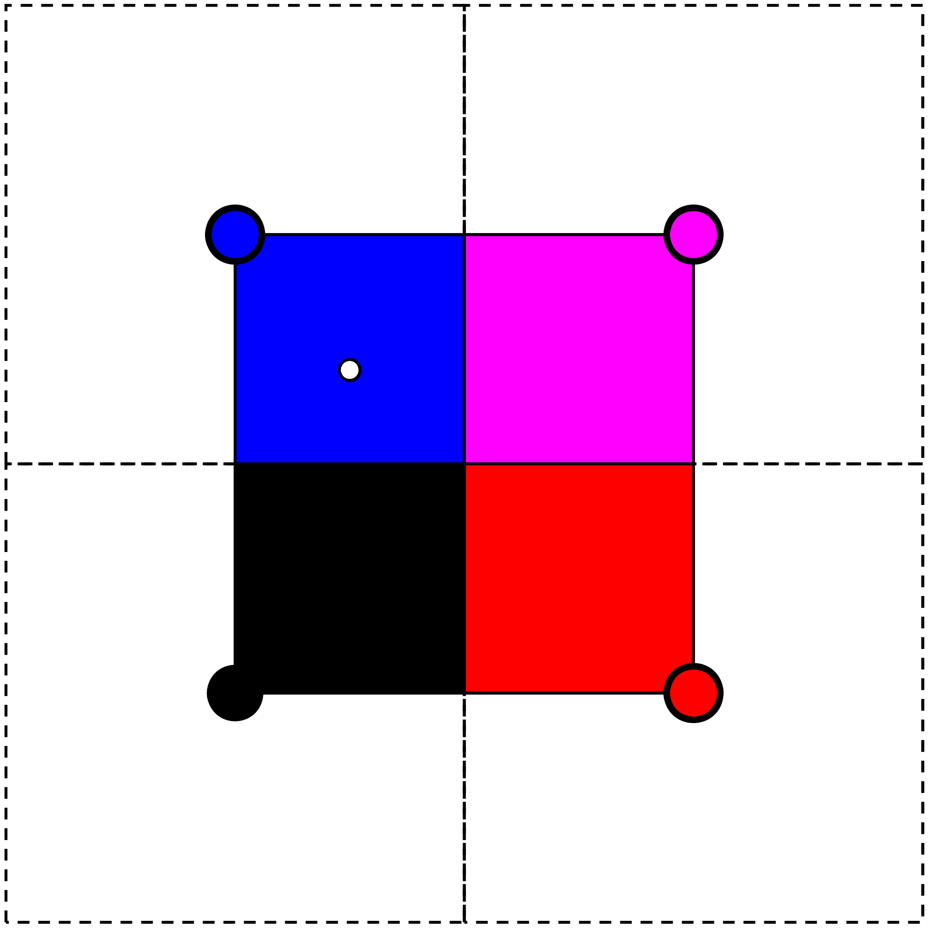
Undersampling
(Minification)

Handling Oversampling



- How do we compute the color to assign to this sample?

