CPSC/ECE 4780/6780

General-Purpose Computation on Graphical Processing Units (GPGPU)

Lecture 20: (OpenGL) Textures

Recap of Last Lecture

- What is Bitmap?
- How to draw character in Bitmap?
- What is image?
- What OpenGL commands are available to manipulate image data?

Why Do We Need Textures?

- Two fundamental ways to display objects on the screen, each has its limitations:
  - Pixels: shows great details but lack 3D properties
  - Geometric objects: not quick enough to model many natural phenomena
- Texture mapping combines the best features of each approach
  - Provides great complexity
  - No overhead of building large geometric models

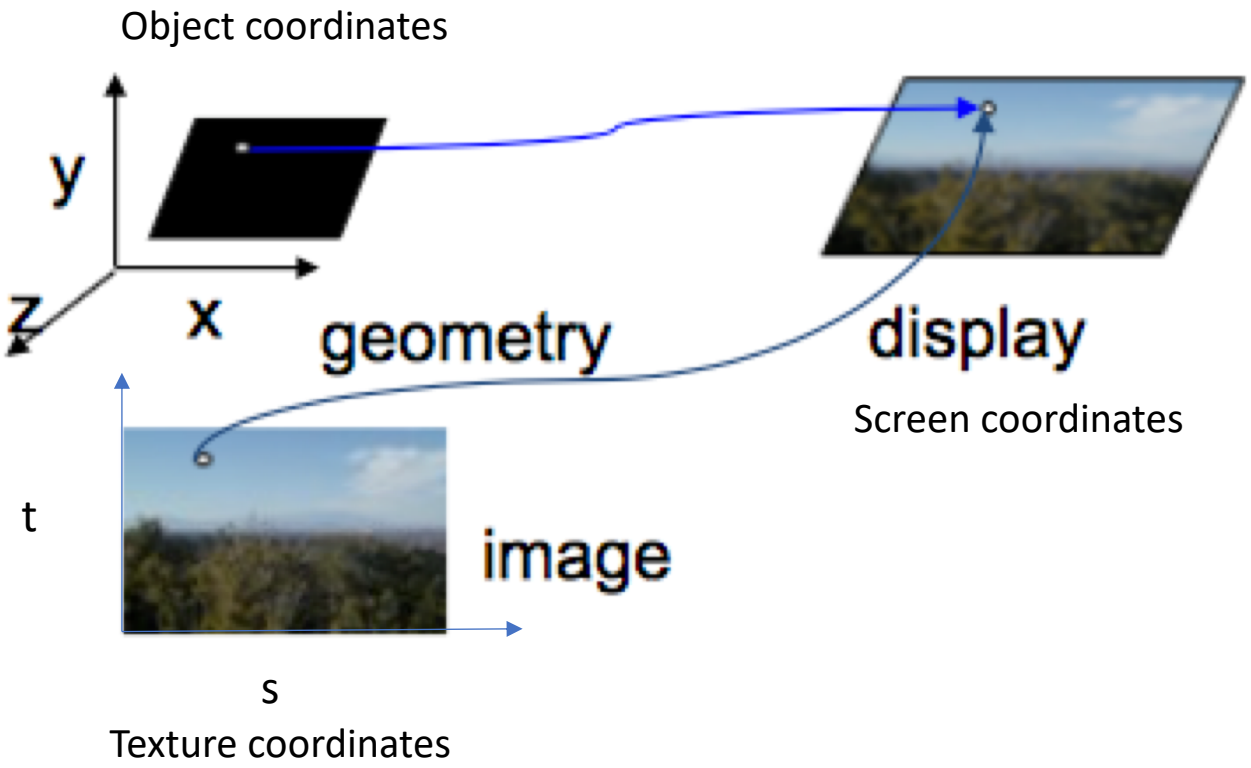
Textures and Texels

- Textures are patterns
  - Regular patterns, e.g., stripes, checkerboards
  - Complex patterns, e.g., natural materials
- Textures can be one, two, three, or four dimensional
  - 2D is of most common interests
- A texture has **texels** (texture elements), as a display has pixels
  - Texture is T(s,t), and s and t are **texture coordinates**
- Texture map associates a texel with each point on a geometric object

