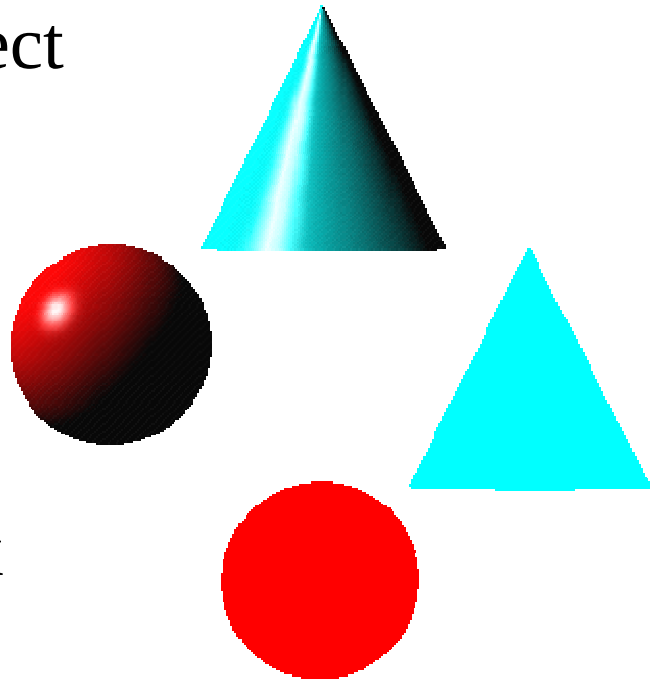
Lighting, Materials, Fog and Textures

Lighting Principles

- Lighting simulates how objects reflect light
  - material composition of object
  - light's color and position
  - global lighting parameters
    - ambient light
    - two sided lighting
  - available in both color index and RGBA mode



Lighting

- Ambient light
  - From no apparent source, simply present
- Diffuse Light
  - Reflected from surface at particular wavelength depending on material
- Specular Light
  - Reflected light, dependent on material

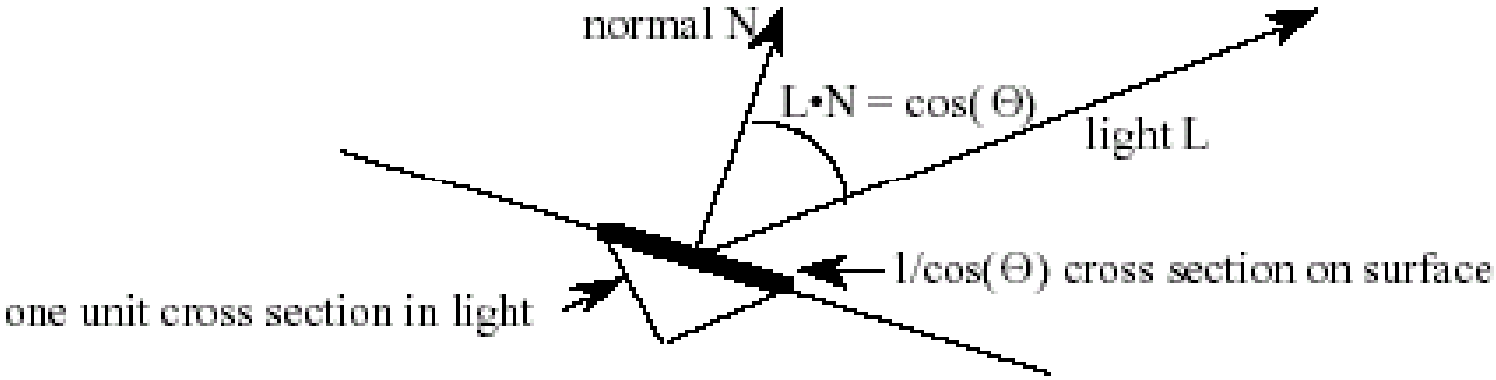
Ambient Light

$$A = L_A * C_A$$

where L_A is the ambient light,

C_A is material dependent

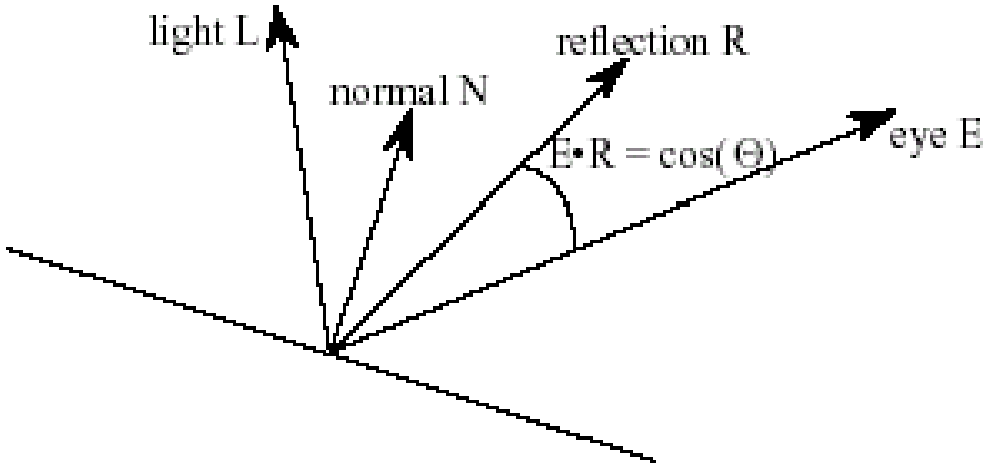
Diffuse Light



$$D = L_D * C_D * \cos(Q) = L_D * C_D * (L \cdot N)$$

where *L* and *N* are unit vectors

Specular Lighting



$$S = L_S * C_S * \cos^n(Q) = L_S * C_S * (E \cdot R)^n$$

where $R = 2\,(N \cdot L)\,N - L$ and
E and R are unit vectors
n is material's specular reflection exponent

