Computer Graphics

Bing-Yu Chen
National Taiwan University

Introduction to OpenGL

- General OpenGL Introduction
- An Example OpenGL Program
- Drawing with OpenGL
- Transformations
- Animation and Depth Buffering
- Lighting
- Evaluation and NURBS
- Texture Mapping
- Advanced OpenGL Topics
- Imaging

modified from
Dave Shreiner, Ed Angel, and Vicki Shreiner.
An Interactive Introduction to OpenGL Programming.
ACM SIGGRAPH 2001 Conference Course Notes #54.
& ACM SIGGRAPH 2004 Conference Course Notes #29.

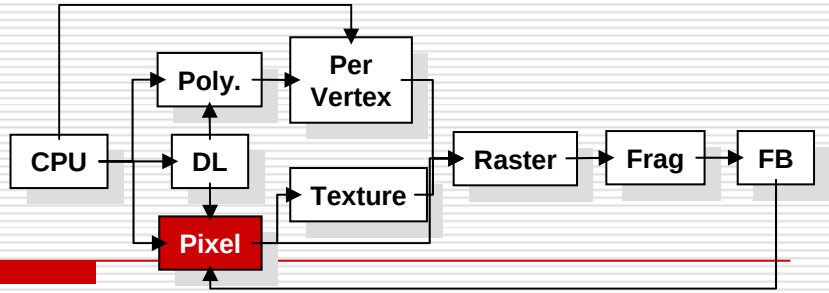
Imaging and Raster Primitives

- Describe OpenGL's raster primitives: bitmaps and image rectangles
- Demonstrate how to get OpenGL to read and render pixel rectangles

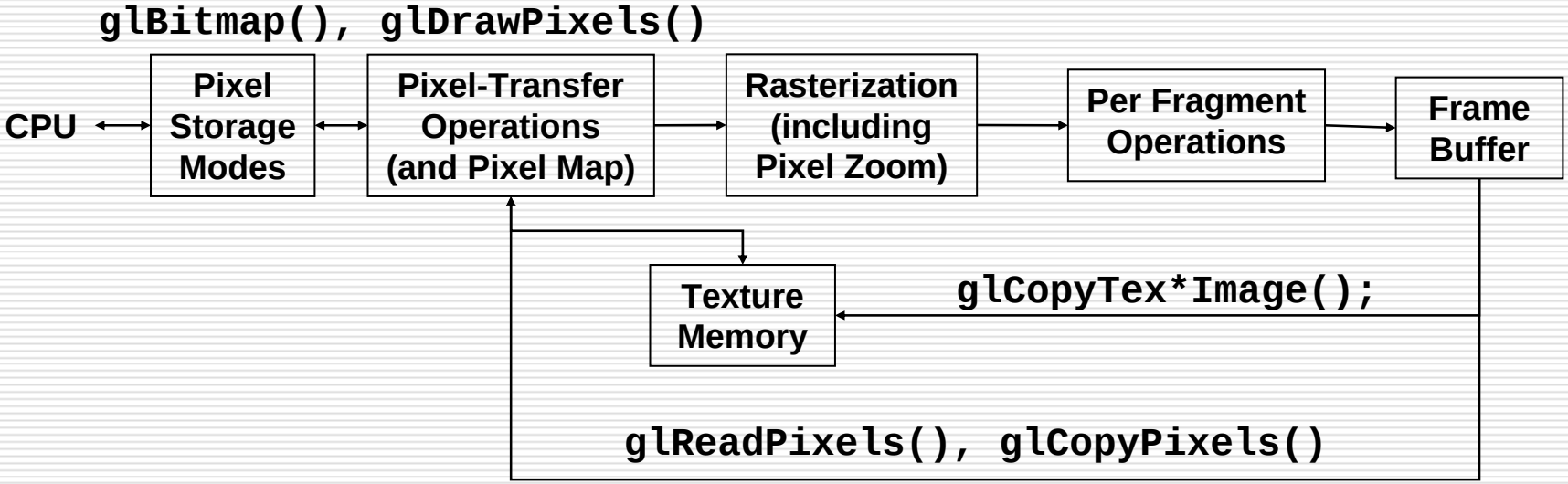
Pixel-based primitives

- Bitmaps
  - 2D array of bit masks for pixels
    - update pixel color based on current color
- Images
  - 2D array of pixel color information
    - complete color information for each pixel
- OpenGL doesn’t understand image formats