Texture Mapping

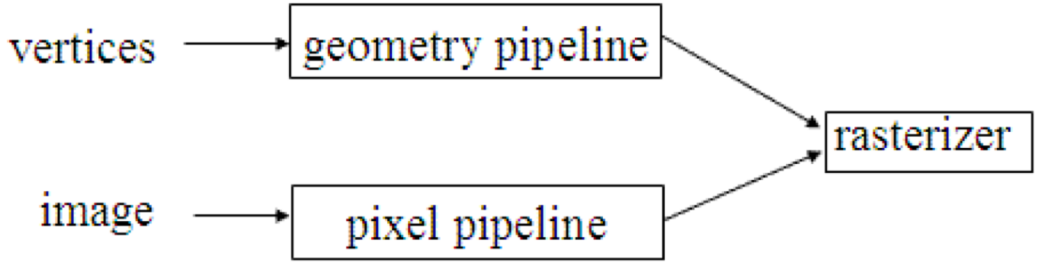
- A small area of the texture pattern maps to the area of the geometric surface, corresponding to a pixel in the final image

s = f(x, y, z)
t = g(x, y, z)



Textures in OpenGL

- Images and geometry flow through separate pipelines that join during rasterization and fragment processing



- Three steps to applying a texture
  - Specify the texture
  - Assign texture coordinates to vertices
    - Mapping function is set in application
  - Specify texture parameters
    - Wrapping, filtering

Specify the Texture

- Define a two-dimensional texture by:
  - void **glTexImage2D**(GLenum target, GLint level,
    GLint internalFormat, GLsizei width, GLsizei height,
    GLint border, GLenum format, GLenum type, const GLvoid *pixels);
    - target: type of texture, e.g., GL_TEXTURE_2D
    - level: used for mipmapping
    - internalFormat: elements per texel, e.g., GL_R3_G3_B2, GL_RGBA8, GL_RGB
    - width, height: width and height of texels in pixels
    - border: width of the border (used for smoothing)
    - format and type: describe texels
    - pixels: pointer to texel array

Converting a Texture Image

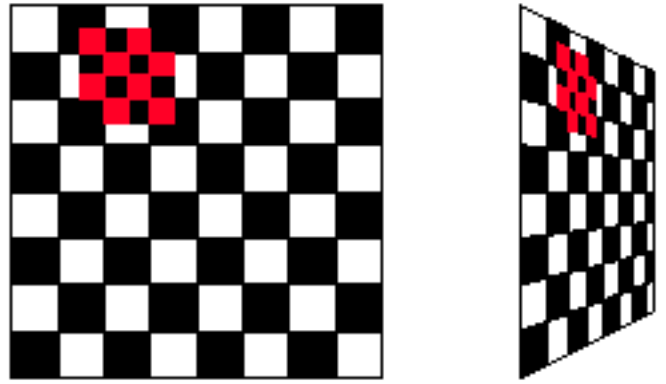
- OpenGL requires texture dimensions to be a powers of 2
- If dimensions of image are not powers of 2
  - int **gluScaleImage**(GLenum format, GLint widthin, GLint heightin, GLenum typein, const void *datain, GLint widthout, GLint heightout, GLenum typeout, void *dataout);
    - datain: source image
    - dataout: destination image
  - Image is interpolated and filtered during scaling

Copying a Texture Image

- The framebuffer itself can be used as source for texture data
  - void **glCopyTexImage2D**(GLenum target, GLint level, GLint internalFormat, GLint x, GLint y, GLsizei width, GLsizei height, GLint border);
    - target: set to GL_TEXTURE_2D
    - x, y: specify the lower-left corner coordinates for a screen-aligned pixel rectangle
    - width, height: width and height of the pixel rectangle

