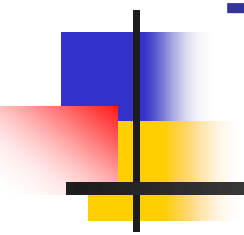
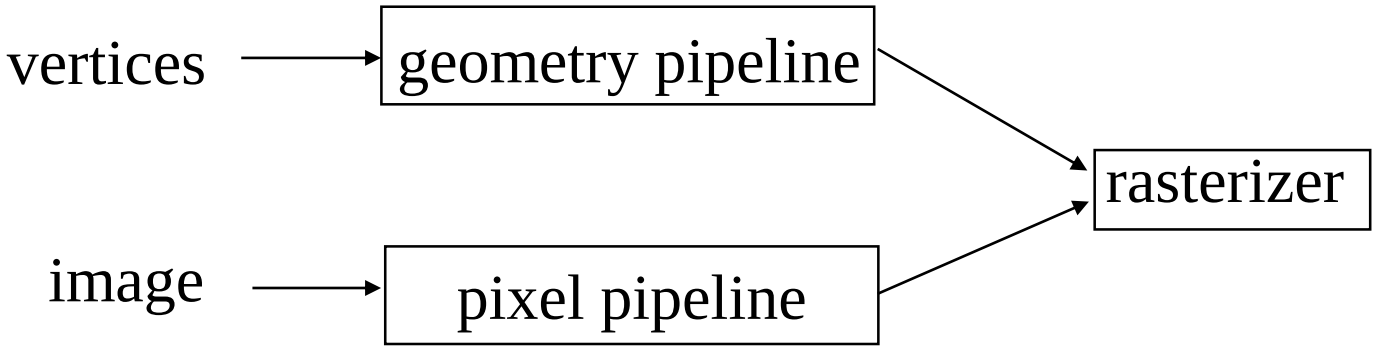


Texture Mapping in OpenGL

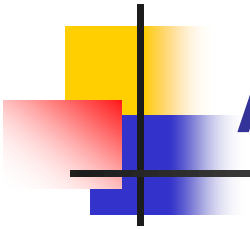
Texture Mapping and the OpenGL Pipeline

- Images and geometry flow through separate pipelines that join at the rasterizer
  - "complex" textures do not affect geometric complexity



Texture Example

- The texture (below) is a 256 x 256 image that has bee mapped to a rectangular polygon which is viewed in perspective



Applying Textures I

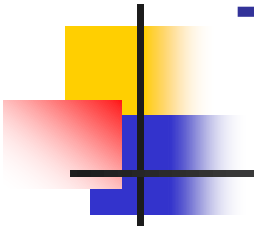
- Three steps
  - ① specify texture
    - read or generate image
    - assign to texture
  - ② assign texture coordinates to vertices
  - ③ specify texture parameters
    - wrapping, filtering

Applying Textures II

1. specify textures in texture objects
2. set texture filter
3. set texture function
4. set texture wrap mode
5. set optional perspective correction hint
6. bind texture object
7. enable texturing
8. supply texture coordinates for vertex

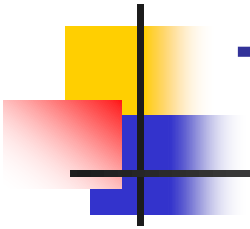
    - coordinates can also be generated

Texture Objects



- Like display lists for texture images
  - one image per texture object
  - may be shared by several graphics contexts
- Generate texture names

```
glGenTextures( n, *texIds );
```



Texture Objects (cont.)

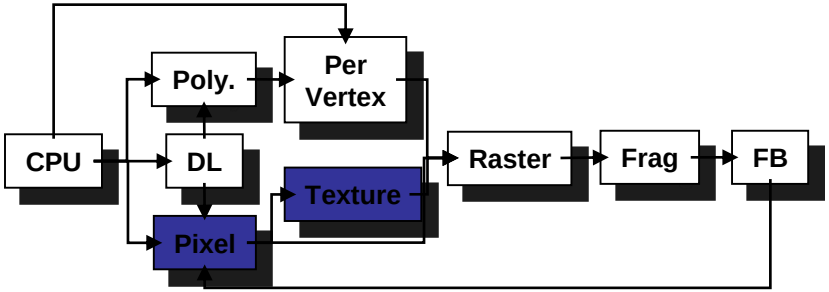
- Create texture objects with texture data and state

```
glBindTexture( target, id );
```

- Bind textures before using

```
glBindTexture( target, id );
```

Specify Texture Image



- Define a texture image from an array of texels in CPU memory

```
glTexImage2D( target, level, components,
      w, h, border, format, type, *texels );
```

  - dimensions of image must be powers of 2
- Texel colors are processed by pixel pipeline
  - pixel scales, biases and lookups can be done

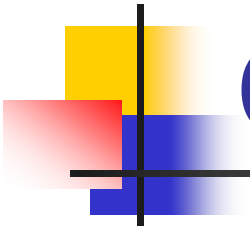
Converting A Texture Image

- If dimensions of image are not power of 2

```
gluScaleImage( format, w_in, h_in,
    type_in, *data_in, w_out, h_out,
        type_out, *data_out );
```

  - *_in ***is for source image***
  - *_out ***is for destination image***
- Image interpolated and filtered during scaling

Specifying a Texture: Other Methods



- Use frame buffer as source of texture image
  - uses current buffer as source image