Specular Highlights

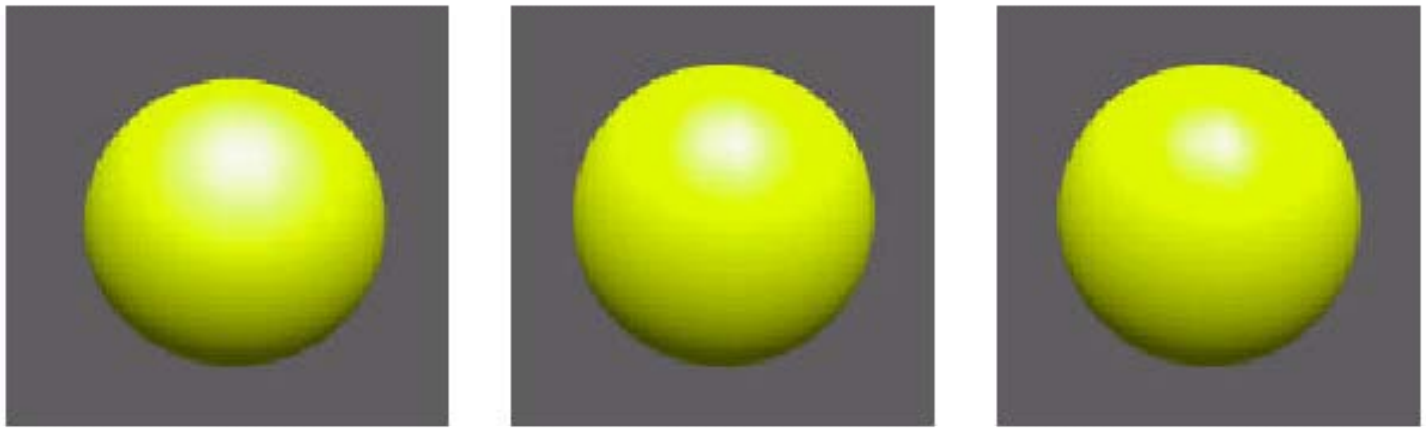


Figure 9.3: specular highlights with specular coefficients 20, 50, and 80 (left, center, and right), respectively

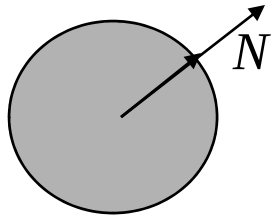
Notice: Higher the coefficient, the shinier; highlight is smaller, more focused

Final Light

- Final light = $\Sigma A + \Sigma D + \Sigma S$ for R, G, B,
- Each value is clamped to 1.
- Light is calculated at vertex
- Need unit length surface normals N

Surface Normals

- For Sphere, center to point on surface

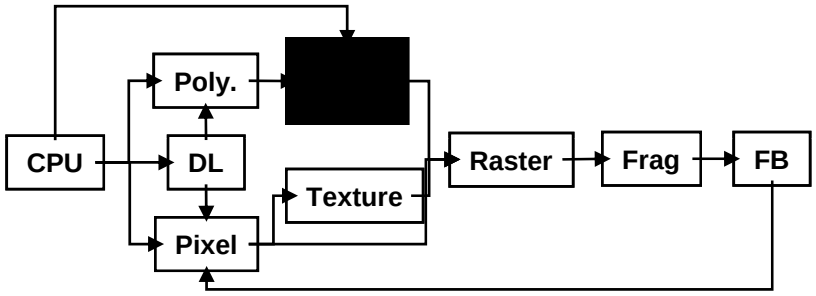


- For Polygon, *Side1 X Side2* (cross product)
- For shared vertices:

$N = (\Sigma\ a_i N_i)/(\Sigma\ a_i)$

where a_i is the inverse cos of the dot product of two edges that meet at the vertex

Surface Normals

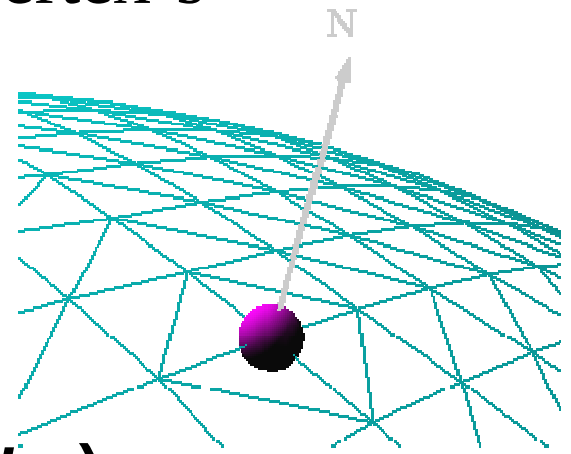


- Normals define how a surface reflects light
  - **`glNormal3f( x, y, z )`**
  - Current normal is used to compute vertex's color
  - Use *unit* normals for proper lighting
    - scaling affects a normal's length

`glEnable( GL_NORMALIZE )`

or

`glEnable( GL_RESCALE_NORMAL )`

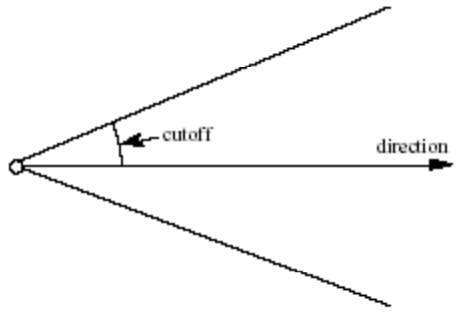


How OpenGL Simulates Lights

- Phong lighting model
  - Computed at vertices
- Lighting contributors
  - Surface material properties
  - Light properties
  - Lighting model properties

Light Properties

- Position or direction
- Color
- How it is attenuated (diminished) over distance
- omni-directional (default) or spotlight
  - direction (3D vector)
  - cutoff (0 to 90)
  - dropoff exponent



Light Properties

- **`glLightfv( light, property, value );`**

– ***`light`*** specifies which light

  - multiple lights, starting with **`GL_LIGHT0`**

    `glGetIntegerv( GL_MAX_LIGHTS, &n );`

– ***`properties`***

  - colors
  - position and type
  - attenuation

Light Sources (cont.)