Replacing Part of a Texture Image

- Creating a texture is computationally expensive
- OpenGL allows us to replace all or part of a texture image with new information
  - void **glTexSubImage2D**(GLenum target, GLint level, GLint xoffset, GLint yoffset, GLsizei width, GLsizei height, GLenum format, GLenum type, const GLvoid *pixels);
    - target: set to GL_TEXTURE_2D
    - width, height: width or height of subimage
    - xoffset, yoffset: specify the texel offset in the x and y directions



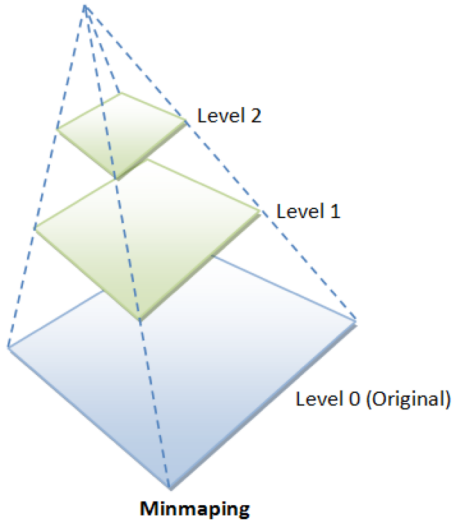
One-Dimensional Textures

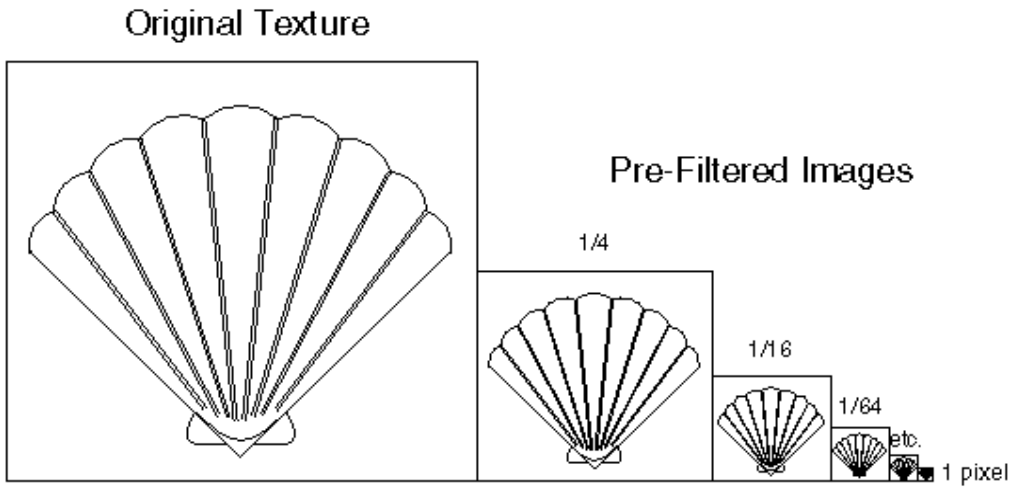
- Behaves like a two-dimensional one with height = 1, and without borders along the top and bottom
- All the two-dimensional texture and subtexture definition routines have corresponding one-dimensional routines
  - void **glTexImage1D**(GLenum target, GLint level, GLint internalFormat, GLsizei width, GLint border, GLenum format, GLenum type, const GLvoid *pixels);
  - void **glTexSubImage1D**(GLenum target, GLint level, GLint xoffset, GLsizei width, GLenum format, GLenum type, const GLvoid *pixels);
  - void **glCopyTexImage1D**(GLenum target, GLint level,
    GLint internalFormat, GLint x, GLint y,
    GLsizei width, GLint border);
  - void **glCopyTexSubImage1D**(GLenum target, GLint level, GLint xoffset,
    GLint x, GLint y, GLsizei width);