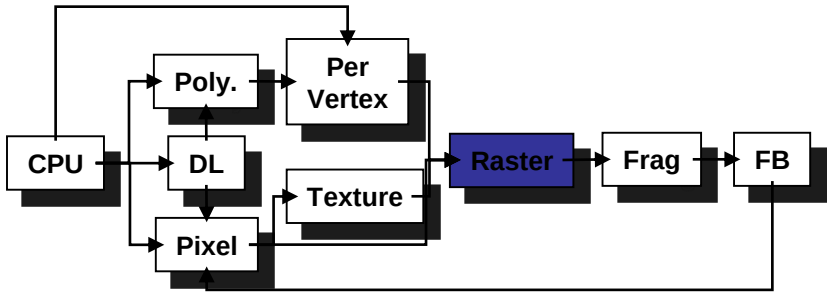
    `glCopyTexImage2D(...)`

    `glCopyTexImage1D(...)`
- Modify part of a defined texture

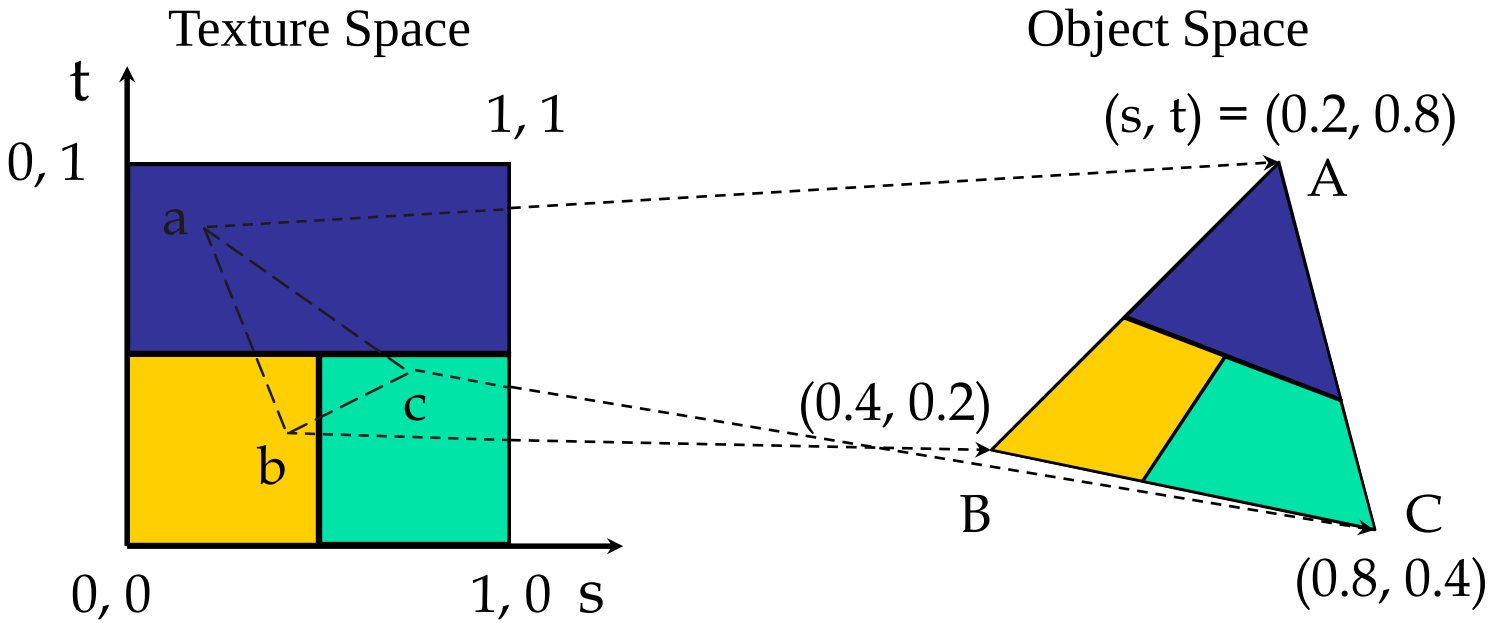
    `glTexSubImage2D(...)`

    `glTexSubImage1D(...)`
- Do both with *`glCopyTexSubImage2D(...)`*, etc.

Mapping a Texture



- Based on parametric texture coordinates
- `glTexCoord*()` specified at each vertex



Generating Texture Coordinates

- Automatically generate texture coords

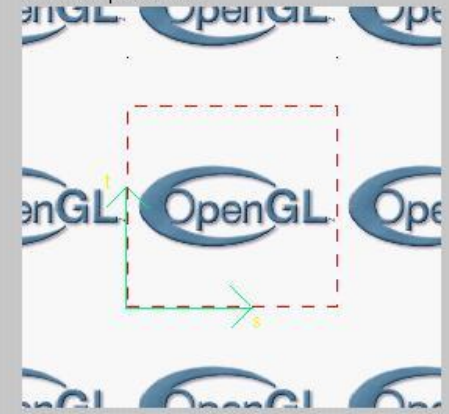
  `glTexGen{ifd}[v]()`

- specify a plane
  - generate texture coordinates based upon distance from plane $Ax + By + Cz + D = 0$
- generation modes
  - `GL_OBJECT_LINEAR`
  - `GL_EYE_LINEAR`
  - `GL_SPHERE_MAP`

Tutorial: Texture

Screen-space view

Texture-space view

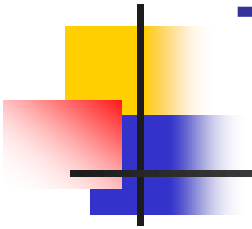


Command manipulation window

```
GLfloat border_color[ ] = {  1.00 ,  0.00 ,  0.00 ,  1.00 };
GLfloat env_color[ ] = {     0.00 ,  1.00 ,  0.00 ,  1.00 };

glTexParameterfv(GL_TEXTURE_2D, GL_TEXTURE_BORDER_COLOR, border_color);
glTexEnvfv(GL_TEXTURE_ENV, GL_TEXTURE_ENV_COLOR, env_color);

glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,  GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,  GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S,  GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T,  GL_REPEAT);
glTexEnvi(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE,  GL_MODULATE);

glEnable(GL_TEXTURE_2D);
gluBuild2DMipmaps(GL_TEXTURE_2D, 3, w, h, GL_RGB, GL_UNSIGNED_BYTE, image);
glColor4f(   0.60  , 0.60  , 0.60  , 1.00  );
glBegin(GL_POLYGON);
glTexCoord2f( 0.0  , 0.0  ); glVertex3f( -1.0 , -1.0 , 0.0  );
glTexCoord2f( 1.0  , 0.0  ); glVertex3f( 1.0  , -1.0 , 0.0  );
glTexCoord2f( 1.0  , 1.0  ); glVertex3f( 1.0  , 1.0  , 0.0  );
glTexCoord2f( 0.0  , 1.0  ); glVertex3f( -1.0 , 1.0  , 0.0  );
glEnd();
```

Click on the arguments and move the mouse to modify values.

Texture Application Methods

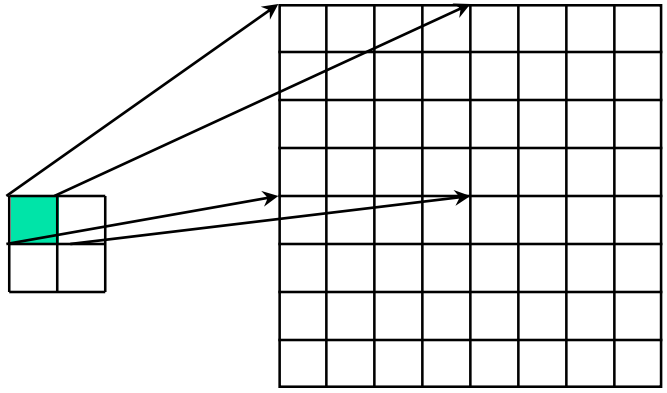


- Filter Modes
  - minification or magnification
  - special mipmap minification filters
- Wrap Modes
  - clamping or repeating
- Texture Functions
  - how to mix primitive's color with texture's color
    - blend, modulate or replace texels

