Texture Mapping in OpenGL

19 March 2007
CMPT370
Dr. Sean Ho
Trinity Western University

TRINITY WESTERN UNIVERSITY

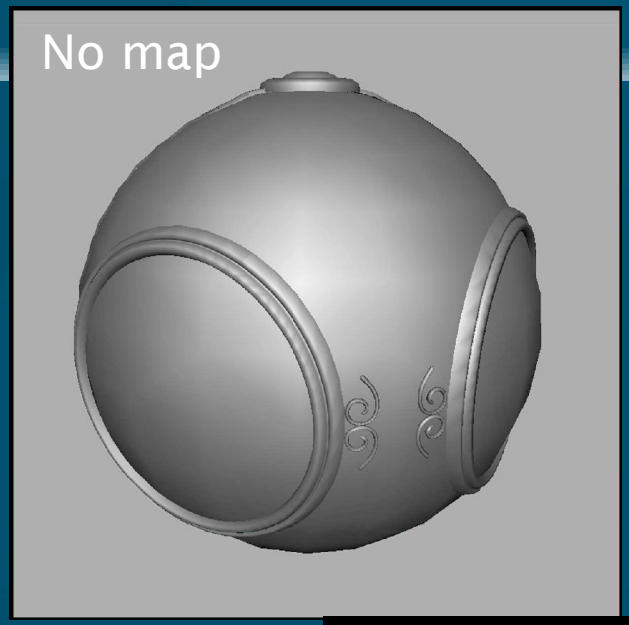
Review last time

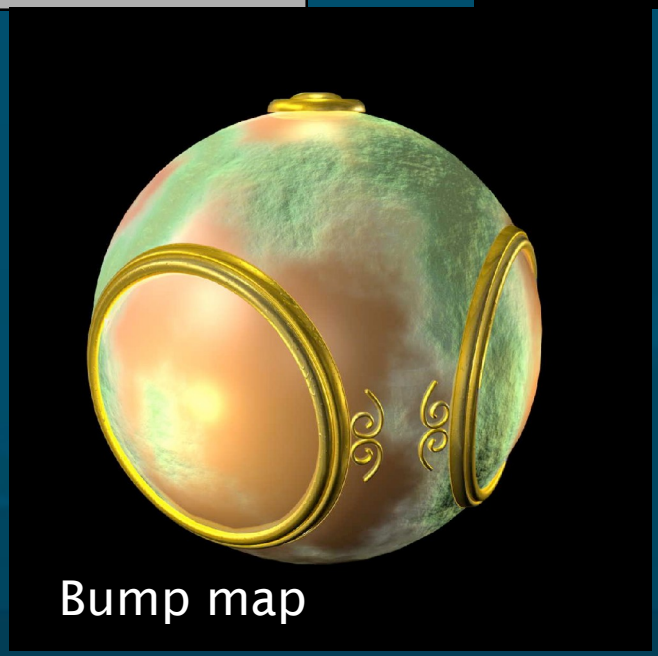
- Shading polygons
  - Flat shading
  - Gouraud shading
  - Phong shading
- Texture mapping
  - Coordinate transforms
  - Cylinder, sphere, cube maps
  - Bump mapping
  - Environment mapping

What's on for today

- Bump mapping theory
- Creating a texture in OpenGL
  - Texture objects: glBindTexture()
  - Loading image data: glTexImage2D()
    - Using the framebuffer as a texture
- Applying a texture in OpenGL
  - Blending modes: glTexEnvf()
  - Texture coordinates: glTexCoord2f()
    - Auto-generated texcoords: glTexGen()
    - Spherical environmental mapping
  - Mip-maps