Mipmapped Textures

- Mipmapping allows for prefiltered texture maps of decreasing resolutions
  - Lessens interpolation errors for smaller textured objects
  - Requires extra computation and texture storage area
- Declare mipmap level during texture definition
  - void **glTexImage2D**(GLenum target, GLint level, …)
- GLU mipmap builder routines construct all the textures from a given image
  - int **gluBuild1DMipmaps**(GLenum target, GLint components, GLint width, GLenum format, GLenum type, void *data);
    int **gluBuild2DMipmaps**(GLenum target, GLint components, GLint width, GLint height, GLenum format, GLenum type, void *data);

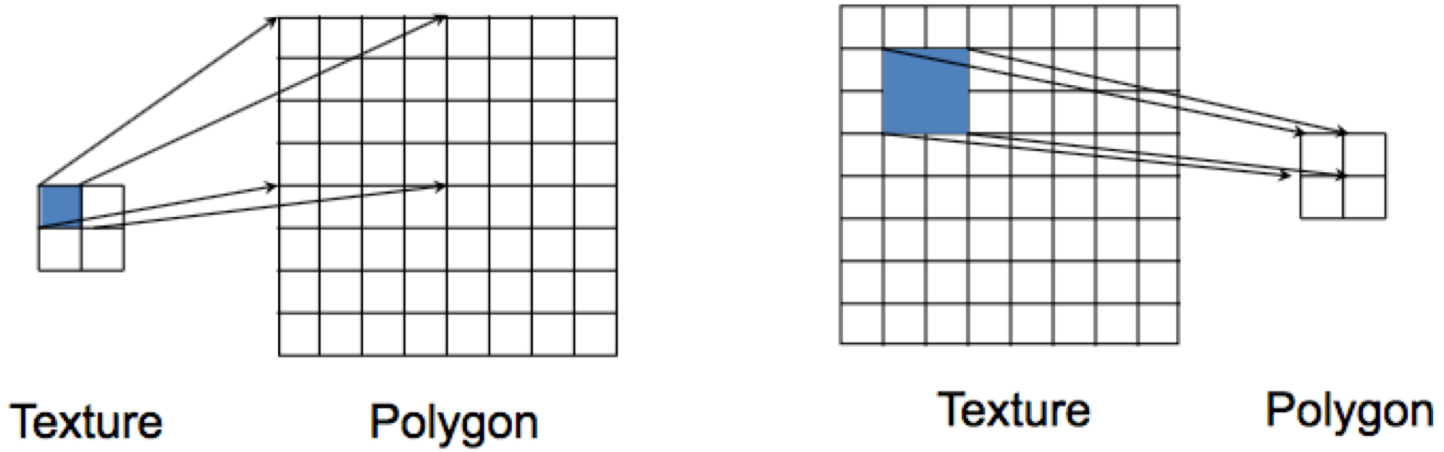
When using mipmapping, OpenGL automatically determines which texture map to use based on the size (in pixels) of the object being mapped.