Filter Modes

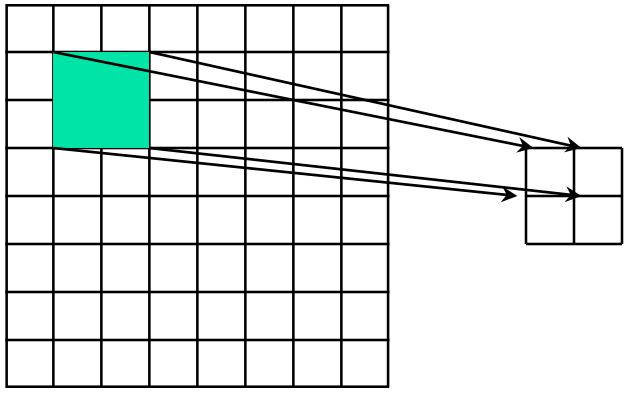
Example:

```
glTexParameteri( target, type, mode );
```



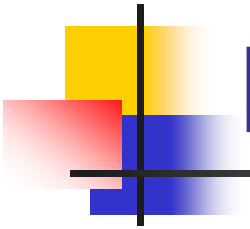
Texture Polygon

Magnification



Texture Polygon

Minification



Filter Modes

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXURE_MAG_FILTER,
                GL_NEAREST);
```

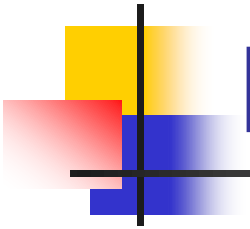
- GL_NEAREST
  - Returns the value of the texture element that is nearest to the center of the pixel being textured. Use this parameter if you would like your texture to appear sharp when rendered.
- GL_LINEAR
  - Returns the weighted average of the four texture elements that are closest to the center of the pixel being textured. These can include border texture elements, depending on the values of GL_TEXTURE_WRAP_S and GL_TEXTURE_WRAP_T, and on the exact mapping. Use this parameter if you would like your texture to appear blurred when rendered.

Image Texturing

- Magnification of 

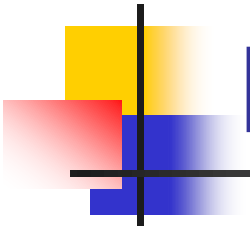
Nearest neighbor

Bilinear interpolation



Mipmapping

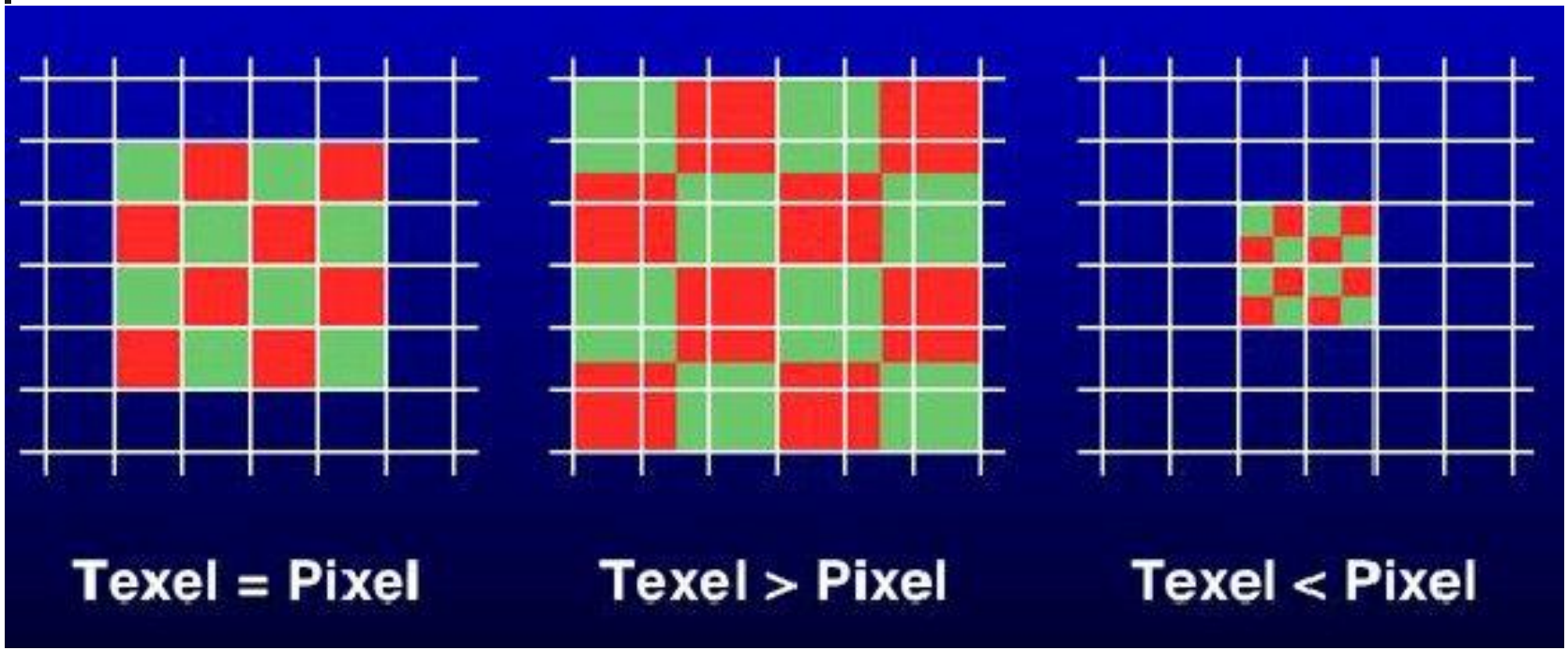
- MIP - *multium in parvo* - many in a small place.