Texture/bump/environment maps



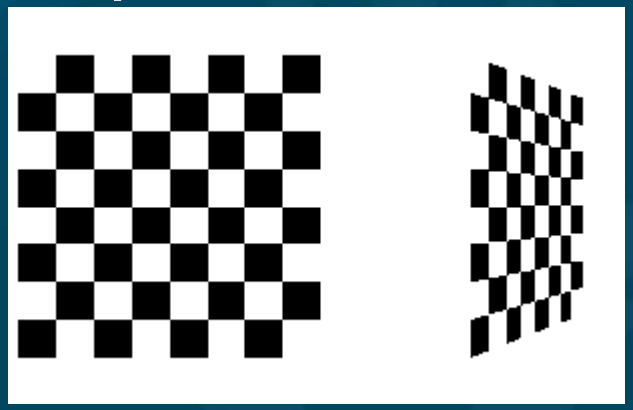


Bump mapping

- Parameterized surface:
  - $p(u,v) = (x(u,v), y(u,v), z(u,v))$
  - Tangent vectors: $p_u = \partial p/\partial u$, $p_v = \partial p/\partial v$
  - Normal vector: $n = p_u \times p_v$
- Perturbed surface: $p'(u,v) = p(u,v) + d(u,v)\, n(u,v)$
  - $d(u,v)$ is the displacement function/map
- Perturbed normal: $n' = p'_u \times p'_v$
  - $n' \approx (\partial d/\partial u)(n \times p_v) + (\partial d/\partial v)(n \times p_u)$

Texture mapping in OpenGL

- Bump mapping / environment mapping are not provided in OpenGL
  - Can be done with fragment programs (GLSL)
- Using texture mapping in OpenGL:
  - Create texture and bind to object
  - Select how texture will affect each pixel
  - Enable texture mapping
  - Draw object, specifying texture coordinates
- See Redbook examples, checker.c



Creating a texture

- These steps should be done during initialization, not on every display refresh

- Read in an image: 3D array (rows, cols, RGBA)
  - Programmatically generate (checker.c), or
  - Read from file (Fl_JPEG_Image->data())
- Bind new texture object: glBindTexture()
- Specify parameters: wrapping, filtering
- Load image data to texture: glTexImage2D()

Texture objects (OpenGL 1.1)

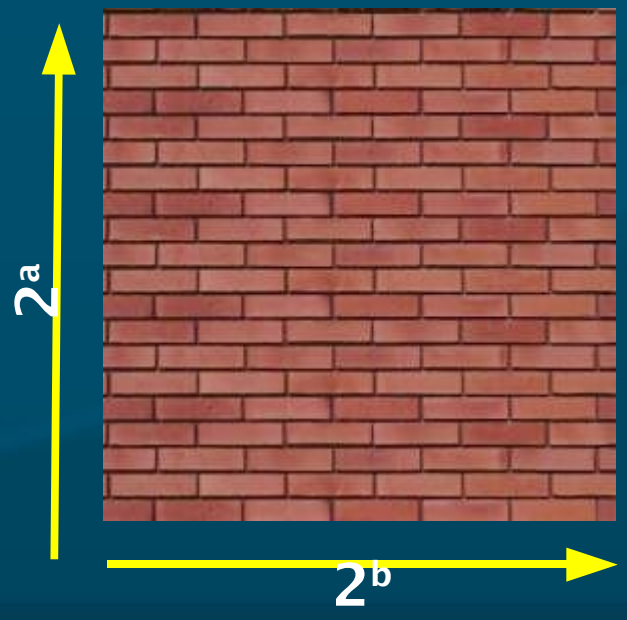
- Akin to display lists, but for textures
- Allows us to reuse textures, bind to objects
  - Request a new texture object id
    - glGenTextures(1, &texName);
  - Can also request several texture object ids
  - Bind this new texture object
    - glBindTexture(GL_TEXTURE_2D, texName);
- All subsequent texture commands are stored in this texture object
- Use glBindTexture() to switch texture objects