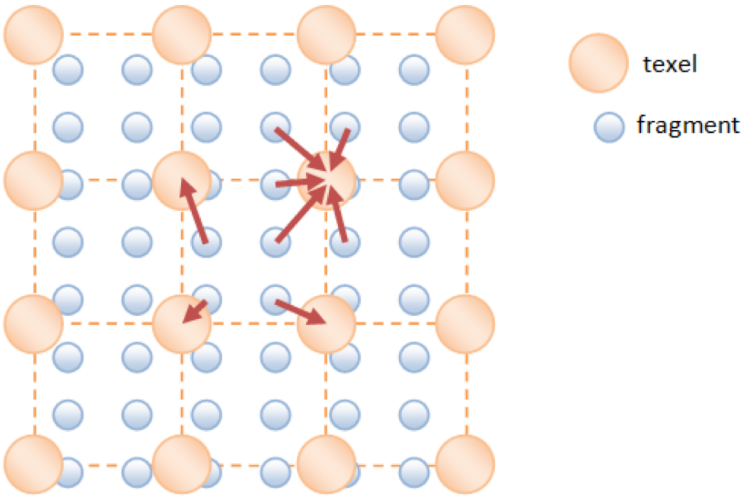
Magnification and Minification

- Magnification:
  - Texel larger than pixel, more than one pixel can cover a texel
- Minification:
  - Texel smaller than pixel, more than one texel can cover a pixel

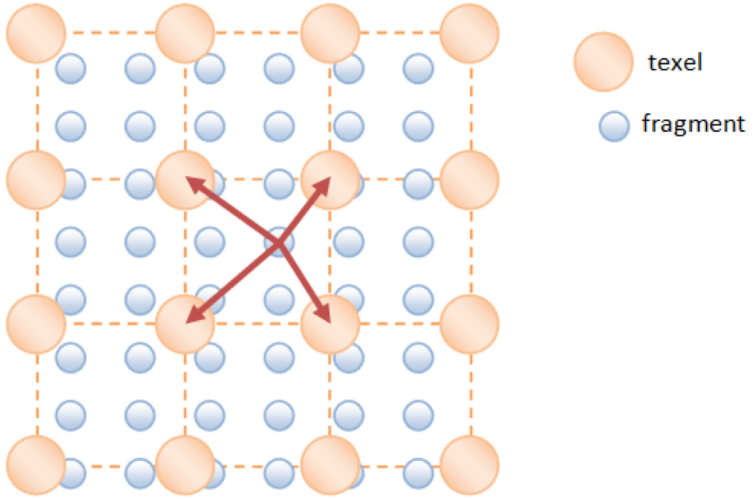


Magnification

- Two commonly used methods:
  - Nearest point filtering: the texture color-value of the fragment is taken from the nearest texel
  - Bilinear interpolation: the texture color-vlaue of the fragment is formed via bilinear interpolation of the four nearest texels