- The basic idea is to construct a pyramid of images that are prefiltered and resampled at sampling frequencies that are a binary fractions (1/2, 1/4, 1/8, etc) of the original image's sampling.
- While rasterizing we compute the index of the decimated image that is sampled at a rate closest to the density of our desired sampling rate



Mipmapping

- If a textured object is far away, one screen pixel (on an object) may map to many texture pixels
  - No need to use the original resolution of the texel array
- A mipmap is a low resolution version of a texture
  - Texture is filtered down as a pre-processing step:
    - `gluBuild2DMipmaps(…)`
  - Those mipmaps are automatically used if we specify
    - `glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST_MIPMAP_NERAEST)`
  - When the textured object is far away, use the mipmap chosen so that one image pixel maps to at most four mipmap pixels
  - Full set of mipmaps requires double the storage of the original texture

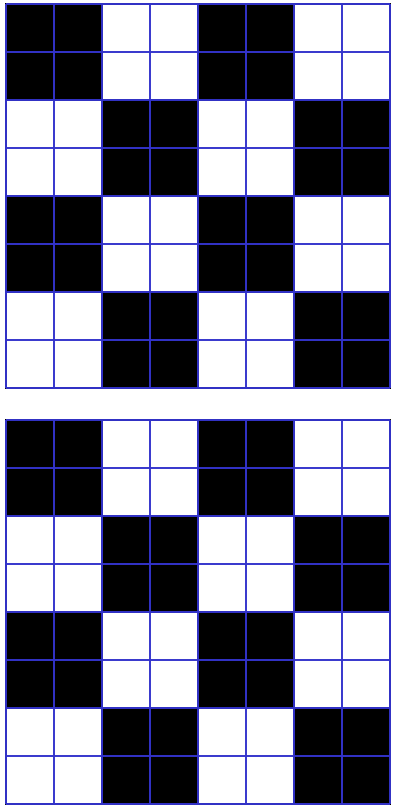
Mipmapping



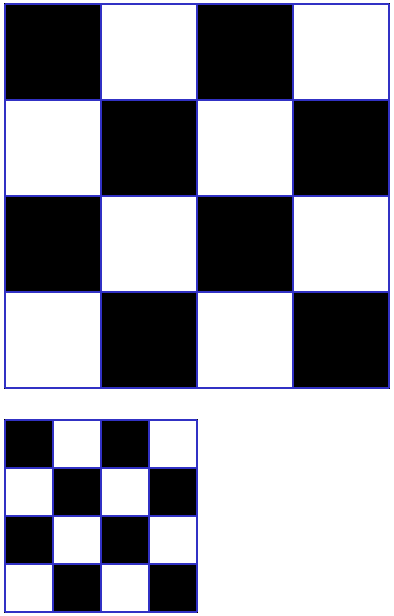
Mipmap allows for prefiltered texture maps of decreasing resolution. (a special case of minification filtering)

