Lab 08 - OpenGL Texturing (chapter 18)

I. Specifying a texture pattern

A. 1D texture

i. Enable/Disable w/ GL_TEXTURE1D

ii. glTexImage1D(GL_TEXTURE_1D, 0, GL_RGBA, nTexColors, 0, dataFormat, datatype, lineTexArray);

- GL_TEXTURE_1D → texture type
- 0 → level ID of MIP
- GL_RGBA → color type in Array
- nTexColors → size of array
- 0 → TRUE/FALSE on border
- dataFormat → color storage format
- datatype → color data type
- lineTexArray → array

B. 2D Texture

i. Enable/Disable w/ GL_TEXTURE_2D

ii. glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, width, height, 0, dataformat, datatype, array)

same except wxh instead of n.

C. 3D texture

i. Enable/Disable w/ GL_TEXTURE_3D

ii. glTexImage3D (GL_TEXTURE_3D, 0, GL_RGBA, width, height, depth, 0, dataformat, datatype, array)

same except wxhxd.

D. Minification/Magnification Filtering

i. In class to select color of pixel we looked at either nearest texel or weighted average of texels. In GL you can separate filter scheme when enlarging texture or shrinking.

ii. glTexParameteri(texture, GL_TEXTURE_MAG_FILTER / GL_TEXTURE_MIN_FILTER, GL_NEAREST / GL_LINEAR)

E. Specifying texture coordinates for vertex

i. Almost same as vertex normals.

ii. With immediates

glTexCoord*(*) ex. glTexCoord1f(0.5) or glTexCoord2f(0.25, 0.33)

iii. With basic VBO (Different for VAO)

glEnableClientState(GL_TEXTURE_COORD_ARRAY);

glTexCoordPointer(n, datatype, offset, array pointer, ie start of tex in VBO)

F. Texture ~~mapping~~ options

i. Mapping - glTexEnvi(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, method)

method 1 of: GL_COMBINE, GL_REPLACE, GL_MODULATE, GL_DECAL, GL_BLEND

ii. Wrapping - glTexParameter*(texture, coordinate, option)

coordinate: GL_TEXTURE_WRAP_S, GL_TEXTURE_WRAP_T, GL_TEXTURE_WRAP_R

option: GL_REPEAT, GL_CLAMP, ~~etc~~

iiii. Many, many more options and specifications!

iii. Border color - glTexParameterfv(texture, GL_TEXTURE_BORDER_COLOR, color)

II. Naming Texture Patterns

A. Generating names - glGenTextures(n, memory) n → number of names; memory → place to store

B. Binding (making texture active) - glBindTexture(texture, name) eg. glBindTexture(GL_TEXTURE_2D, 3);

C. After binding change mapping or params.

D. Deleting textures - glDeleteTextures(n, memory of names)

E. Validity checking - glIsTexture(name)

III. Mipmapping.

A. Creating - gluBuild2DMipmaps(texture, GL_RGBA, width, height, ~~GL_RGBA~~ data format, data type, array)
Can 1D or 3D

or

gluBuild2DMipmapLevels(texture, GL_RGBA, width, height, data format, data type, 0, 0 current texture level
↑ or 1D or 3D
minlevel, maxlevel, array);

B. Filtering - additional options on min filter: GL_NEAREST_MIPMAP_NEAREST

GL_LINEAR_MIPMAP_NEAREST

GL_NEAREST_MIPMAP_LINEAR

GL_LINEAR_MIPMAP_LINEAR