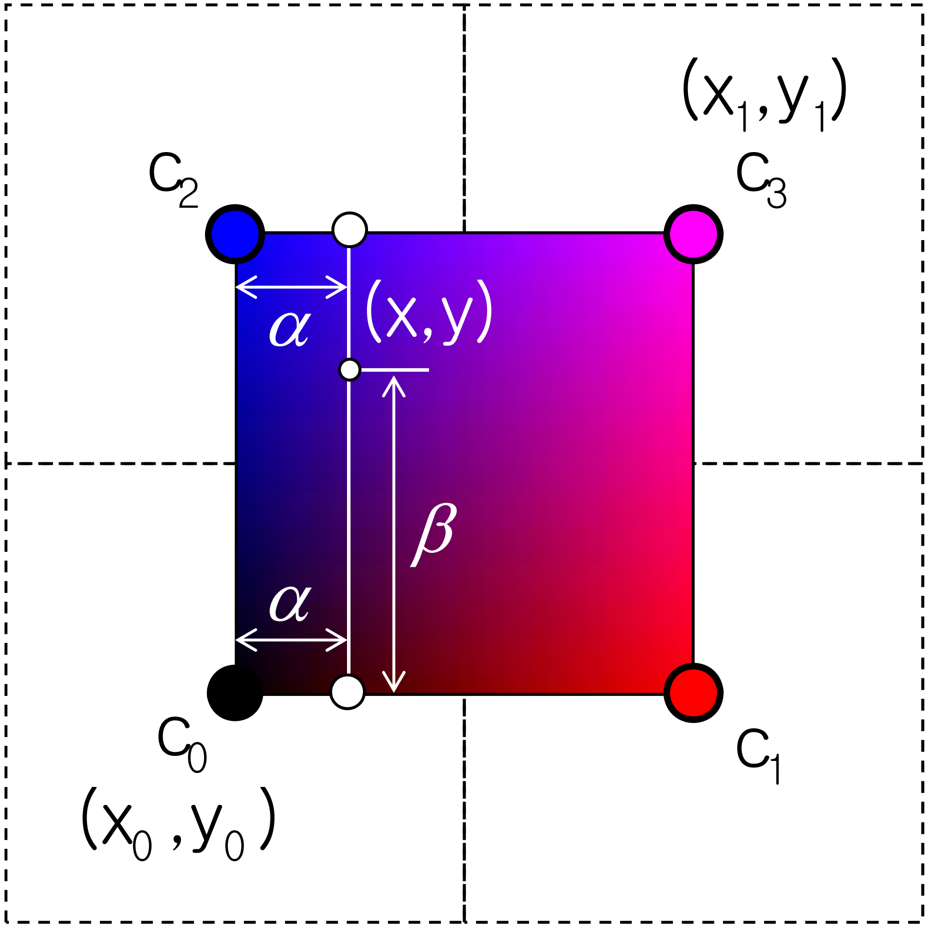
Handling Oversampling



texture

- How do we compute the color to assign to this sample?
- Nearest neighbor – take the color of the closest texel

Handling Oversampling



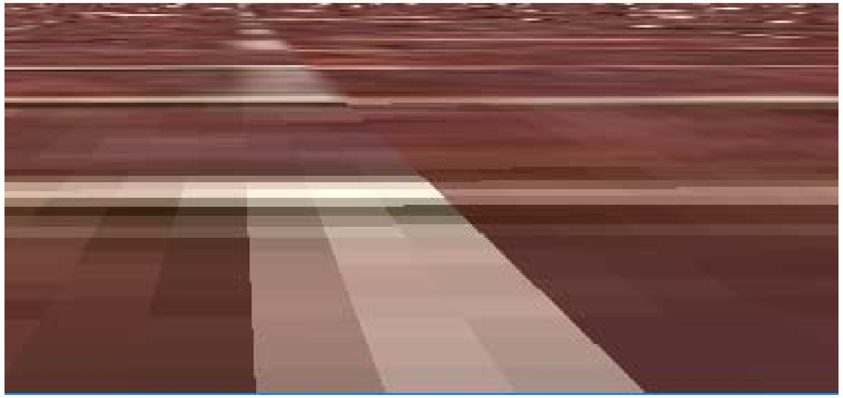
texture

- **How do we compute the color to assign to this sample?**
- **Nearest neighbor – take the color of the closest texel**
- **Bilinear interpolation**