Mipmaps

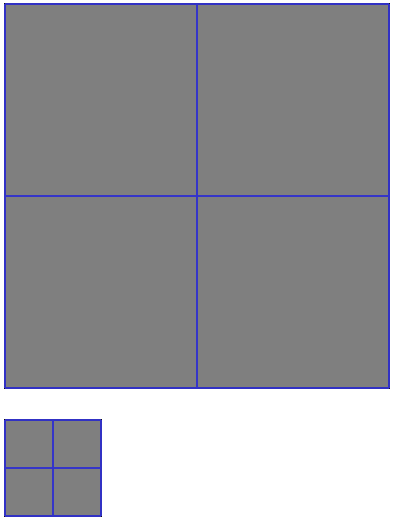
Use on near objects



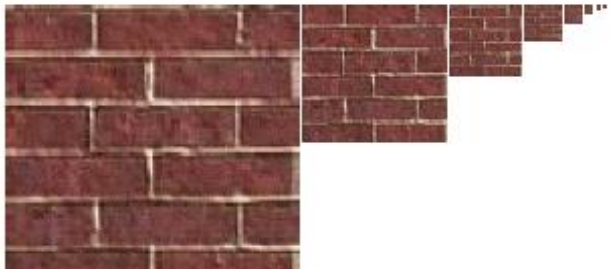
Use on middle objects

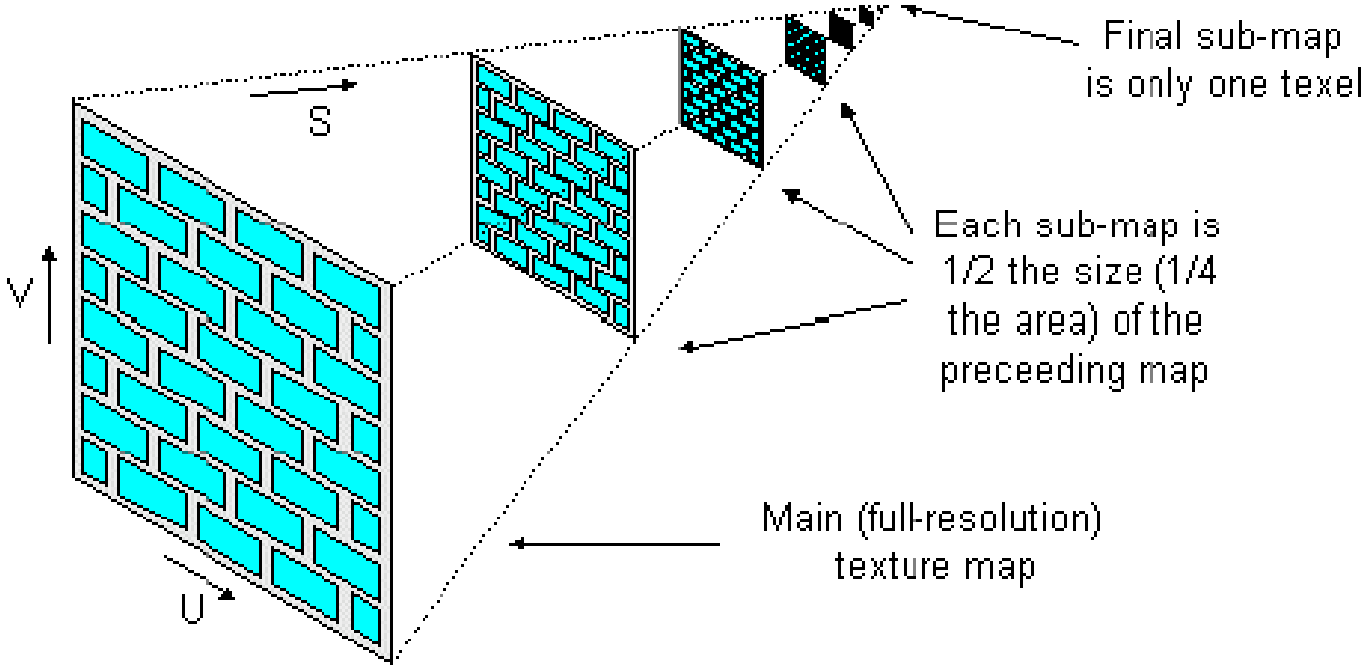


Use on far objects

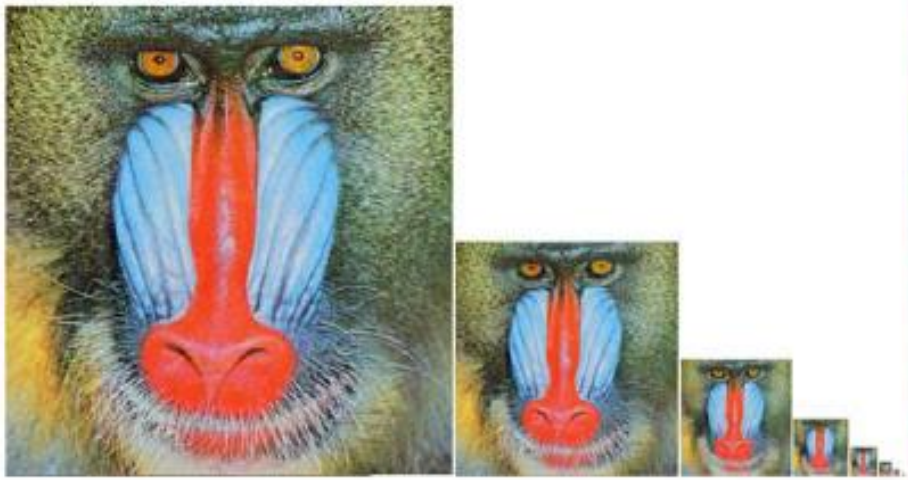


Mipmapping

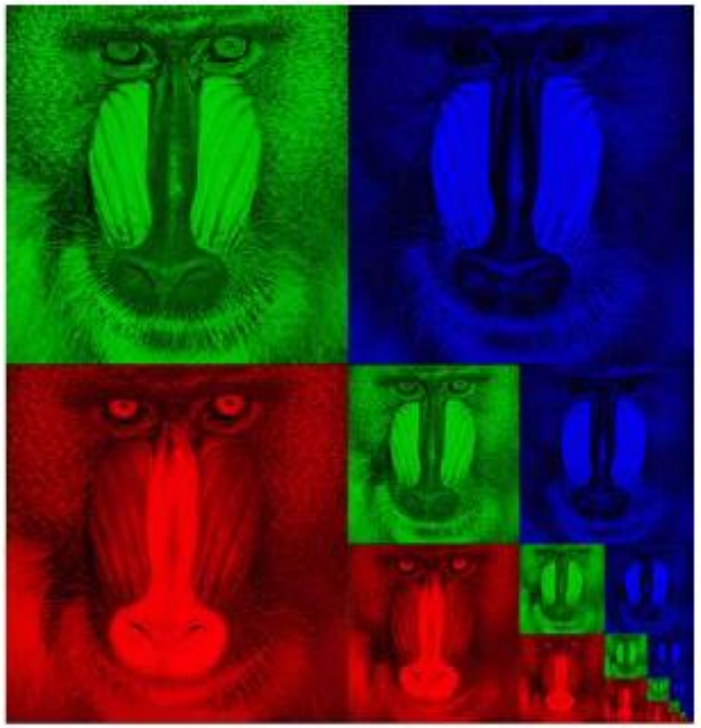



S
V
U
Final sub-map is only one texel
Each sub-map is 1/2 the size (1/4 the area) of the preceeding map
Main (full-resolution) texture map

Storing Mip Maps



10-level mip map



Memory format of a mip map

Anisotropic Mipmapping

What happens when the surface is tilted?

Different mipmap for the two directions

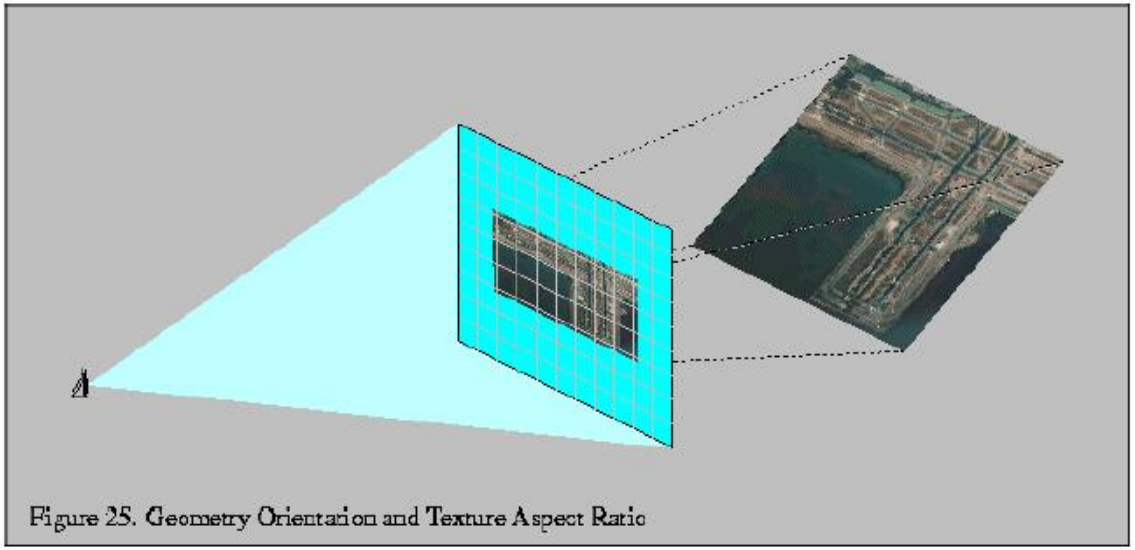


Figure 25. Geometry Orientation and Texture Aspect Ratio

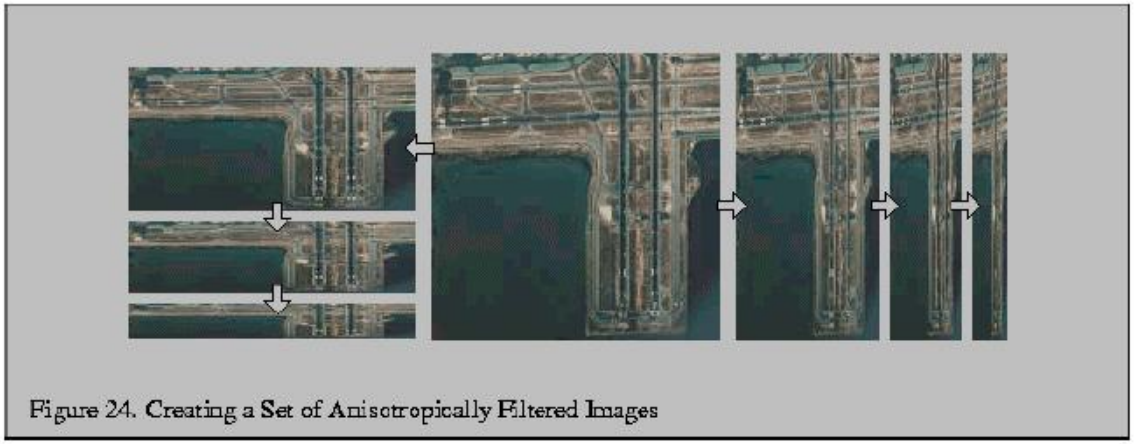


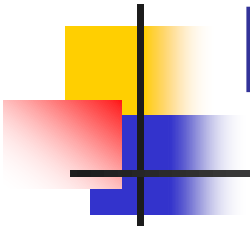
Figure 24. Creating a Set of Anisotropically Filtered Images

MIP-map Example

- No filtering:

- MIP-map texturing:

Mipmapped Textures



- Mipmap allows for prefiltered texture maps of decreasing resolutions
- Lessens interpolation errors for smaller textured objects
- Declare mipmap level during texture definition