- Light color properties
  - **GL_AMBIENT**
  - **GL_DIFFUSE**
  - **GL_SPECULAR**

Types of Lights

- OpenGL supports two types of Lights
  - Local (Point) light sources
  - Infinite (Directional) light sources
- Type of light controlled by w coordinate

$w = 0$ ***Infinite Light directed along*** $\begin{pmatrix} x & y & z \end{pmatrix}$

$w \neq 0$ ***Local Light positioned at*** $\begin{pmatrix} x/w & y/w & z/w \end{pmatrix}$

Turning on the Lights

- Flip each light's switch

```
glEnable( GL_LIGHTn );
```

- Turn on the power

```
glEnable( GL_LIGHTING );
```

Controlling a Light's Position

- Modelview matrix affects a light's position
  - Different effects based on when position is specified
    - eye coordinates
    - world coordinates
    - model coordinates
  - Push and pop matrices to uniquely control a light's position

Advanced Lighting Features

- Spotlights
  - localize lighting affects
    - ***GL_SPOT_DIRECTION***
    - ***GL_SPOT_CUTOFF***
    - ***GL_SPOT_EXPONENT***

Advanced Lighting Features

- Light attenuation
  - decrease light intensity with distance
    - **`GL_CONSTANT_ATTENUATION`**
    - **`GL_LINEAR_ATTENUATION`**
    - **`GL_QUADRATIC_ATTENUATION`**

$$f_i = \frac{1}{k_c + k_l d + k_q d^2}$$

Light Model Properties

- **`glLightModelfv( property, value );`**
- Enabling two sided lighting
  `GL_LIGHT_MODEL_TWO_SIDE`
- Global ambient color
  `GL_LIGHT_MODEL_AMBIENT`
- Local viewer mode
  `GL_LIGHT_MODEL_LOCAL_VIEWER`
- Separate specular color
  `GL LIGHT MODEL COLOR CONTROL`

Tips for Better Lighting

- Recall lighting computed only at vertices
  - model tessellation heavily affects lighting results
    - better results but more geometry to process
- Use a single infinite light for fastest lighting
  - minimal computation per vertex

Lights in OpenGL

- **glEnable**(GL_LIGHTING)
  - If enabled, use the current lighting parameters to compute the vertex color or index. Otherwise, simply associate the current color or index with each vertex.
- **glEnable**(GL_LIGHT_i)
  - If enabled, include light i in the evaluation of the lighting equation.
- **glEnable**(GL_NORMALIZE)
  - If enabled, normal vectors specified with ***glNormal*** are scaled to unit length after transformation.
  - before any geometry is specified, will automatically normalize vectors!

glLightModel[f,i] (*pname*, *param*)

- Set the lighting model parameters
- *pname*
  - GL_LIGHT_MODEL_AMBIENT
  - GL_LIGHT_MODEL_LOCAL_VIEWER
  - GL_LIGHT_MODEL_TWO_SIDE
- *param* (with respect to *pname*)
  - ambient RGBA intensity of the entire scene
  - how specular reflection angles are computed. 0 (default) view direction to be parallel to and in the direction of the -z-axis. Otherwise, from the origin of the eye coordinate system.
  - specifies whether one- (0, front only, default) or two-sided lighting (non-zero) calculations are done for polygons

glLight[f,I]**v**(light, pname, *params)

- *light* - Specifies a light: **GL_LIGHT$_i$**
- *pname* - Specifies a light source parameter for *light*:
  - GL_AMBIENT — ambient intensity, RGBA, default (0, 0, 0, 1)
  - GL_DIFFUSE — diffuse intensity, RGBA, default (1, 1, 1, 1)
  - GL_SPECULAR — specular intensity, RGBA, default (1, 1, 1, 1)
  - GL_POSITION — light position, world coords, default (0, 0, 1, 0, directional, parallel to z-axis)
  - GL_SPOT_DIRECTION — eye coords, default (0,0,-1)
  - GL_SPOT_EXPONENT — intensity distribution [0, 128], default (0)
  - GL_SPOT_CUTOFF — maximum spread angle [0, 90], default 180
  - GL_CONSTANT_ATTENUATION — default 1
  - GL_LINEAR_ATTENUATION — default 0
  - GL_QUADRATIC_ATTENUATION — default 0
- *params* - Specifies a pointer to the value or values that parameter pname of light source light will be set to (See column 2).

glNormal3[b,d,f,i,s](nx, ny, nz)
glNormal3[b,d,f,i,s]**v**(*v)

- Set the current normal vector (for a vertex)
- Specify the x, y, and z coordinates of the new current normal. The initial value of the current normal is (0,0,1)

OR

- Specifies a pointer to an array of three elements: the x, y, and z coordinates of the new current normal.

Material Properties

- C_A - Ambient light coefficient
- C_D - Diffuse light coefficient
- C_S - Specular light coefficient
- N - Surface normal

Material Properties

- Define the surface properties of a primitive
- **`glMaterialfv( face, property, value );`**

`GL_DIFFUSE`	Base color
`GL_SPECULAR`	Highlight Color
`GL_AMBIENT`	Low-light Color
`GL_EMISSION`	Glow Color
`GL_SHININESS`	Surface Smoothness

  - separate materials allowed for front and back

Materials in OpenGL

- **glMaterial**[f,i]**v**(face, pname, *params)

  OR

- **glColorMaterial**(face, mode) (preferred)
  - ***glColorMaterial*** specifies which material parameters track the current color.
  - ***glEnable (GL_COLOR_MATERIAL)*** needed
  - ***glColorMaterial*** allows a subset of material parameters to be changed for each vertex using only the ***glColor*** command, without calling ***glMaterial***. If only such a subset of parameters is to be specified for each vertex, ***glColorMaterial*** is preferred over calling ***glMaterial***.