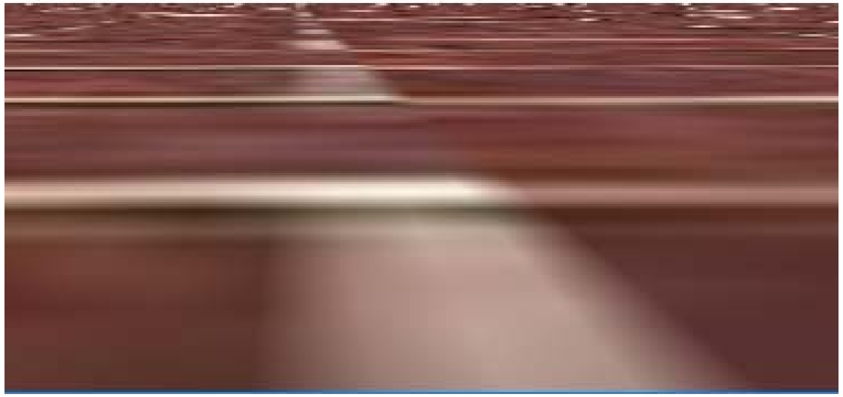
$$\alpha = \frac{x - x_0}{x_1 - x_0} \qquad \beta = \frac{y - y_0}{y_1 - y_0}$$

$$c = ((1 - \alpha)c_0 + \alpha c_1)(1 - \beta) + ((1 - \alpha)c_2 + \alpha c_3)\beta$$

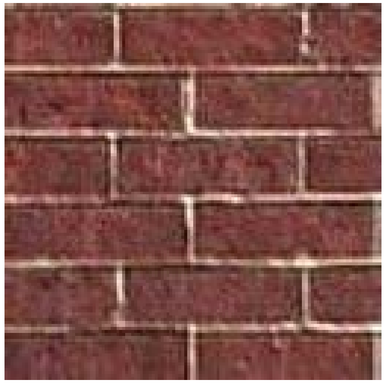
Visual Comparison



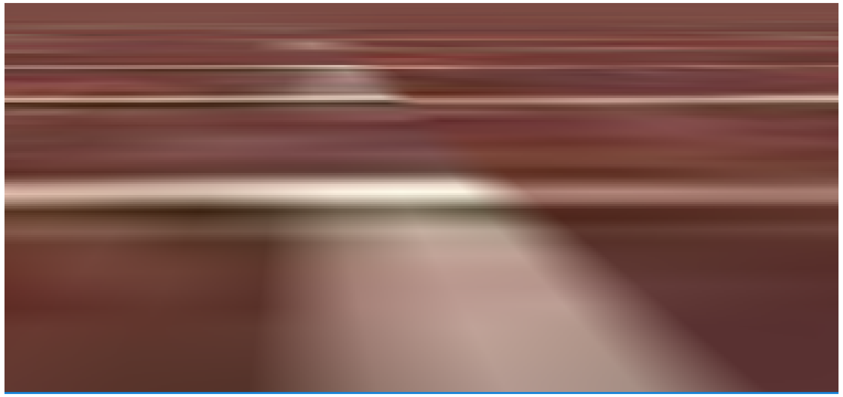
Mag. filter: nearest
Min. filter: linear