Pixel Pipeline



- Programmable pixel storage and transfer operations



Positioning Image Primitives

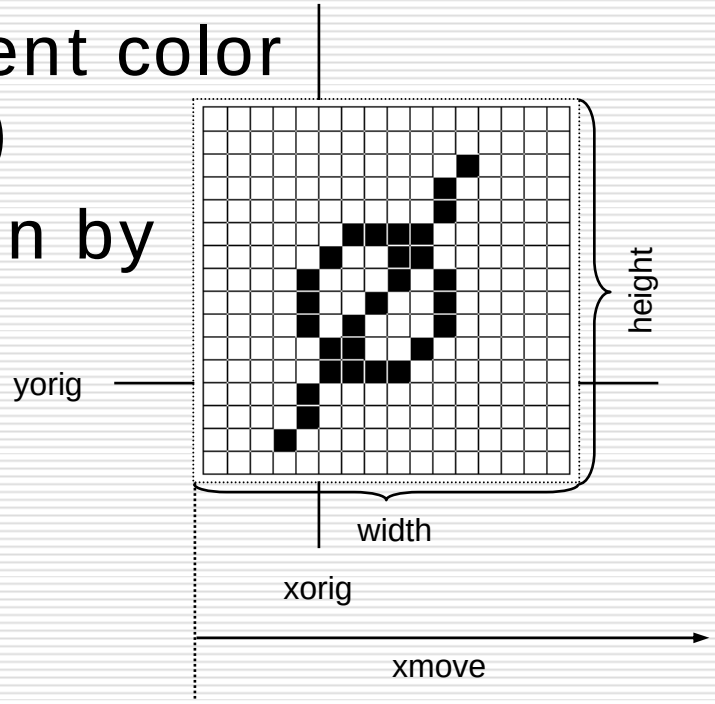
`glRasterPos3f( x, y, z )`

- raster position transformed like geometry
- discarded if raster position is outside of viewport
  - may need to fine tune viewport for desired results

Raster Position

Rendering Bitmaps

- **glBitmap(*width, height, xorig, yorig, xmove, ymove, bitmap*)**
  - render bitmap in current color at $(\lfloor x - xorig \rfloor \quad \lfloor y - yorig \rfloor)$
  - advance raster position by $(xmove \quad ymove)$ after rendering



Rendering Fonts using Bitmaps

- OpenGL uses bitmaps for font rendering
  - each character is stored in a display list containing a bitmap
  - window system specific routines to access system fonts
    - **`glXUseXFont()`**
    - **`wglUseFontBitmaps()`**

Rendering Images

- **glDrawPixels(*width, height, format, type, pixels*)**
  - render pixels with lower left of image at current raster position
  - numerous formats and data types for specifying storage in memory
    - best performance by using format and type that matches hardware

Reading Pixels

- **`glReadPixels( x, y, width, height, format, type, pixels )`**
  - read pixels from specified (x,y) position in framebuffer
  - pixels automatically converted from framebuffer format into requested format and type
- Framebuffer pixel copy
- **`glCopyPixels( x, y, width, height, type )`**

Pixel Zoom

`glPixelZoom( x, y )`

- expand, shrink or reflect pixels around current raster position
- fractional zoom supported

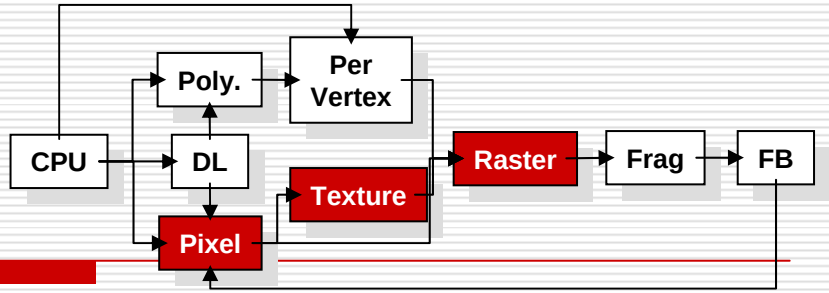
Raster Position

```
glPixelZoom(1.0, -1.0);
```