glMaterial[f,i]**v**(face, pname, *params)

- Specify material parameters for the lighting model
- *face*
  - GL_FRONT
  - GL_BACK
  - GL_FRONT_AND_BACK
- *pname*

	params (default)
– GL_AMBIENT	ambient RGBA reflectance (0.2, 0.2, 0.2, 1.0)
– GL_DIFFUSE	diffuse RGBA reflectance (0.8, 0.8, 0.8, 1.0)
– GL_SPECULAR	specular RGBA reflectance (0.0,0.0, 0.0, 1.0)
– GL_EMISSION	RGB emitted light intensity (0.0,0.0, 0.0, 1.0)
– GL_SHININESS	specular exponent, range [0,128] (0)
– GL_AMBIENT_AND_DIFFUSE	equivalent to calling glMaterial twice
– GL_COLOR_INDEXES	color indices for ambient, diffuse, and specular lighting - RGB

glColorMaterial(face, mode)

- *face*
  - GL_FRONT
  - GL_BACK
  - GL_FRONT_AND_BACK
- *mode* - Specifies which of several material parameters track the current color.
  - GL_EMISSION
  - GL_AMBIENT
  - GL_DIFFUSE
  - GL_SPECULAR
  - GL_AMBIENT_AND_DIFFUSE (default)

Fog in OpenGL

- glEnable(GL_FOG)
  - If enabled, blend a fog color into the post texturing color.
- glFog(…)
  - specify fog parameters
- glHint(GL_FOG_HINT, mode)
  - specify implementation-specific hints
  - *mode*: GL_FASTEST, GL_NICEST, and GL_DONT_CARE

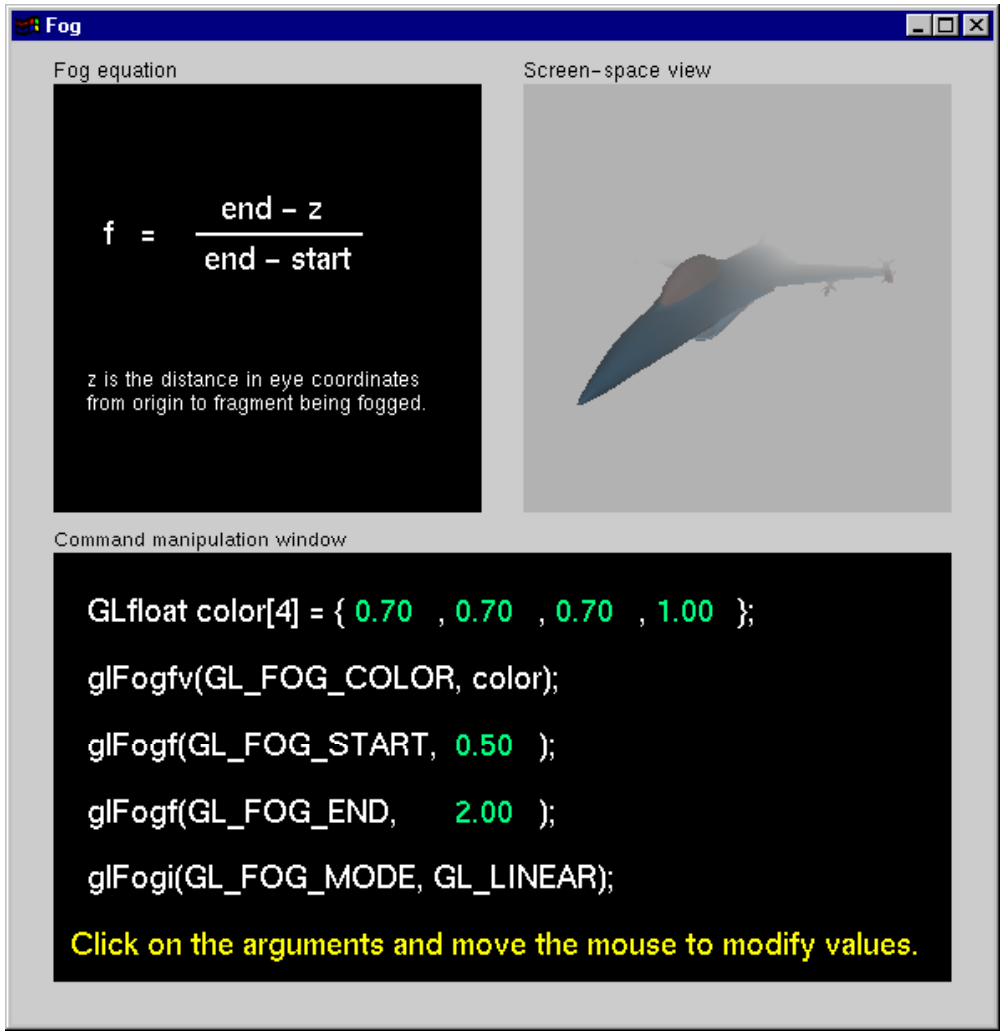
Fog

- **`glFog{if}(` *`property, value`* `)`**
- Depth Cueing
  - Specify a range for a linear fog ramp
    - **`GL_FOG_LINEAR`**
- Environmental effects
  - Simulate more realistic fog
    - **`GL_FOG_EXP`**
    - **`GL_FOG_EXP2`**

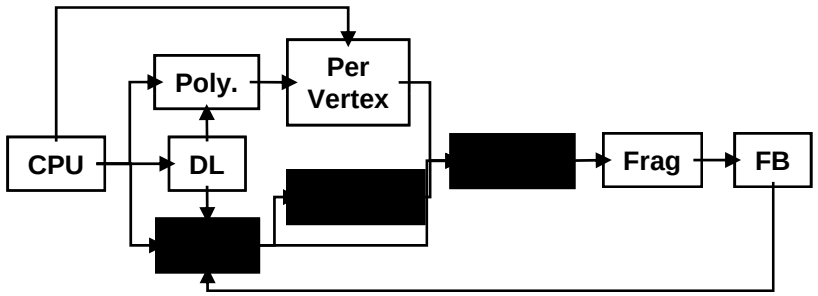
glFog[f,i]**v**(pname, *params)

