Texture Scanning Process

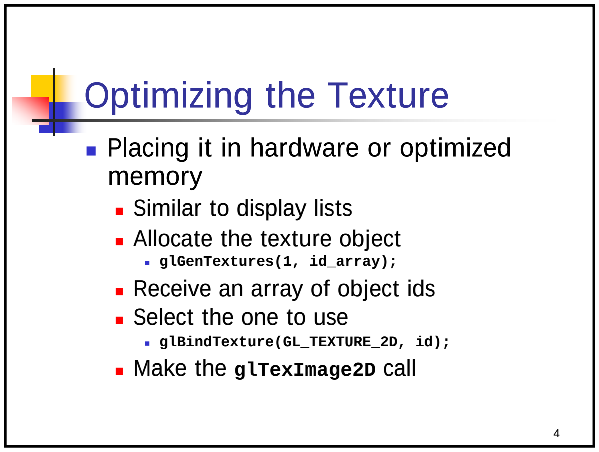
- Recall from last time
  1. Generate the texture pattern
  2. Choose how it will be “mixed”
  3. Position relative to the object face/image fragment
  4. Apply to the fragment

“Storing” a Texture for OpenGL

- Need an array of texels
  - Size of the texture map
  - Color information specified
    - RGB, RGBA, Red only, Luminance, etc.
    - See Table 8-1 GL pp. 309-10
  - Packing
    - How many bits per component?
    - See Table 8-2 GL pp. 310-11

Giving the Texture to OpenGL

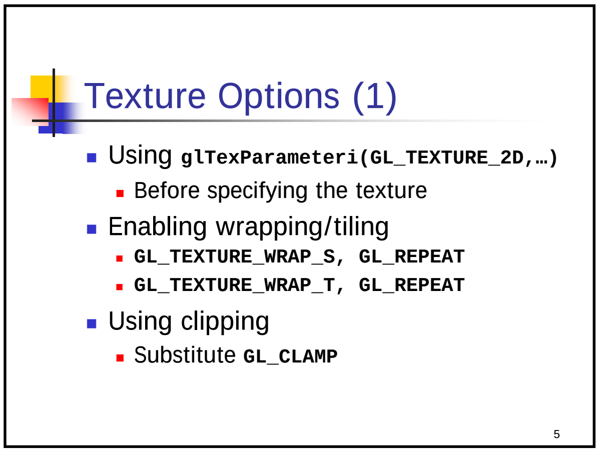
- **`glTexImage2D(GL_TEXTURE_2D, level, components, width, height, border, format, packing, texel array)`**
  - level – Levels of resolution (0 for simple)
  - components – Internal format (not guaranteed)
  - width, height – texture size
  - border – extra dummy pixels for anti-aliasing
  - format – Table 8-1
  - packing – Table 8-2
  - texel array – 2D array matching packing



Optimizing the Texture

- Placing it in hardware or optimized memory
  - Similar to display lists
  - Allocate the texture object
    - `glGenTextures(1, id_array);`
  - Receive an array of object ids
  - Select the one to use
    - `glBindTexture(GL_TEXTURE_2D, id);`
  - Make the **`glTexImage2D`** call

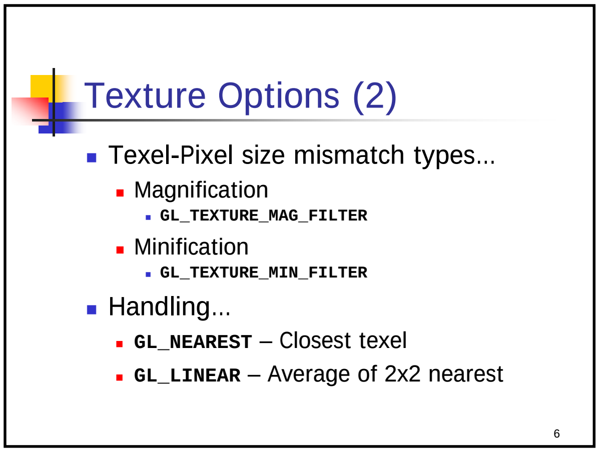
4



Texture Options (1)