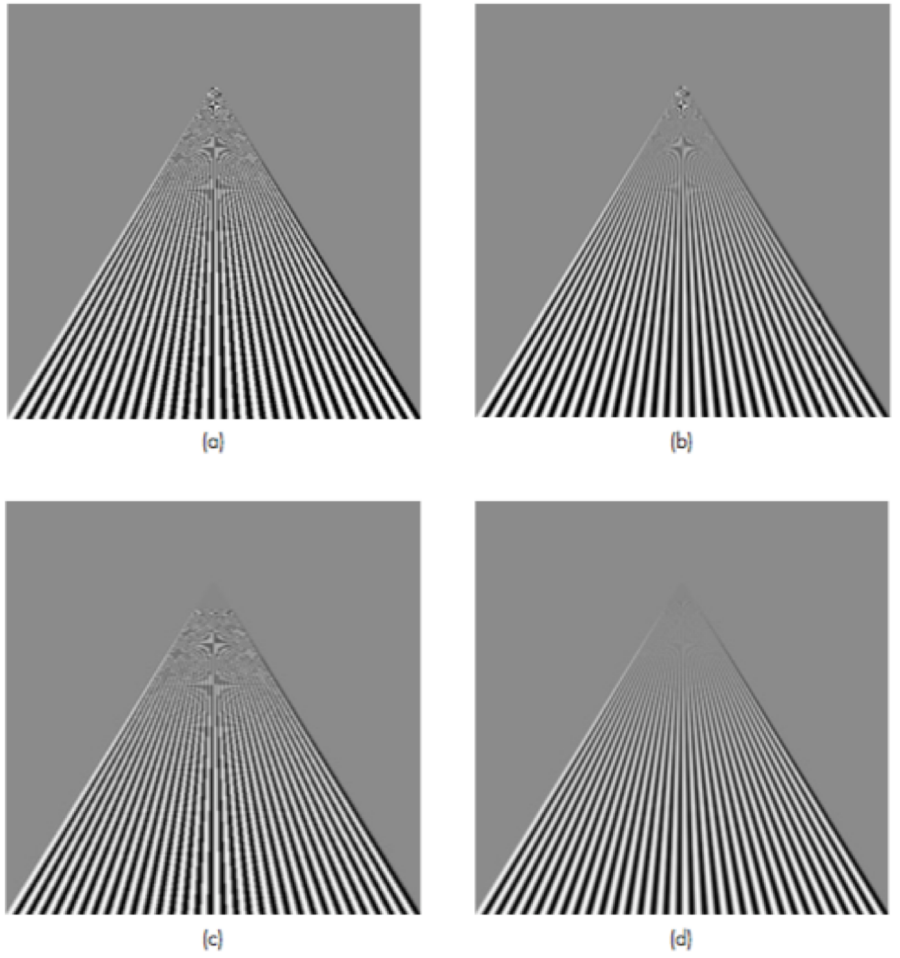
Magnification – Nearest Point Sampling



Magnification – Bilinear Interpolation

Filtering a Texture Image

- Filter modes determined by
  - **glTexParameteri**(target, type, mode);
    - target: GL_TEXTURE_2D or GL_TEXTURE_1D
    - type: GL_TEXTURE_MAG_FILTER or GL_TEXTURE_MIN_FILTER
    - mode: filtering method: GL_NEAREST, GL_LINEAR, GL_NEAREST_MIPMAP_NEAREST, GL_NEAREST_MIPMAP_LINEAR, GL_LINEAR_MIPMAP_NEAREST, or GL_LINEAR_MIPMAP_LINEAR



Texture mapping to a quadrilateral with (a) point sampling, (b) linear filtering, (c) mipmapping point sampling, and (d) mipmapping linear filtering

Texture Objects

- Texture is part of the OpenGL state
- OpenGL have texture objects to store texture data and makes it readily available
  - One image per texture object
  - Texture memory can hold multiple texture objects
  - Fastest way to apply textures
- For a single texture:
  - Glunit mytex[1];
  - glGenTextures(1, mytex);
  - glBindTexture(GL_TEXTURE_2D, mytex[0]);

Texture Functions

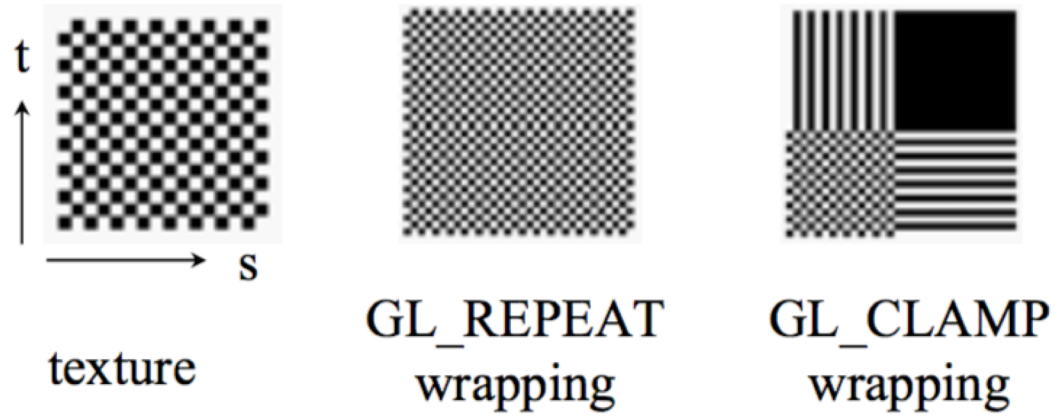
- Controls how texture is applied
  - void **glTexEnv**{if}(GLenum target, GLenum pname, TYPEparam);
    void **glTexEnv**{if}**v**(GLenum target, GLenum pname, TYPE *param);
    - target: GL_TEXTURE_ENV
    - pname: GL_TEXTURE_ENV_MODE, or GL_TEXTURE_ENV_COLOR
    - param:
      - If pname is GL_TEXTURE_ENV_MODE, param can be GL_DECAL, GL_REPLACE, GL_MODULATE, or GL_BLEND, to specify how texture values are to be combined with the color values of the fragment being processed
      - If pname is GL_TEXTURE_ENV_COLOR, param is an array of four floating-point values representing R, G, B, and A components