Storage and Transfer Modes

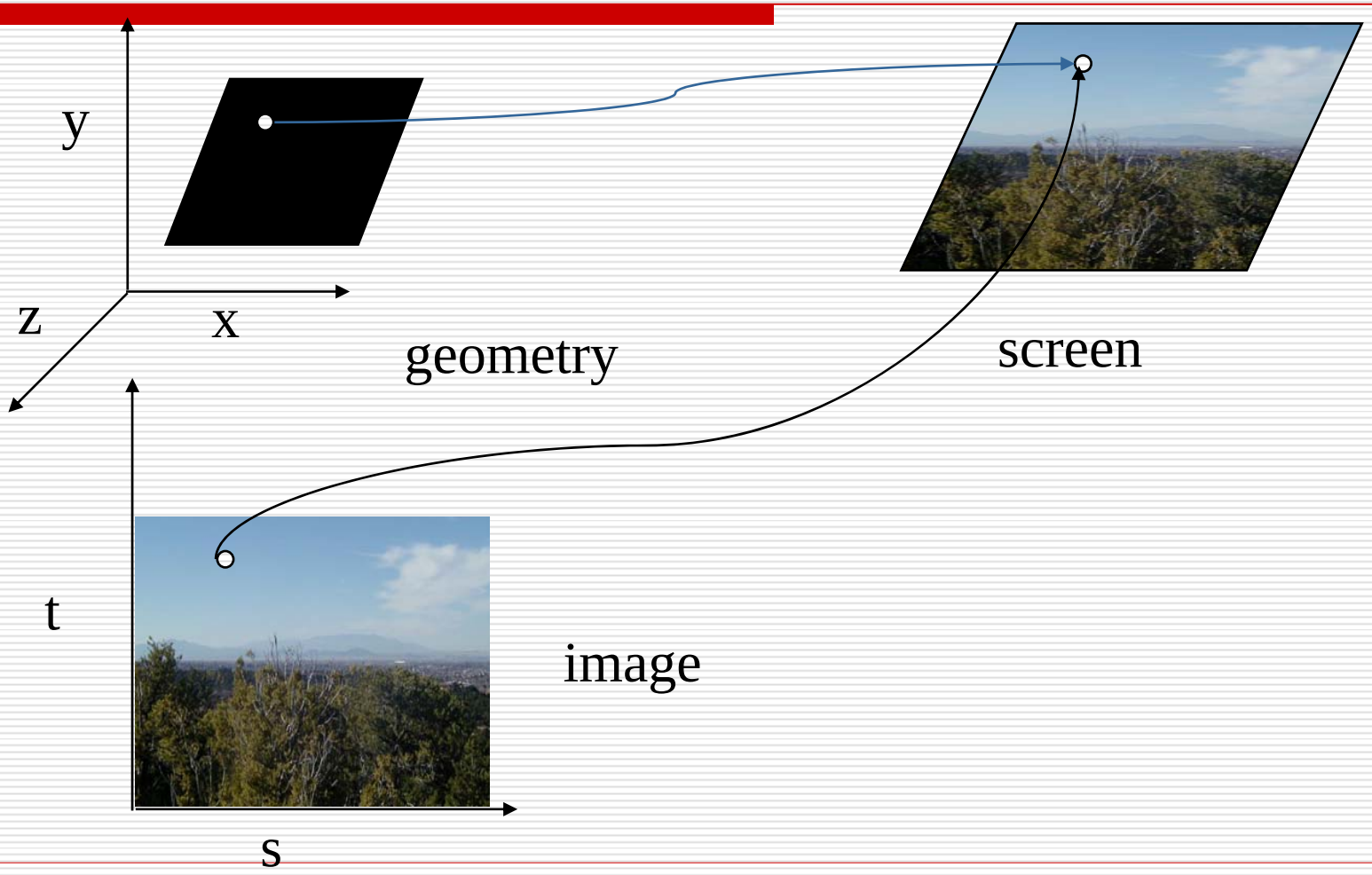
- Storage modes control accessing memory
  - byte alignment in host memory
  - extracting a subimage
- Transfer modes allow modify pixel values
  - scale and bias pixel component values
  - replace colors using pixel maps