- *pname* — *params (default*
  - GL_FOG_MODE — specifies the equation to be used to compute the fog blend factor, *f*: GL_LINEAR, (GL_EXP), and GL_EXP2
  - GL_FOG_DENSITY — the fog density, >= 0 (1)
  - GL_FOG_START — near distance used in the linear fog equation (0.0)
  - GL_FOG_END — the far distance used in the linear fog equation (1)
  - GL_FOG_INDEX — the fog color index (0)
  - GL_FOG_COLOR — RGBA, fog color, [0,1] (0, 0, 0, 0)

Fog Tutorial

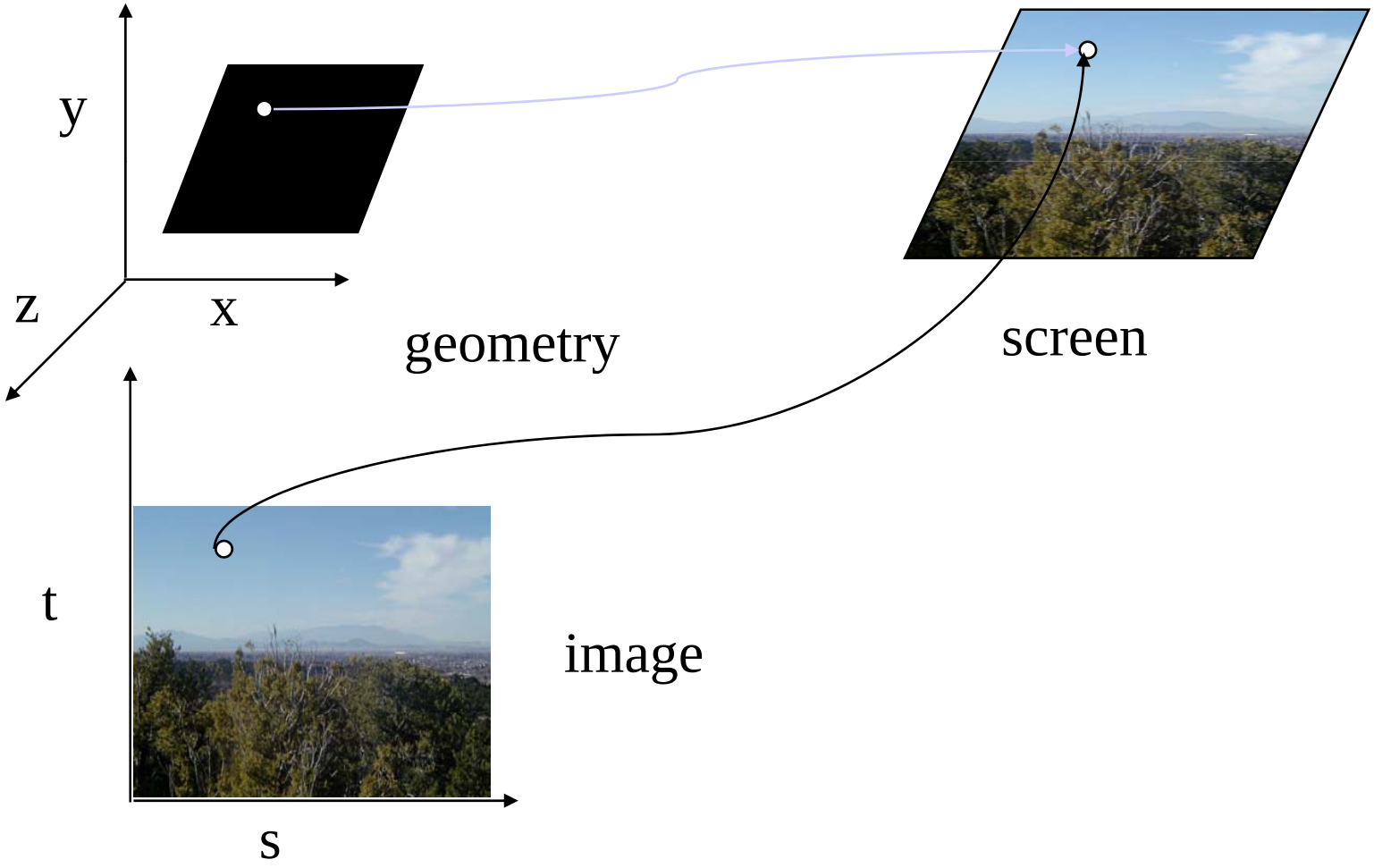


Texture Mapping



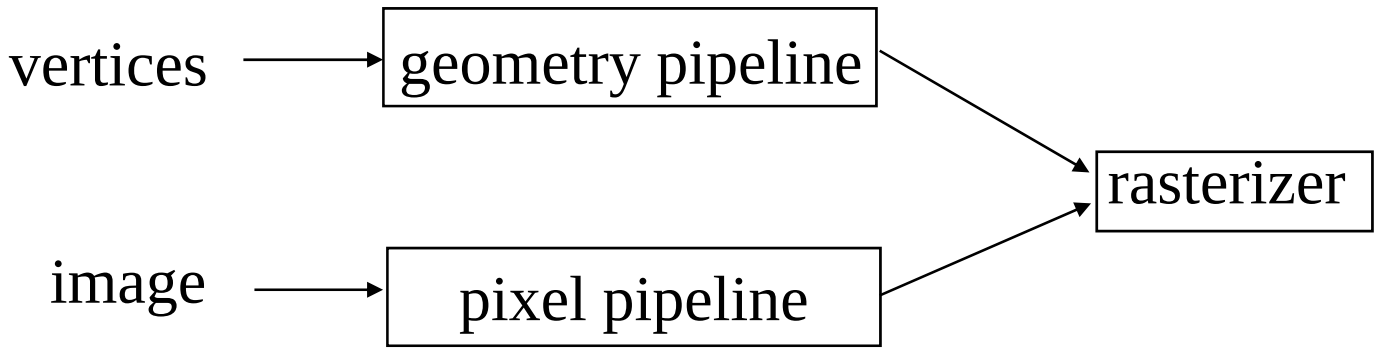
- Apply a 1D, 2D, or 3D image to geometric primitives
- Uses of Texturing
  - simulating materials
  - reducing geometric complexity
  - image warping
  - reflections