Loading image data to a texture

- glTexImage2D(GL_TEXTURE_2D, *level*, *intFmt*, *width*, *height*, 0, *format*, *type*, *pixels*)
  - level: mip-mapping level, usually 0
  - intFmt: GL_RGB, GL_RGBA, etc.
  - width, height: must be power of 2, ≥64
  - format, type: describe incoming pixels:
    - e.g., GL_RGB, GL_UNSIGNED_BYTE
    - Affected by glPixelStore(), similar to glDrawPixels()
  - pixels: pointer to the actual image data

Texture size must be power of 2

- OpenGL requires the width and height of textures to be powers of 2 (need not be square)
- GLU provides a function to scale:
  - gluScaleImage(fmtIn, wIn, hIn, typeIn, *pixelsIn, wOut, hOut, typeOut, *pixelsOut)



Using the framebuffer as a texture

- Instead of loading a JPEG file for a texture, you can use the framebuffer itself:
  - glCopyTexImage2D(GL_TEXTURE_2D, *level*, *intFmt*, *x*, *y*, *w*, *h*, *border*)
    - Copies a rectangle from the framebuffer, starting at (x,y) with size (w,h)
    - level, intFmt, border just as in glTexImage2D
- Can use to do cheap reflections:
  - Flip model-view matrix and render
  - Texture-map framebuffer onto object

Applying the texture

- These steps are done during display() refresh
- Enable texture-mapping
  - glEnable(GL_TEXTURE_2D);
- Set blending mode
  - glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_DECAL);
- Bind the preloaded texture
  - glBindTexture(GL_TEXTURE_2D, texName);
- Specify texture coordinates with every vertex
  - glTexCoord2f(0.0, 0.0); glVertex3f(1.0, 2.5, -1.0);

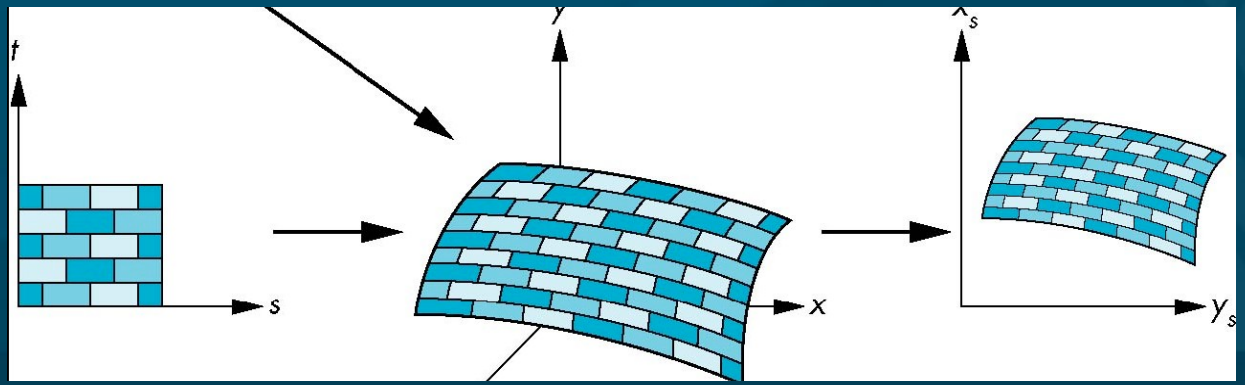
Blending modes

- glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_DECAL);

- Last param is the blending mode:
  - How the texture colour is combined with the shaded colour of the fragment (e.g., Gouraud- or flat-shaded)
  - GL_DECAL: pastes texture on top
  - GL_MODULATE: multiplies colours
  - GL_BLEND: uses texture to determine amount of blend between shaded colour and a fixed blend colour

Texture coordinates

- The rectangular texture is parameterized by (s,t) in the range (0,1)
- Specify texture coordinates with each vertex
  - glTexCoord2f(0.5, 0.7)
  - Part of the OpenGL state for the vertex, just like colour / material properties
  - Texcoords are interpolated just like shades