Texture Mapping



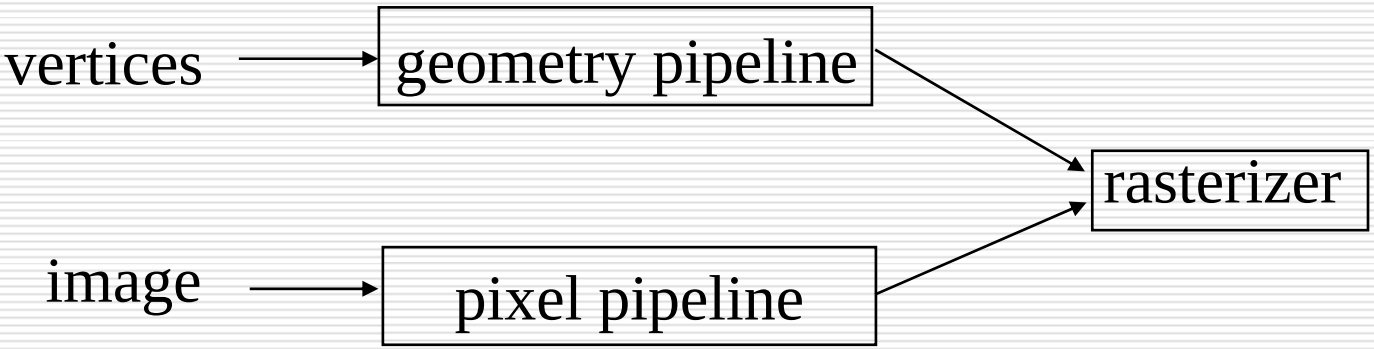
- Apply a 1D, 2D, or 3D image to geometric primitives
- Uses of Texturing
  - simulating materials
  - reducing geometric complexity
  - image warping
  - reflections

Texture Mapping



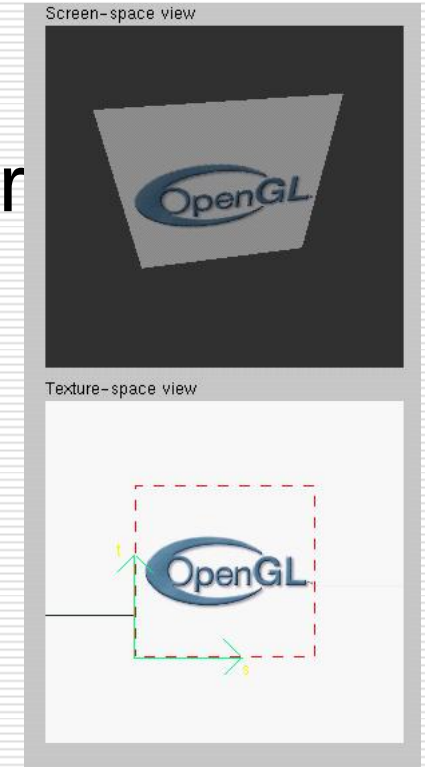
Texture Mapping and the OpenGL Pipeline

- 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 been 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
     - enable texturing
  2. assign texture coordinates to vertices
  3. specify texture parameters
     - wrapping, filtering

Applying Textures II

- specify textures in texture objects
- set texture filter
- set texture function
- set texture wrap mode
- set optional perspective correction hint
- bind texture object
- enable texturing
- 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.)