Mag. filter: linear
Min. filter: linear

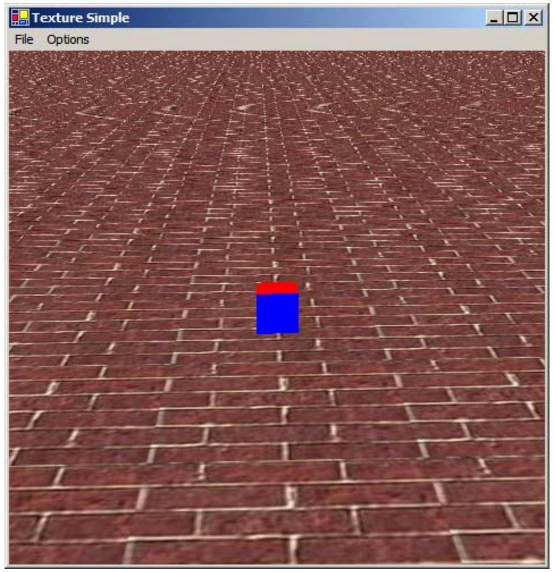


Original texture



Mag. filter: linear
Min. filter: mipmap

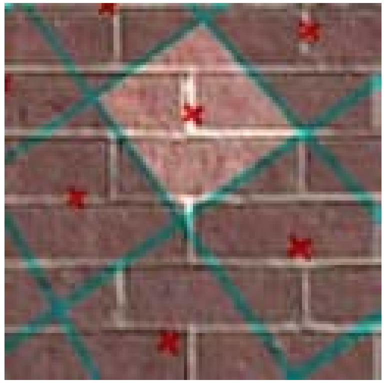
Undersampling



- Details in the texture tend to pop (disappear and reappear)
  - Mortar (white substances) in the brick
- High-frequency details lead to strange patterns
  - Aliasing

Spatial Filtering

- To avoid aliasing we need to prefilter the texture to remove high frequencies
  - Prefiltering is essentially a spatial integration over the texture
  - Integrating on the fly is expensive: perform integration in a pre-process

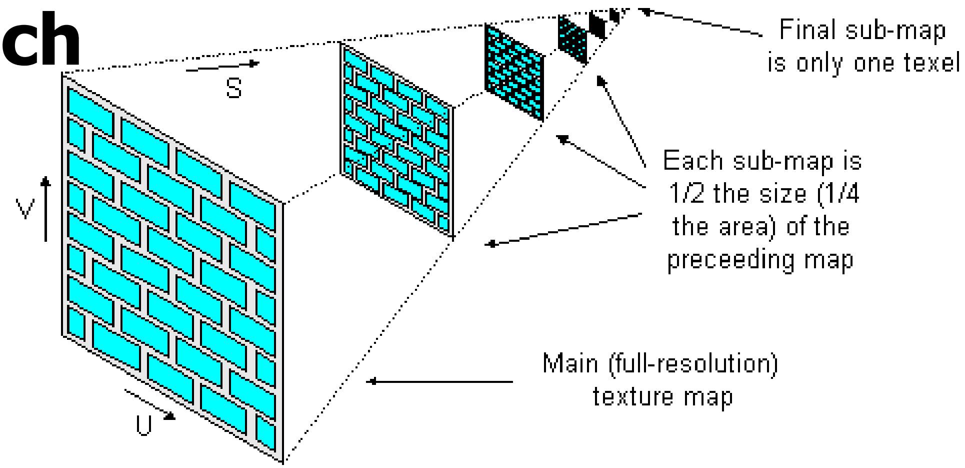


Samples and their extents

Proper filtering removes aliasing

MIP Mapping

- MIP is an acronym for the Latin phrase *multium in parvo*, which means "many in one place"
  - Constructs an *image pyramid*
  - Each level is a prefiltered version of the level below resampled at half the frequency



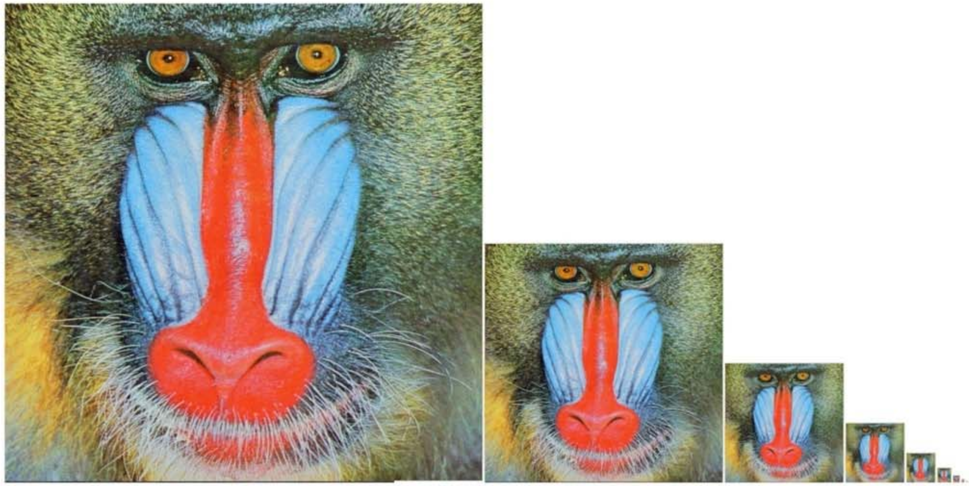
S
V
U
Final sub-map is only one texel
Each sub-map is 1/2 the size (1/4 the area) of the preceeding map
Main (full-resolution) texture map