Texture Mapping



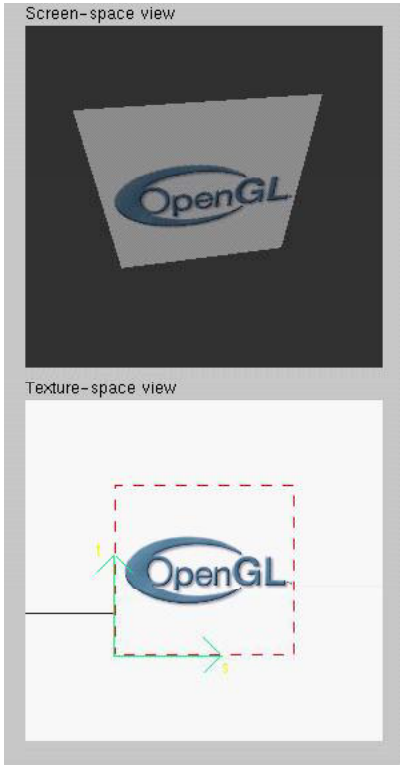
Texture Mapping and the OpenGL Pipeline

- 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
- Create texture objects with texture data and state

Texture Objects (cont.)

- Generate texture names

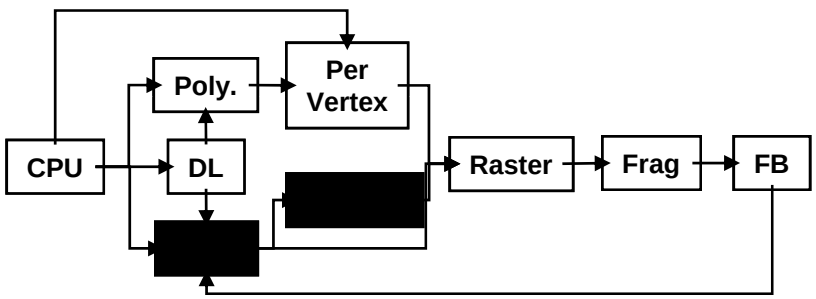
  ```
  glGenTextures( n, *texIds );
  ```

  n is the number of names to generate
- Bind textures before using

  ```
  glBindTexture( target, texId );
  ```

  target is a number
- Essential when more than one texture:
  - In OpenGL there is a SINGLE current texture
  - Reloading textures is inefficient
  - Allows more than one texture to be resident in texture memory even though current texture changes

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

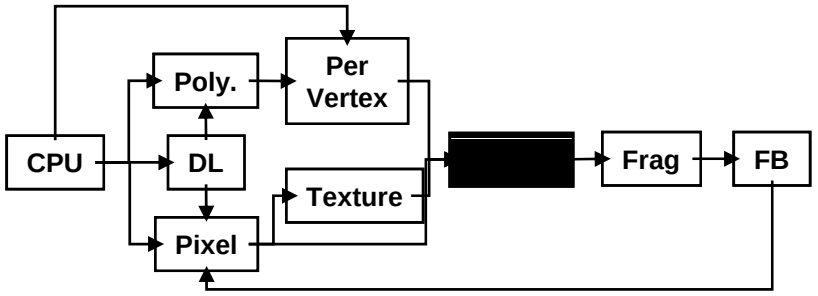
Specifying a Texture: Other Methods

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

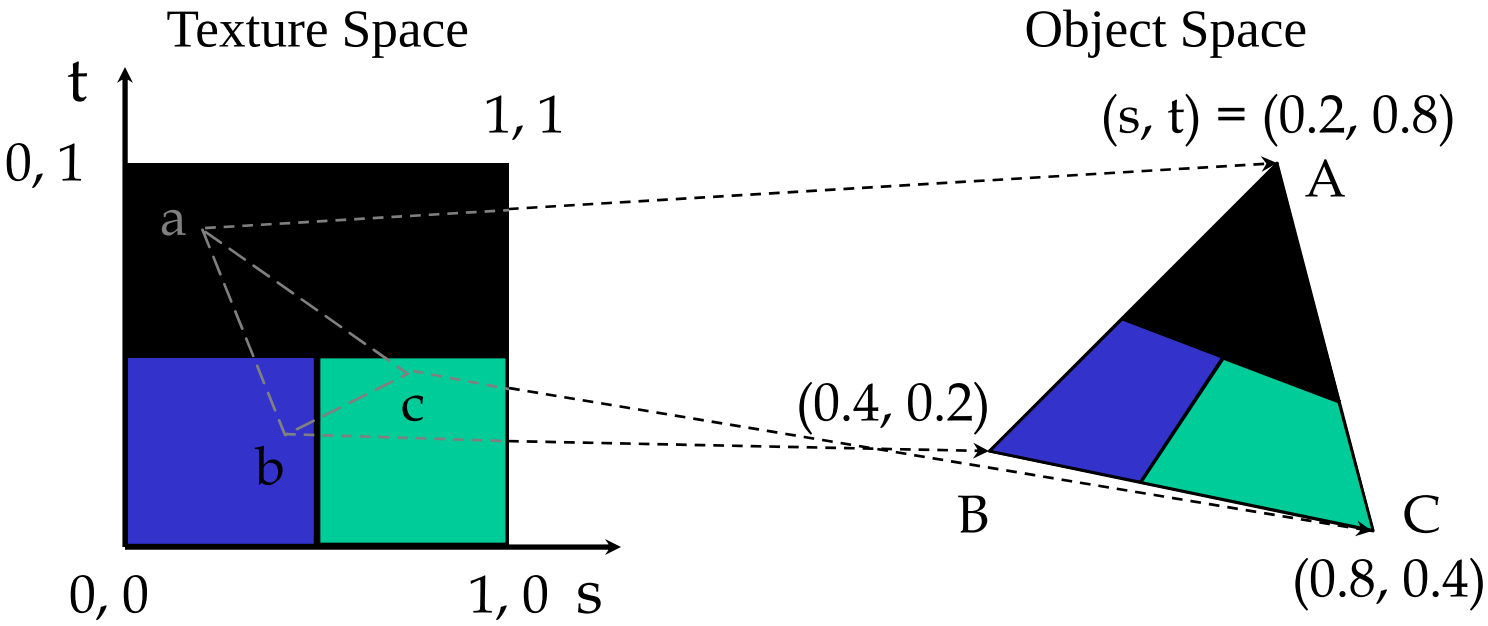
    `glCopyTexImage1D(...)`
    `glCopyTexImage2D(...)`
- Modify part of a defined texture

