Texture Mapping in Practice

Dinesh K. Pai

Textbook Appendix A4, Chapter 15

Some slides courtesy of M. Kim, KAIST

Today

- Announcements
  - Assignment 3: signup at [http://doodle.com/9g5hndwk2denfx74](http://doodle.com/9g5hndwk2denfx74)
  - Don’t forget to sign up for face-to-face grading in time! This is necessary to get a mark for the assignment. Sign up will close on Sunday night.
  - Assignment part 2 discussion
  - Last call for unclaimed umbrella, left during office hour

- Texture mapping, continued
Texture mapping

- In basic texturing, we simply ‘glue’ part of an image onto a triangle by specifying texture coordinates at the three vertices.

Bunch of OpenGL/WebGL functions to load a texture and set various parameters (lin/const, mipmap, wrapping rules).
- A uniform variable is used to point to the desired texture unit.
Texture mapping

- Varying variables are used to store texture coordinates.
- In this simplest incarnation, we just fetch r,g,b values from the texture and send them directly to the frame buffer.

Texture mapping

- Alternatively, the texture data could be interpreted as, say, the diffuse material color of the surface point, which would then be followed by the diffuse material computation described earlier.
Steps for Texture Mapping

1. Create a *texture object* and load texels into it
2. Include *texture coordinates* with your vertices
3. Associate a *texture sampler* with each texture map used in shader
4. Retrieve texel values

(Reference: Red Book)
Texture coordinates

3D Earth

2D Earth (longitudes)

Aside: texture is named

Longitude

Object

Texture

-180

180

\[ u \]

\[ \overrightarrow{u} \text{ (on } S) \]

\[ u = \tan^{-1} \left( \frac{p_y}{p_x} \right) \]

Can't do this for most real objects
Piecewise linear mapping

\[ P_1, u_1, \quad a \text{ new per-vertex attribute} \]

linear interpolation between vertex values approximating the mapping

\[ u_1, u_2 \]

Same machinery as previous classes \( L_{21}, L_{22} \)

\( u, v \) are called texture coordinates.

One more refinement: want to abstract out the fact that texture images have finite extent.

\[ \text{image} \]

\[ 0 \quad u \rightarrow \quad 1 \]

\( u \) between 0 and 1 within image, outside this, coordinates can repeat, or be clamped.
Second basic example:
Texture mapping a square

- See Appendix A.4

- See Nehe texture tutor
Texture mapping in OpenGL (WebGL is very similar)

- initGLState()

...  
glActiveTexture(GL_TEXTURE0);  
glGenTextures(1, &h_texture);  
glBindTexture(GL_TEXTURE_2D, h_texture);  
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP);  
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP);  
int twidth, theight;  
packed_pixel_t * pixdata = ppmread("reachup.ppm", &twidth, &theight);  
assert(pixdata);  
free(pixdata);

Texture mapping

- initShaders()

h_texUnit0 = safe_glGetUniformLocation(h_program, "texUnit0");  
h_aTexCoord = safe_glGetAttribLocation(h_program, "aTexCoord");

- display()

safe_glUniform1i(h_texUnit0, 0);

Texturing location (0,0) \(\rightarrow\) lower left, (1,1) \(\rightarrow\) upper right

Glfloat sqTex[12] =
{  
0, 0,  
1, 1,  
1, 0,  
0, 0,  
0, 1,  
1, 1
};
Texture mapping

- Vertex shader, just “pass through”

```glsl
#version 330
uniform float uVertexScale;
uniform mat4 uProjMatrix;
uniform mat4 uModelViewMatrix;
in vec2 aVertex;
in vec2 aTexCoord;
in vec3 aColor;
out vec3 vColor;
out vec2 vTexCoord;

void main()
{
    gl_Position = vec4(uProjMatrix * uModelViewMatrix * aVertex);
vColor = aColor;
vTexCoord = aTexCoord;
}
```
Texture mapping

- Fragment shader changes

```
#version 330

uniform sampler2D texUnit0;
in vec2 vTexCoord;
out vec4 fragColor;

void main() {
    vec4 texColor0 = texture2D(texUnit0, vTexCoord);
    fragColor = texColor0;
}
```

Texture Mapping in Three.js

- Demo