```
glTexImage*D( GL_TEXTURE_*D, level,
  … )
```

- GLU mipmap builder routines

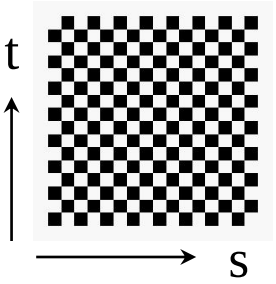
```
gluBuild*DMipmaps( … )
```

- OpenGL 1.2 introduces advanced LOD controls

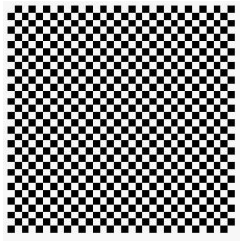
Wrapping Mode

- Example:

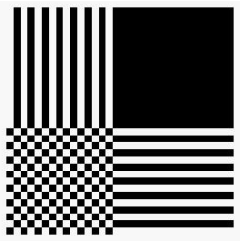
```
glTexParameteri( GL_TEXTURE_2D,
     GL_TEXTURE_WRAP_S, GL_CLAMP )
glTexParameteri( GL_TEXTURE_2D,
     GL_TEXTURE_WRAP_T, GL_REPEAT )
```



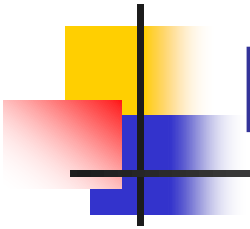
texture



GL_REPEAT wrapping



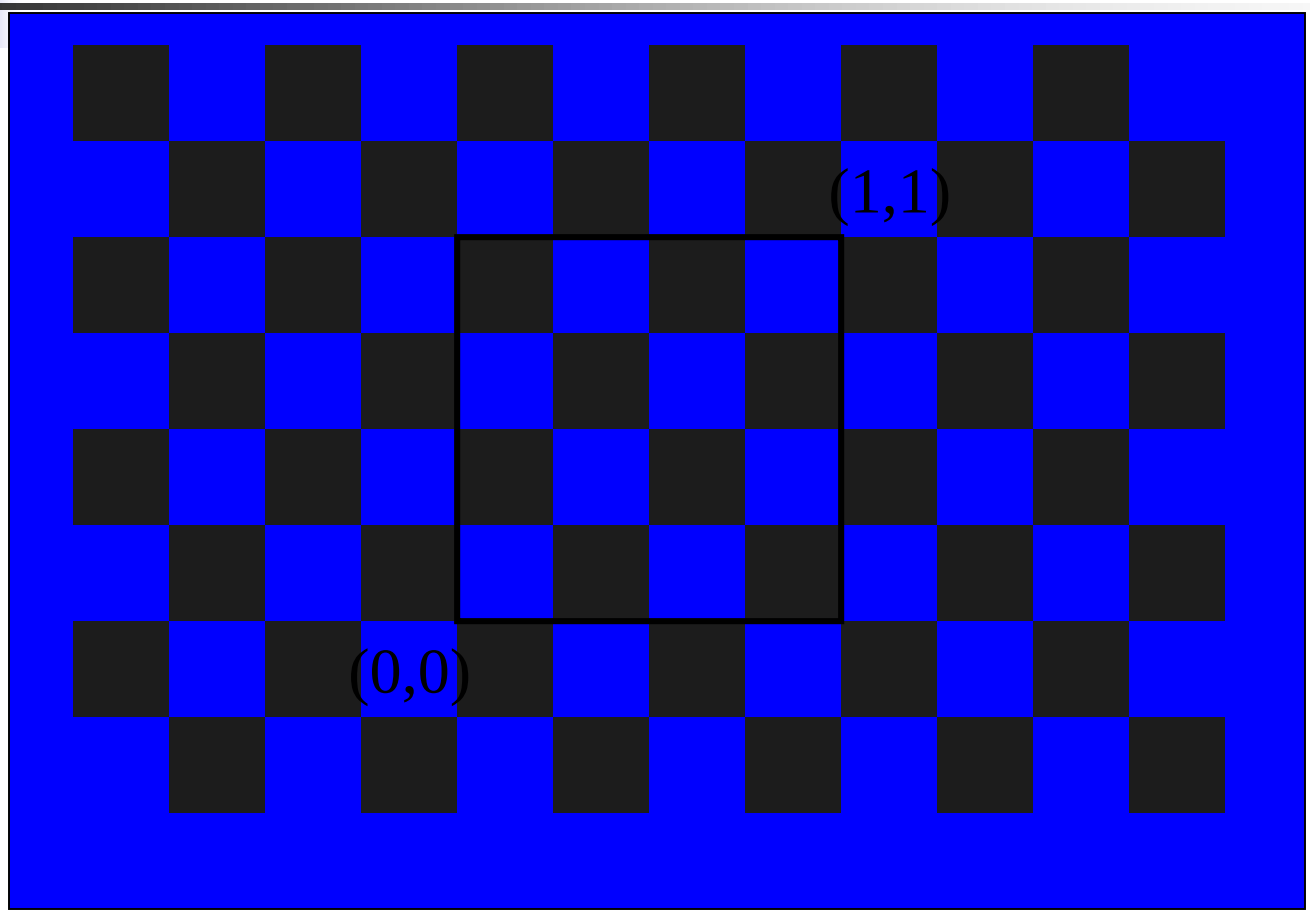
GL_CLAMP wrapping



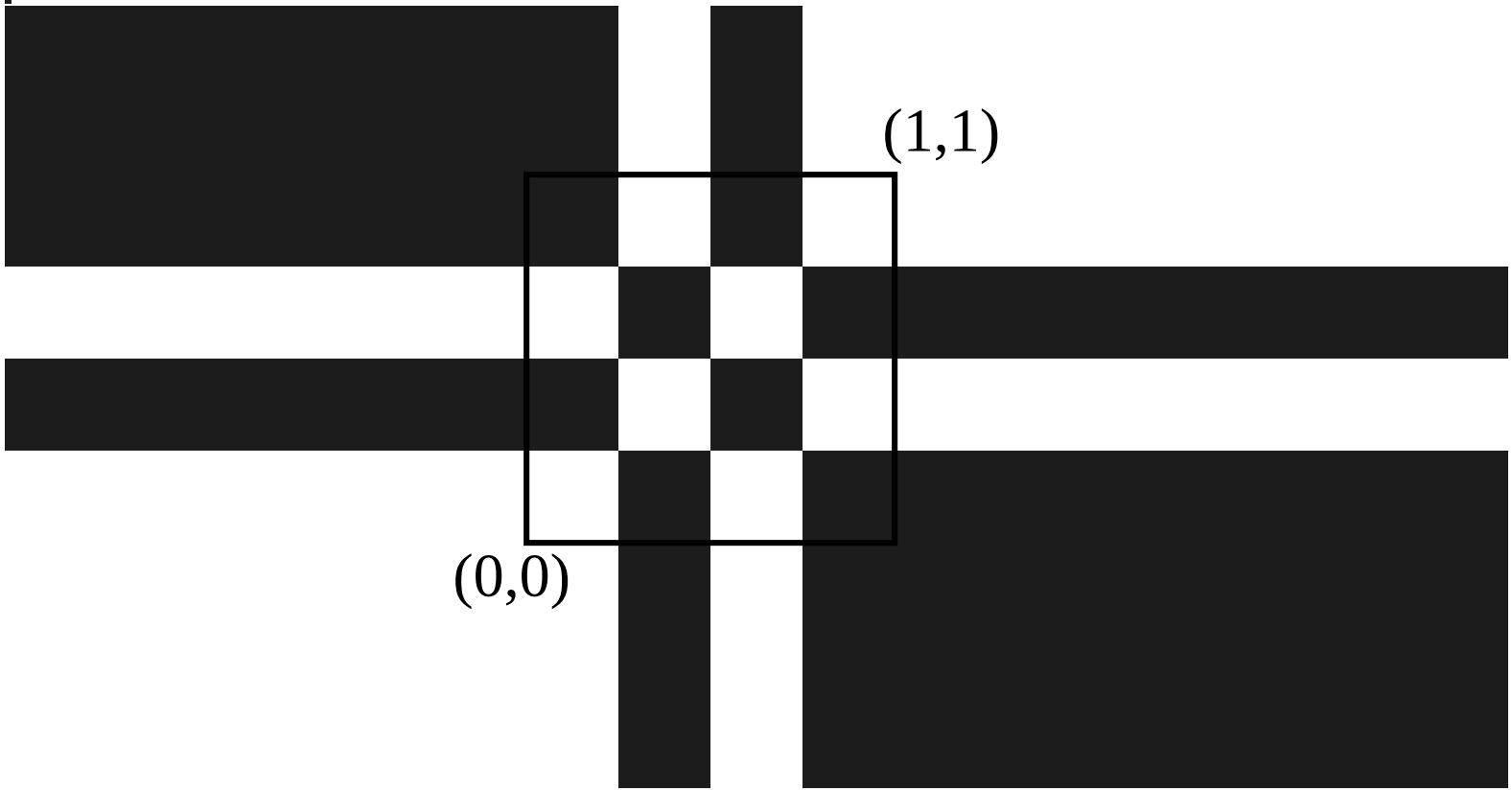
Boundaries

- You can control what happens if a point maps to a texture coordinate outside of the texture image
  - All textures are assumed to go from (0,0) to (1,1) in texture space
- Repeat: Assume the texture is tiled
  - glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT)
- Clamp: Clamp to Edge: the texture coordinates are truncated to valid values, and then used
  - glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP)
- Can specify a special border color:
  - glTexParameterfv(GL_TEXTURE_2D, GL_TEXTURE_BORDER_COLOR, R,G,B,A)

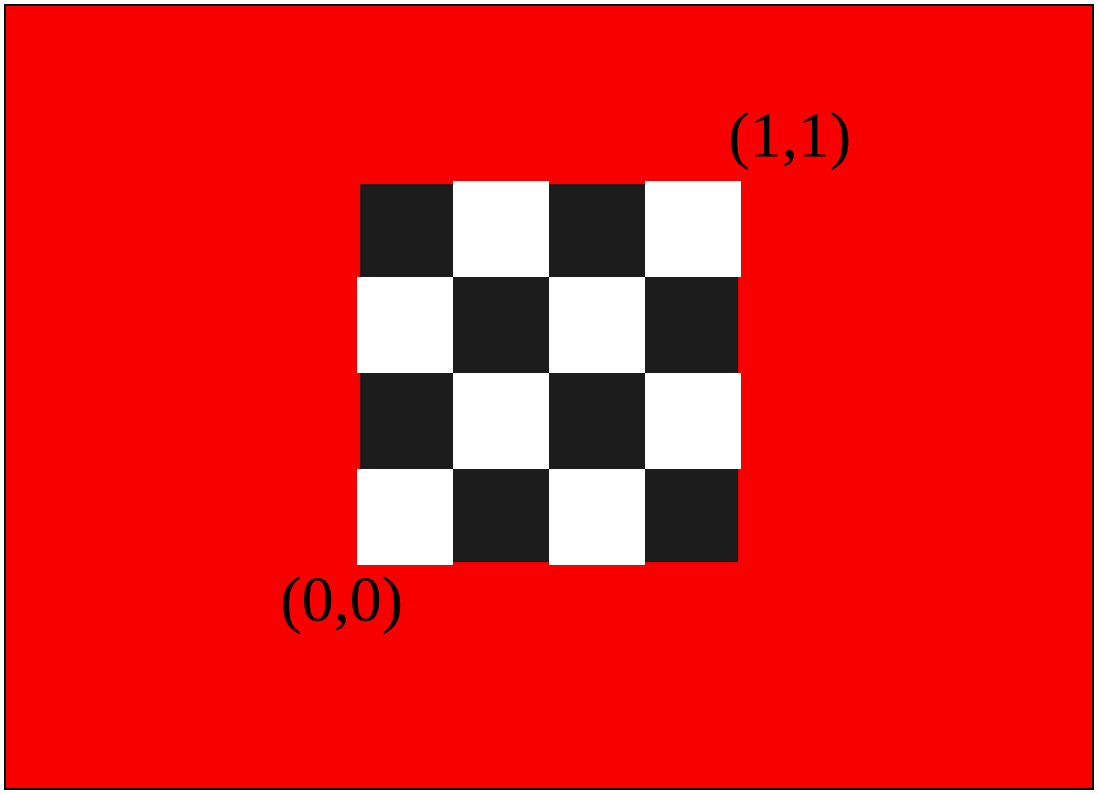
Repeat Border



Clamp Border



Border Color



Texture Functions

- Controls how texture is applied

  `glTexEnv{fi}[v](` *`GL_TEXTURE_ENV, prop, param`* `)`

- *`GL_TEXTURE_ENV_MODE`* modes
  - **`GL_MODULATE`**
  - **`GL_BLEND`**
  - **`GL_REPLACE`**
- Set blend color with *`GL_TEXTURE_ENV_COLOR`*

Perspective Correction Hint

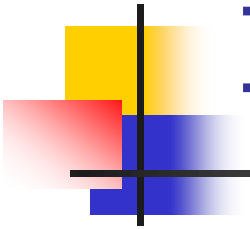
- Texture coordinate and color interpolation
  - either linearly in screen space
  - or using depth/perspective values (slower)
- Noticeable for polygons “on edge”