Parameters for texture mapping

- Wrapping: texcoords outside (0,1)
  - Repeat (tile) or clamp (only one copy)
  - In either s or t directions in texture
    - glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
- Filtering:
  - Magnification (MAG) and minimization (MIN)
    - glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
  - Nearest-neighbor or GL_LINEAR interpolation

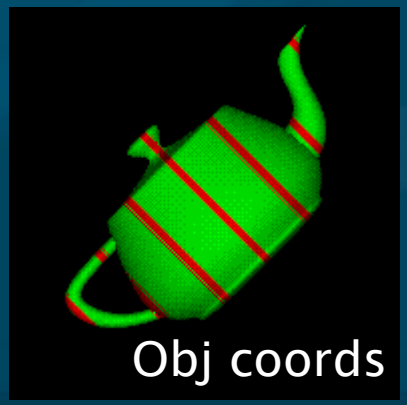
Automatic texcoord generation

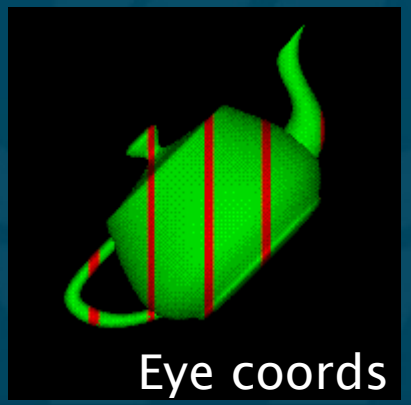
- glEnable(GL_TEXTURE_GEN_S);
- glTexGeni(coord, GL_TEXTURE_GEN_MODE, mode)

- coord: GL_S or GL_T
- If mode is GL_OBJECT_LINEAR:
  - Texture is fixed to object and reference plane
  - Specify reference plane with:
    - glTexGenfv(coord, GL_OBJECT_PLANE, $\{p_1, p_2, p_3, p_4\}$)
  - Generated texcoord is distance from vertex to plane: $p_1x_0 + p_2y_0 + p_3z_0 + p_4w_0$
- Teapot example: slanted plane = {1., 1., 1., 0.}

Object vs. eye coordinates

- If mode is GL_OBJECT_LINEAR, generated texcoords are in the model coordinate system
- If mode is GL_EYE_LINEAR, generated texcoords are in the eye (camera) coordinate system
  - Object appears to "swim" in the texture
  - Reference plane is specified with
    - glTexGenfv(coord, GL_OBJECT_PLANE, $\{p_1, p_2, p_3, p_4\}$)

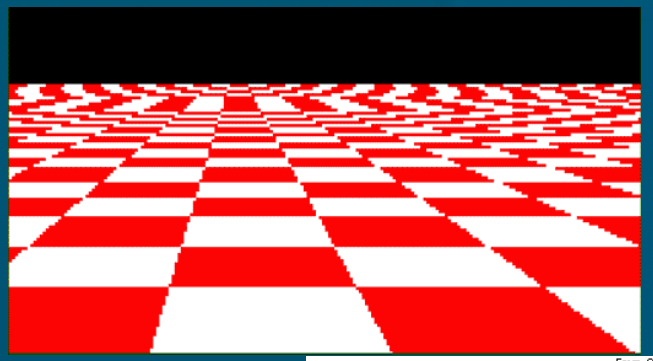




Spherical environment mapping

- Photograph a large silvered ball, or use a fisheye wide-angle lens
- Use automatic spherical texcoords for both s and t:
  - glTexGeni(GL_S, GL_TEXTURE_GEN_MODE, GL_SPHERE_MAP)
- Assumes environment is far away (e.g., small object in large room)

Mip-maps



- Aliasing (jaggies) occurs when textures become very small on-screen
- Pre-calculate low-res versions of the texture: levels of detail (LoD)
  - Use gluBuild2DMipmaps() instead of glTexImage2D()