Assigning Texture Coordinates

- The texture coordinates determine which texel in the texture map is assigned to a particular vertex on the screen
- Texture coordinates can comprise one, two, three, or four coordinates, usually referred to as the s, t, r, and q coordinates
- glTexCoord*() specifies texture coordinates:
  - void **glTexCoord**{1234}{sifd}(TYPEcoords);
    void **glTexCoord**{1234}{sifd}**v**(TYPE *coords);

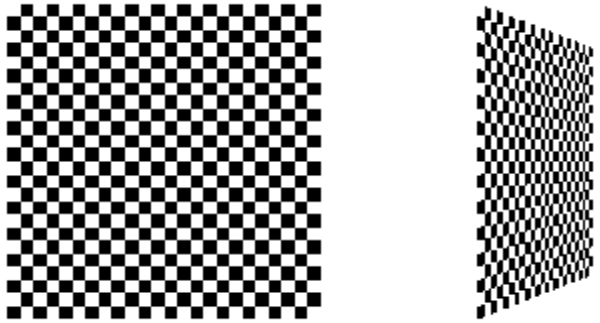
Repeating and Clamping Textures

- You can assign texture coordinates outside the range [0,1] and have them either clamp or repeat in the texture map
  - During repeating, the integer part of texture coordinates is ignored, and copies of the texture map tile the surface
  - Any values greater than 1.0 are set to 1.0, and any values less than 0.0 are set to 0.0



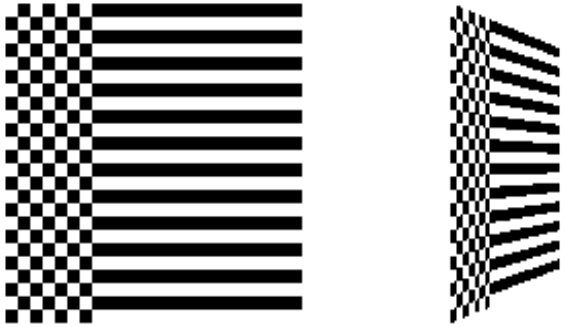
Repeating and Clamping Textures (Cont.)

- Wrap modes determined by:
  - **glTexParameteri**(target, type, mode);
    - target: GL_TEXTURE_2D or GL_TEXTURE_1D
    - type: GL_TEXTURE_WRAP_S, GL_TEXTURE_WRAP_T
    - mode: wrapping method: GL_CLAMP_TO_EDGE, GL_MIRRORED_REPEAT, or GL_REPEAT



The texture is repeated in both the s and t directions

The texture is clamped in both the s and t directions



The texture is clamped in one direction and repeated in the other

Automatic Texture-Coordinate Generation

- OpenGL can generate texture coordinates automatically
  - void **glTexGen**{ifd}(GLenum coord, GLenum pname, TYPEparam);
    void **glTexGen**{ifd}**v**(GLenum coord, GLenum pname, TYPE *param);
    - coord: GL_S, GL_T, GL_R, or GL_Q
    - pname: GL_TEXTURE_GEN_MODE, GL_OBJECT_PLANE, or GL_EYE_PLANE
    - param: a pointer to an array of values such as GL_OBJECT_LINEAR, GL_EYE_LINEAR, GL_SPHERE_MAP