- Using **`glTexParameteri(GL_TEXTURE_2D,…)`**
  - Before specifying the texture
- Enabling wrapping/tiling
  - **`GL_TEXTURE_WRAP_S, GL_REPEAT`**
  - **`GL_TEXTURE_WRAP_T, GL_REPEAT`**
- Using clipping
  - Substitute **`GL_CLAMP`**

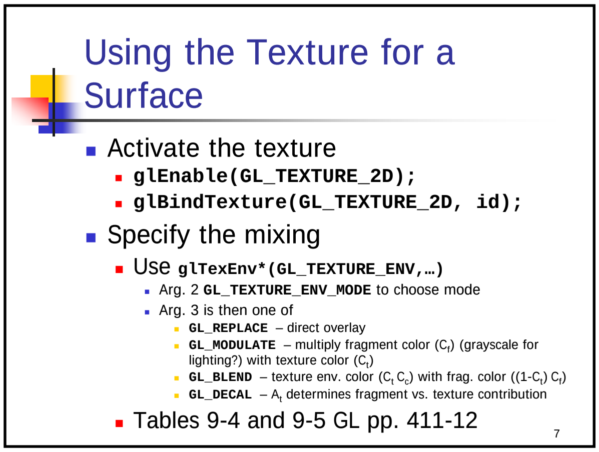
5



Texture Options (2)

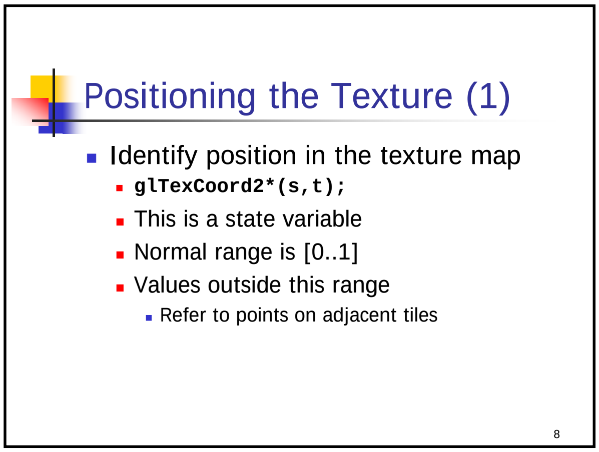
- Texel-Pixel size mismatch types…
  - Magnification
    - **`GL_TEXTURE_MAG_FILTER`**
  - Minification
    - **`GL_TEXTURE_MIN_FILTER`**
- Handling…
  - **`GL_NEAREST`** – Closest texel
  - **`GL_LINEAR`** – Average of 2x2 nearest

6



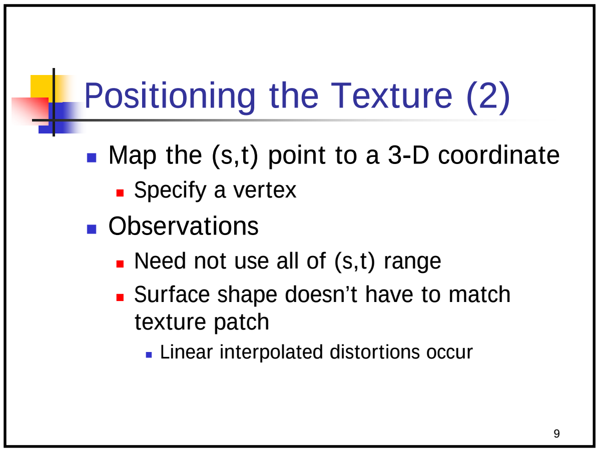
Using the Texture for a Surface

- Activate the texture
  - **`glEnable(GL_TEXTURE_2D);`**
  - **`glBindTexture(GL_TEXTURE_2D, id);`**
- Specify the mixing
  - Use **`glTexEnv*(GL_TEXTURE_ENV,…)`**
    - Arg. 2 **`GL_TEXTURE_ENV_MODE`** to choose mode
    - Arg. 3 is then one of
      - **`GL_REPLACE`** – direct overlay
      - **`GL_MODULATE`** – multiply fragment color (C_f) (grayscale for lighting?) with texture color (C_t)
      - **`GL_BLEND`** – texture env. color (C_t C_c) with frag. color ((1-C_t) C_f)
      - **`GL_DECAL`** – A_t determines fragment vs. texture contribution
  - Tables 9-4 and 9-5 GL pp. 411-12