- While rasterizing use the level with the sampling rate closest to the desired sampling rate
  - Can also interpolate between pyramid levels
- How much storage overhead is required?

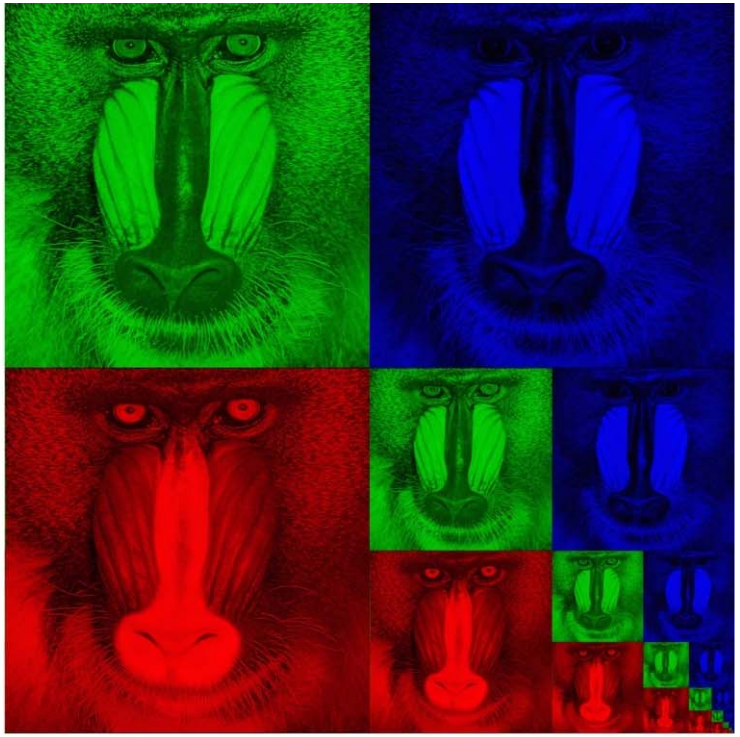
$$\text{mip map size} = \sum_{i=0}^{\infty}\left(\frac{1}{4}\right)^{i} = \frac{1}{1-\frac{1}{4}} = \frac{4}{3}$$

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**



10-level mip map



Memory format of a mip map

Finding the MIP Level

- Use the projection of a pixel in screen into texture space to figure out which level to use

Texture Filtering in OpenGL

- **Automatic creation**

```
gluBuild2DMipmaps(GL_TEXTURE_2D, GL_RGBA, width, height,
                  GL_RGBA, GL_UNSIGNED_BYTE, data)
```

- **Filtering**

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, filter )
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, filter )
```

where filter is:

```
GL_NEAREST
GL_LINEAR

GL_LINEAR_MIPMAP_LINEAR
GL_NEAREST_MIPMAP_NEAREST
GL_NEAREST_MIPMAP_LINEAR
GL_LINEAR_MIPMAP_NEAREST
```

inter-level **intra-level**

Class Objectives were:

- Texture mapping overview
- Texture filtering

Next Time

- **Various applications of texture mapping**
- **Visibility and ray tracing**

Homework

- **Go over the next lecture slides before the class**
- **Watch 2 SIGGRAPH videos and submit your summaries before every Mon. class**
- **Submit questions two times during the whole semester**