```
glHint( GL_PERSPECTIVE_CORRECTION_HINT, hint
                                        )
```

where ***hint*** is one of

- ***GL_DONT_CARE***
- ***GL_NICEST***
- ***GL_FASTEST***

Is There Room for a Texture?

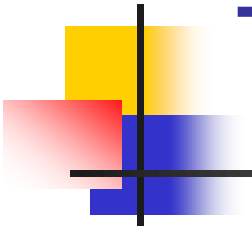


- Query largest dimension of texture image
  - typically largest square texture
  - doesn't consider internal format size

```
glGetIntegerv( GL_MAX_TEXTURE_SIZE, &size
                                    )
```

- Texture proxy
  - will memory accommodate requested texture size?
  - no image specified; placeholder
  - if texture won't fit, texture state variables set to 0
    - doesn't know about other textures
    - only considers whether this one texture will fit all of memory

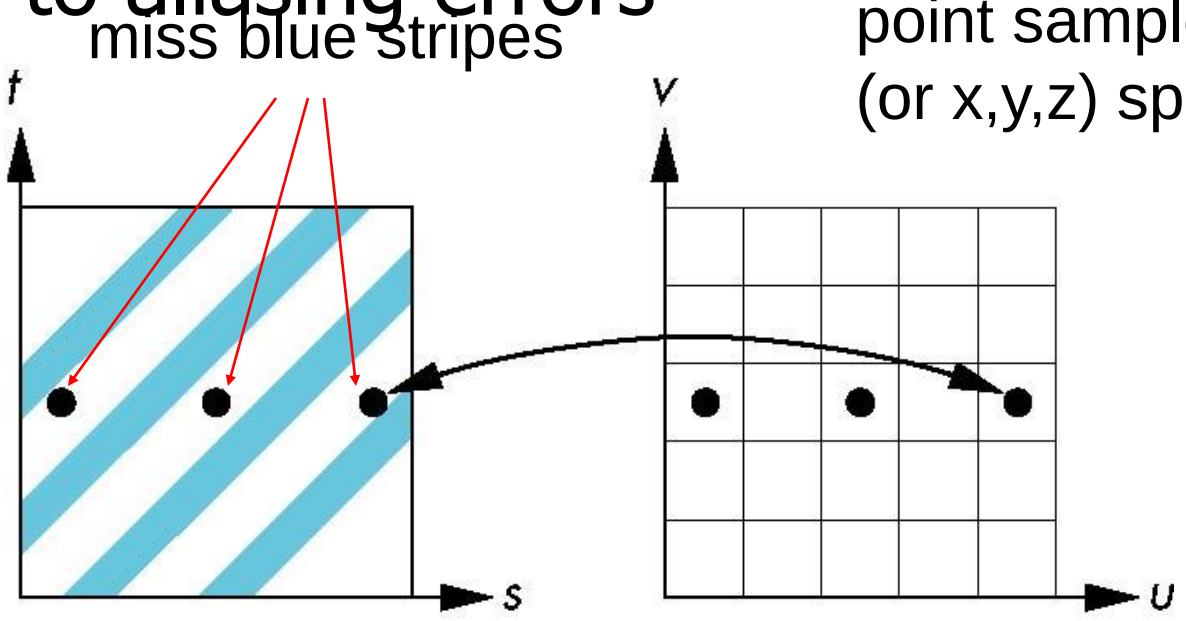
Texture Residency



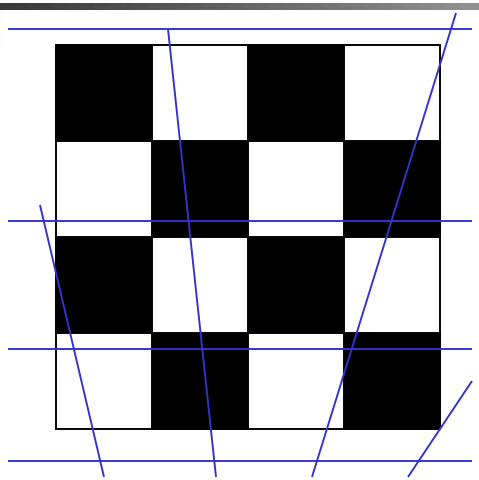
- Working set of textures
  - high-performance, usually hardware accelerated
  - textures must be in texture objects
  - a texture in the *working set* is *resident*
  - for residency of current texture, check **`GL_TEXTURE_RESIDENT`** state
- If too many textures, not all are resident
  - can set priority to have some kicked out first
  - establish 0.0 to 1.0 priorities for texture objects

Aliasing

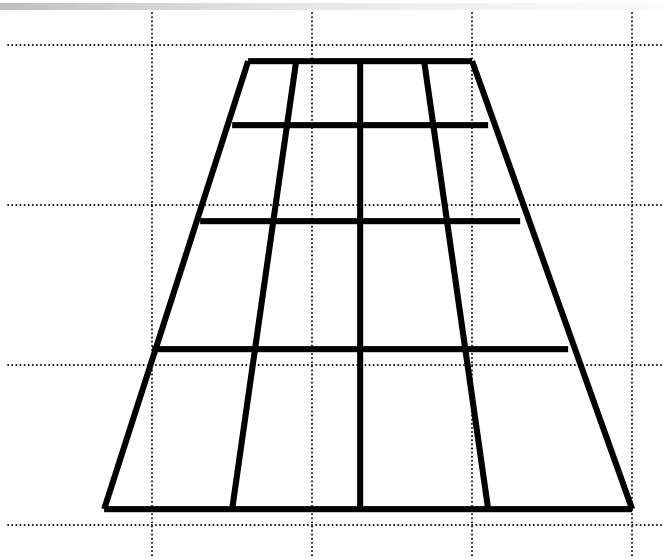
- Point sampling of the texture can lead to aliasing errors



One Pixel to Many Texels



Texture map with pixels drawn on it.
Some pixels cover many texture elements (texels)



Polygon far from the viewer in perspective projection