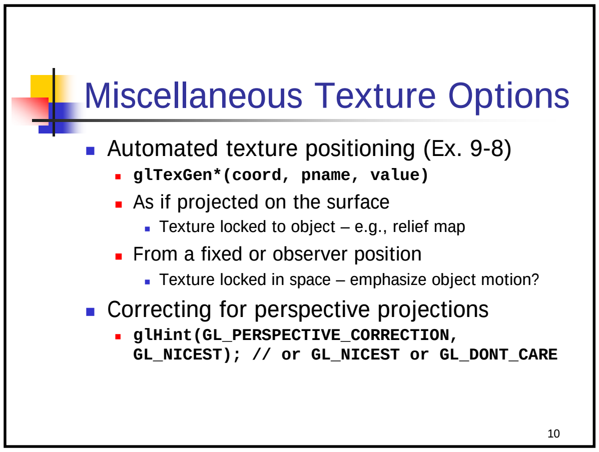
Positioning the Texture (1)

- Identify position in the texture map
  - **`glTexCoord2*(s,t);`**
  - This is a state variable
  - Normal range is [0..1]
  - Values outside this range
    - Refer to points on adjacent tiles



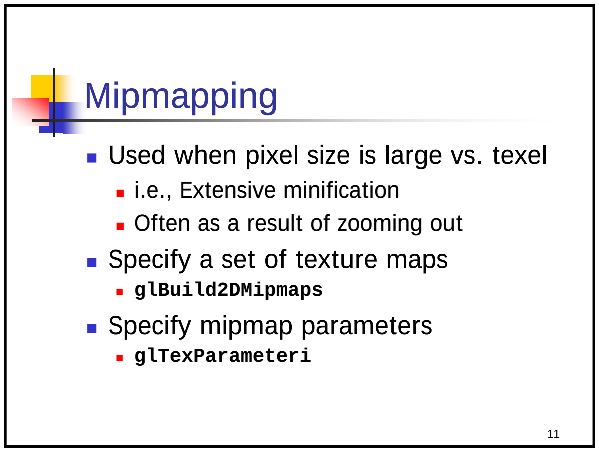
Positioning the Texture (2)

- Map the (s,t) point to a 3-D coordinate
  - Specify a vertex
- Observations
  - Need not use all of (s,t) range
  - Surface shape doesn’t have to match texture patch
    - Linear interpolated distortions occur



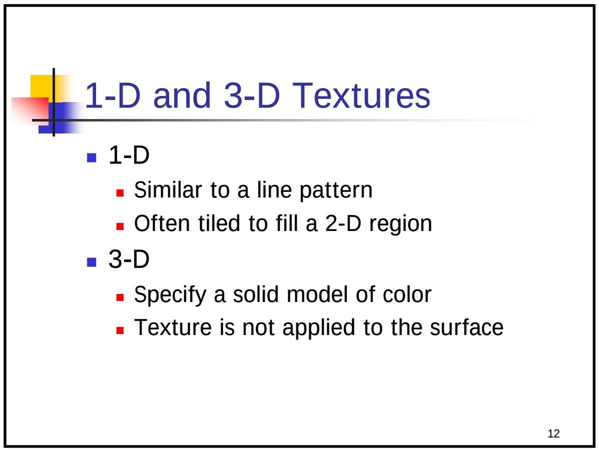
Miscellaneous Texture Options

- Automated texture positioning (Ex. 9-8)
  - `glTexGen*(coord, pname, value)`
  - As if projected on the surface
    - Texture locked to object – e.g., relief map
  - From a fixed or observer position
    - Texture locked in space – emphasize object motion?
- Correcting for perspective projections
  - `glHint(GL_PERSPECTIVE_CORRECTION, GL_NICEST); // or GL_NICEST or GL_DONT_CARE`



Mipmapping

- Used when pixel size is large vs. texel
  - i.e., Extensive minification
  - Often as a result of zooming out
- Specify a set of texture maps
  - `glBuild2DMipmaps`
- Specify mipmap parameters
  - `glTexParameteri`



1-D and 3-D Textures

- 1-D
  - Similar to a line pattern
  - Often tiled to fill a 2-D region
- 3-D
  - Specify a solid model of color
  - Texture is not applied to the surface