    `glTexSubImage1D(...)`
    `glTexSubImage2D(...)`
    `glTexSubImage3D(...)`
- Do both with ***`glCopyTexSubImage2D(...)`***, etc.

Mapping a Texture



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



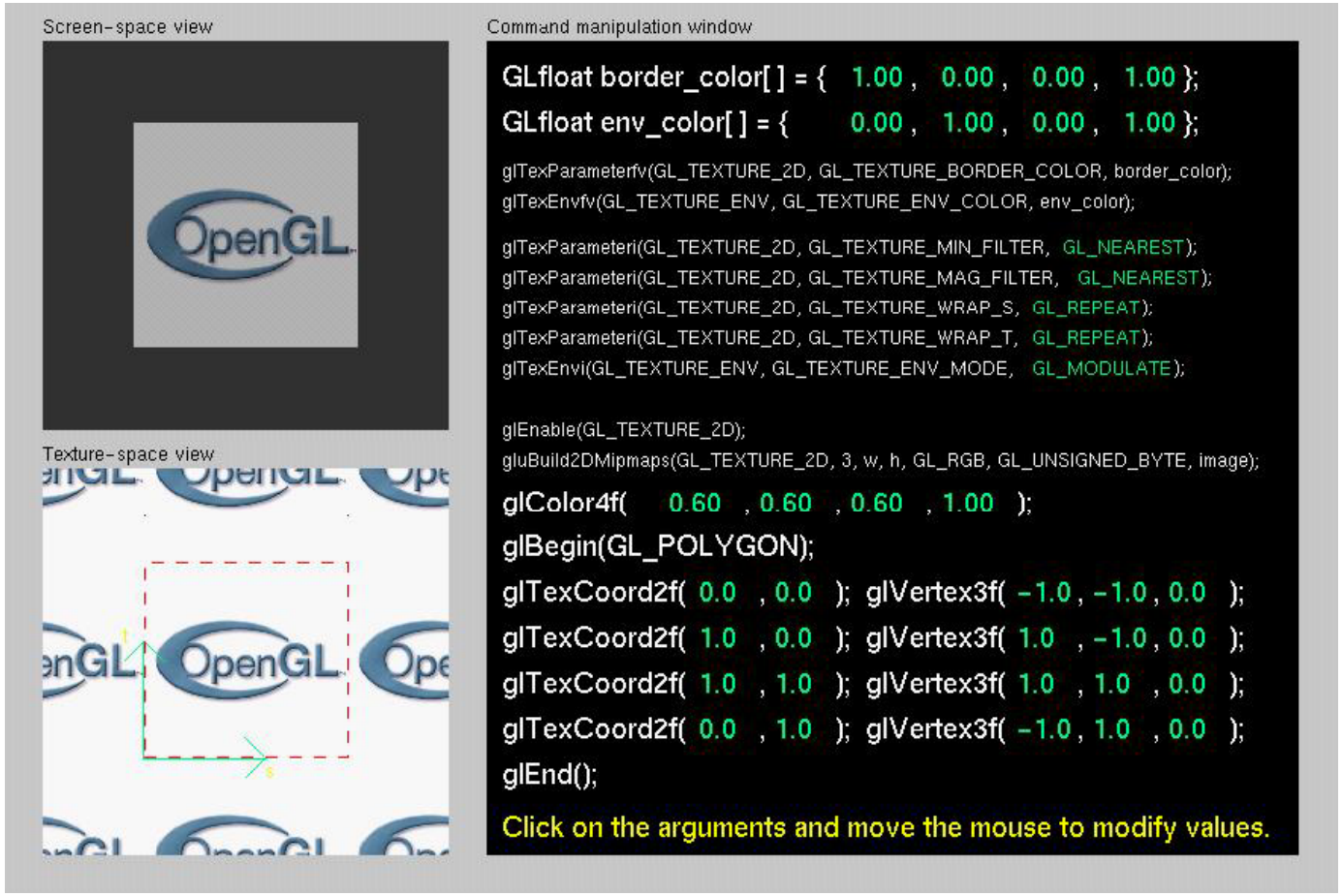
Generating Texture Coordinates

- 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



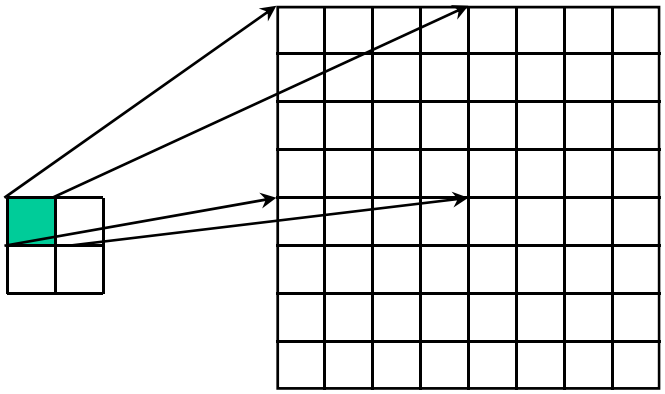
Texture Application Methods

- 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

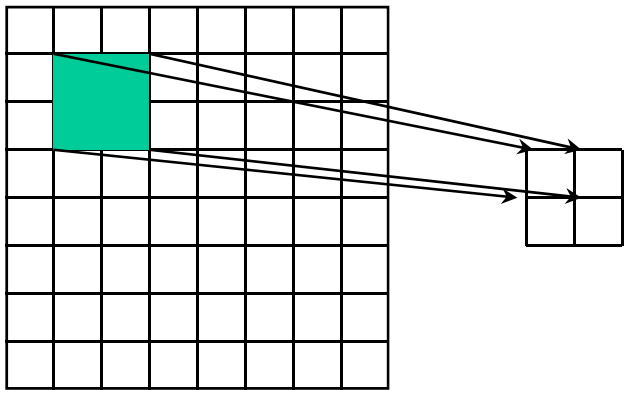
Example:

```
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 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

Texture in OpenGL

- glTexCoord - Specifies texture coordinate to use for this vertex.
- glTexEnv - Specifies a texture environment
- glTexGen - Control the generation of texture coordinates
- glTexImage1D - Specify a one-dimensional texture image
- glTexImage2D - Specify a one-dimensional texture image
- glTexParameter - Set texture parameters

glTexCoord[1,2,3,4][d,f,i,s]**v**(*v)

- Set the current texture coordinates
- *v*
  - Specifies a pointer to an array of one, two, three, or four elements, which in turn specify the *s*, *t*, *r*, and *q* texture coordinates.
- The current texture coordinates are part of the data that is associated with polygon vertices.

glTexEnv[f,]**v**(target, pname, *params)

- *target* - a texture environment, *GL_TEXTURE_ENV*.
- *pname* - texture environment parameter name
  - GL_TEXTURE_ENV_MODE
  - GL_TEXTURE_ENV_COLOR
- **params* - a pointer to an array of parameters:
  - symbolic constant GL_MODULATE, GL_DECAL, GL_BLEND, GL_REPLACE
  - RGBA color

glTexGen[d,f,i]**v**(coord, pname, *params)

- *coord* - texture coordinate. Must be one of: GL_S, GL_T, GL_R, or GL_Q.
- *pname* - texture-coordinate generation function: GL_TEXTURE_GEN_MODE, GL_OBJECT_PLANE, or GL_EYE_PLANE.
- *params* - a pointer to an array of texture generation parameters: If *pname* is GL_TEXTURE_GEN_MODE: GL_OBJECT_LINEAR, GL_EYE_LINEAR, or GL_SPHERE_MAP. Otherwise, the coefficients for the texture-coordinate generation function specified by *pname*.