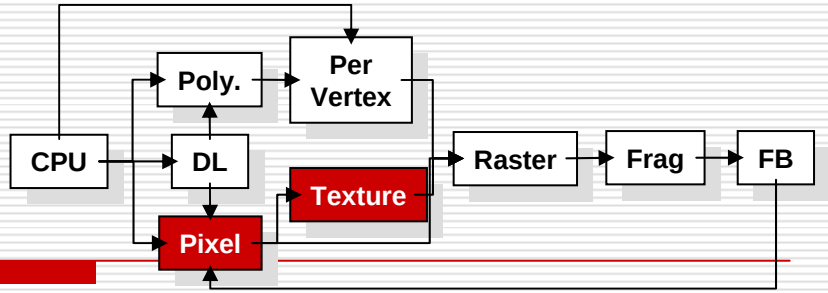
- Create texture objects with texture data and state

  ```
  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

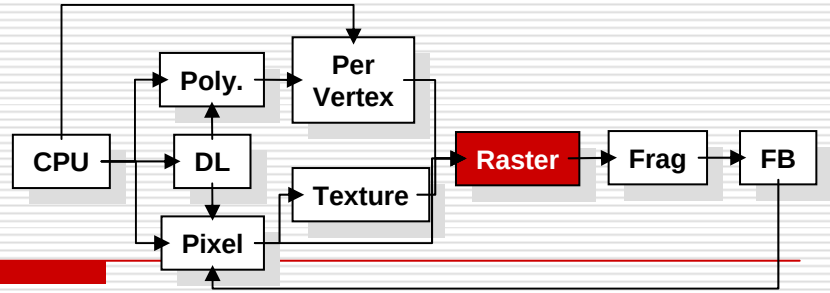
Converting A Texture Image

- 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

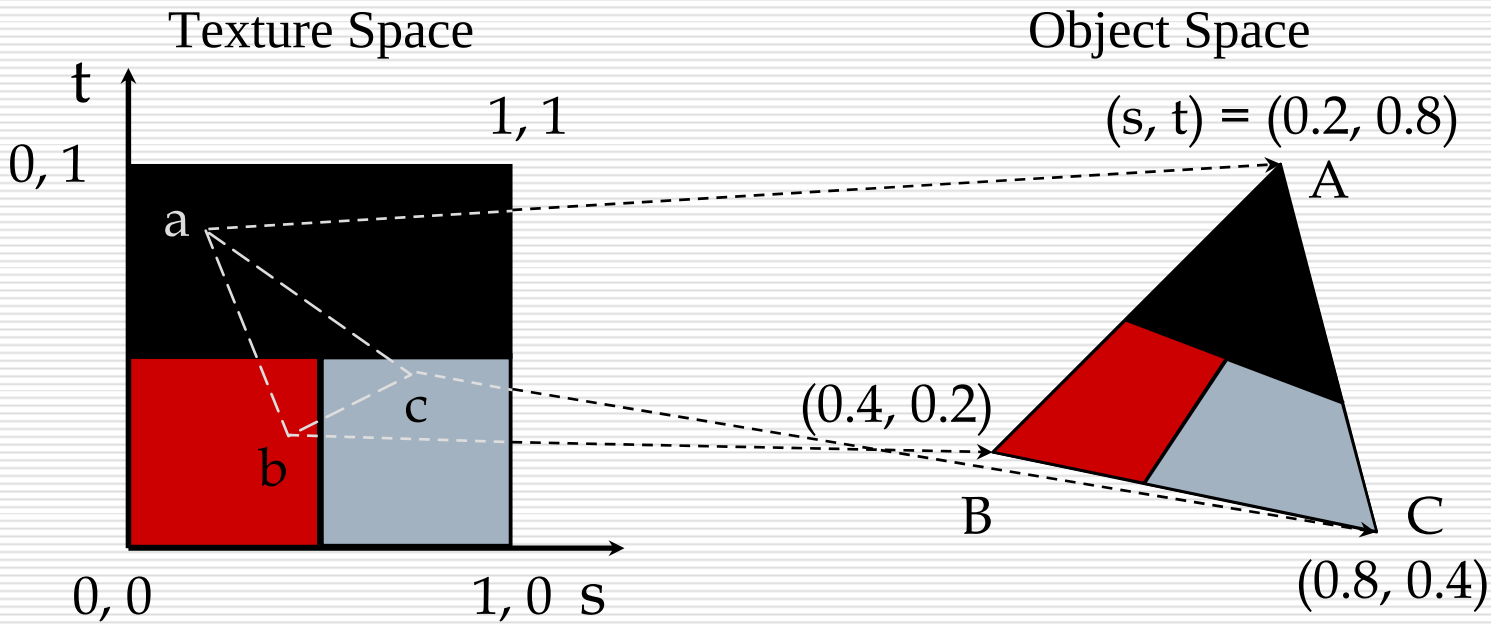
Specifying a Texture: Other Methods

- Use frame buffer as source of texture image
  - uses current buffer as source image
  - **`glCopyTexImage1D(...)`**
  - **`glCopyTexImage2D(...)`**
- Modify part of a defined texture
  - **`glTexSubImage1D(...)`**
  - **`glTexSubImage2D(...)`**
  - **`glTexSubImage3D(...)`**
- 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

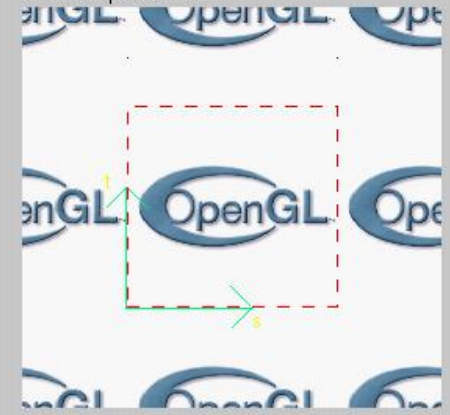
  `glTexGen{ifd}[v]()`

- 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.

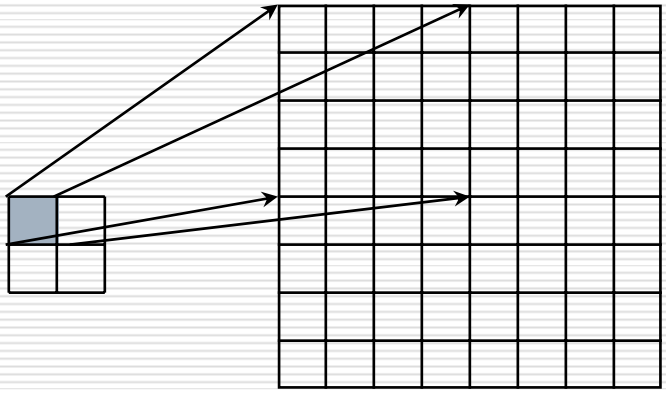
Texture Application Methods

- 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

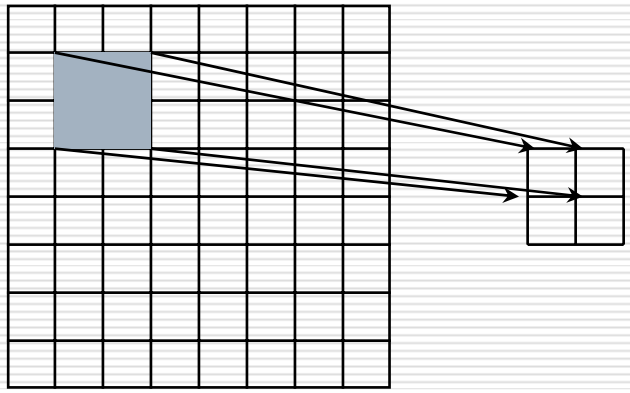
- Example:

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



Texture Polygon

Magnification



Texture Polygon

Minification

Mipmapped Textures

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

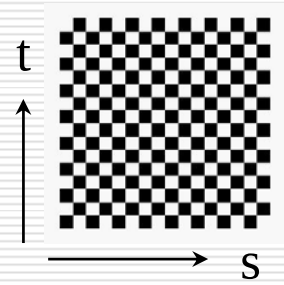
  `glTexImage*D( GL_TEXTURE_*D, level, … )`
- GLU mipmap builder routines

  `gluBuild*DMipmaps( … )`
- OpenGL 1.2 introduces advanced LOD controls

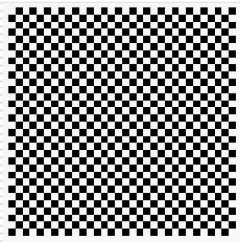
Wrapping Mode

- Example:

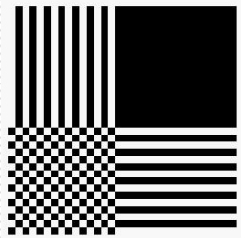
```
glTexParameteri( GL_TEXTURE_2D,
    GL_TEXTURE_WRAP_S, GL_CLAMP )
glTexParameteri( GL_TEXTURE_2D,
    GL_TEXTURE_WRAP_T, GL_REPEAT )
```



texture



GL_REPEAT wrapping



GL_CLAMP wrapping

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