glTexImage[1,2]**D**(target, level, components, width, border, format, type, *pixels)

- *target* - GL_TEXTURE_[1,2]D
- *level* - 0 is the base image, n - nth mipmap reduction image.
- *components* - number of color components in the texture: 1, 2, 3, or 4.
- *width* - width of the texture image.
- *border* - width of the border (0 or 1).
- *format* - format of the pixel data. : GL_COLOR_INDEX, GL_RED, GL_GREEN, GL_BLUE, GL_ALPHA, GL_RGB, GL_RGBA, GL_LUMINANCE, and GL_LUMINANCE_ALPHA.
- *type* - data type of the pixel data: GL_UNSIGNED_BYTE, GL_BYTE, GL_BITMAP, GL_UNSIGNED_SHORT, GL_SHORT, GL_UNSIGNED_INT, GL_INT, and GL_FLOAT.
- pixels - pointer to the image data in memory.

glTexParameter[f,i]**v**(target, pname, *params)

- *target* - the target texture: GL_TEXTURE_1D or GL_TEXTURE_2D.
- *pname* - symbolic name of a texture parameter: GL_TEXTURE_MIN_FILTER, GL_TEXTURE_MAG_FILTER, GL_TEXTURE_WRAP_S, GL_TEXTURE_WRAP_T, or GL_TEXTURE_BORDER_COLOR.
- *params* - pointer to an array where the value or values of *pname* are stored.

Texture Map from an Image

- ***glReadBuffer*** (mode) - select a color buffer source for pixels
  - GL_FRONT_LEFT, GL_FRONT_RIGHT, GL_BACK_LEFT, GL_BACK_RIGHT, GL_FRONT, GL_BACK, GL_LEFT, GL_RIGHT, and GL_AUXi (GL_FRONT in single-buffered and GL_BACK in double-buffered)
- ***glReadPixels*** (x, y, width, height, format, type, *pixels) - read a block of pixels from the frame buffer
  - x, y - the window coordinates (lower left corner) of the first pixel that is read from the frame buffer.
  - width, height - dimensions of the pixel rectangle..
  - Format - of the pixel data: GL_COLOR_INDEX, GL_STENCIL_INDEX, GL_DEPTH_COMPONENT, GL_RED, GL_GREEN, GL_BLUE, GL_ALPHA, GL_RGB, GL_RGBA, GL_LUMINANCE, and GL_LUMINANCE_ALPHA.
  - Type - the data type of the pixel data: GL_UNSIGNED_BYTE, GL_BYTE, GL_BITMAP, GL_UNSIGNED_SHORT, GL_SHORT, GL_UNSIGNED_INT, GL_INT, or GL_FLOAT.
  - Pixels - Returns the pixel data.
- Others: ***glCopyPixels***, ***glDrawPixels*** , ***glPixelMap*** , ***glPixelStore*** , ***glPixelTransfer***