CS4620 Fall 2011
HW3 - Textures

Due: Nov 8 2011

This assignment has two written problems and a programming problem.

1 Texture Mapping and Texture Coordinates

Consider a 2D triangle whose three vertices are at $(1,1)$, $(8,2)$, and $(9,3)$. The triangle is texture mapped with a texture T of resolution 512×512 pixels. The color of pixel (i,j) where i and j are integers is denoted $T(i,j)$. This color corresponds to point with texture coordinate

$$((i+0.5)/512, (j+0.5)/512).$$

In this problem, we want to find the color assigned to point $Q = (6,2)$. You can assume that the triangle is viewed using an orthographic projection, so no perspective correction is necessary when calculating texture coordinate.

1.1

Suppose the three vertices have texture coordinates $(0.3, 0.3)$, $(0.6, 0.45)$, and $(0.9, 0.9)$, respectively.

(a) What is the color assigned to Q using nearest neighbor sampling? If more than one point is equidistant from Q, write down all such points.

(b) What is the color assigned to Q using bilinear interpolation?

All answers should be expressed in terms of T, and all the arguments of T must be integers.

1.2

Suppose the three vertices have texture coordinates $(0.6, 0.25)$, $(1, 0.35)$, and $(1.4, 0.6)$, respectively. The texture mapping is set up so that it loops around in both u and v direction. That is, the color for point with texture coordinate (u,v) is the same as the point with texture coordinate $(u - \lfloor u \rfloor, v - \lfloor v \rfloor)$. Here, $\lfloor x \rfloor$ is the greater positive integer not exceeding x.

(a) What is the color assigned to Q using nearest neighbor sampling? If more than one point is equidistant from Q, write down all such points.

(b) What is the color assigned to Q using bilinear interpolation?

All answers should be expressed in terms of T, and all the arguments of T must be integers.

2 MIP-mapping

A rectangle in 2D with corners $(0, 0, 0)$, $(4, 0, 0)$, $(4, 2, 0)$, $(0, 2, 0)$ is textured mapped so that the vertices have texture coordinates $(0, 0)$, $(1, 0)$, $(1, 1)$, and $(1, 0)$, respectively. The texture has resolution 512×512 pixels.

A MIP-map is constructed on the texture. MIP-map level 0 corresponds to the original image and is denoted by the function $T_0(i, j)$. MIP-map level 1 corresponds to the downsampled version with 256×256 pixels and is denoted by the function $T_1(i, j)$. The same goes for MIP-map level 2, 3, and so on.

2.1

How many MIP-map levels are there?

2.2

Consider a viewing situation where the eye is looking squarely at the rectangle and Q is at the center of the rectangle. (Assume there is no distortion due to perspective.) The rectangle is at a distance from the eye so that a pixel maps to a square region of area 0.04 around the point Q.

What MIP-map levels should be used to texture the pixel? Evaluate the level assuming the level is determined by the maximum of the rates of change defined in the lecture.

2.3

Find the color assigned to Q in Problem 2.2 using trilinear interpolation. The answer should be expressed as a linear combination of $T_k(i, j)$s, where the arguments to the T_ks must be integers.

3 Texture Mapping in Shaders

In this problem, you will write shaders that make use of textures. The framework for writing the shaders is distributed in the course web site and is basically the one you worked with in PA2. However, we have augmented the framework to handle texture mapping as follows:

- We have added a new class `Texture`, which is in the `pipeline.misc` package.

- The class `FragmentProcessor` has a new field `texture`, which stores the texture being used by the fragment processor.

  On the other hand, the `VertexProcessor` class does not have a reference to a texture. So, texturing is only available in fragment shaders.

- Each `Vertex` now has a new field `t`, which stores the texture coordinate for the vertex.

  This means that, when you implement the `vertex`, method of the `VertexProcessor` class, you must set the `t` field of the vertex being passed in. This `t` value will be interpolated by the rasterizer and passed to the fragment shader. We have implemented perspective-correct texture coordinate interpolation. So, you don't have to worry about it.

- The field `t`, which is used to store the texture coordinate of each fragment, will be set by the rasterizer.

  This means that, in the fragment shaders you implement, you can use `t` to fetch color from the texture.

This problem has two subproblems you need to complete.

3.1 Bilinear Interpolation

The method

```
public void sample(Vector2f t, Color3f cOut)
```

of the `Texture` class should compute the color of the point with texture coordinate `t` using bilinear interpolation.

Note that you can use the method

```
public void sample(int ix, int iy, Color3f cOut)
```

to read the color of the (ix, iy)-texel of the texture. This function works similar to the T function in the first problem.

3.2 Per-Fragment Textured Blinn-Phong

Implement `TexturePhongVP` and `TexturedPhongFP` so that it shades each fragment according to the equation:

$$L = k_a + \sum_{j=1}^{n} (k_d T \max(0, \mathbf{n} \cdot \mathbf{l}) + k_s (\max(0, \mathbf{h} \cdot \mathbf{n}))^p) I_j$$

where

- L is the color of the pixel,
- n is the number of light sources,
- k_a is the ambient color,
- k_d is the diffuse color, which is the interpolated vertex color (`f.c`),
- k_s is the specular color,
- T is the color of the texture at the fragment,
- $\mathbf{n}$ is the normal vector,
- $\mathbf{l}$ is the light vector,
- $\mathbf{h}$ is the half vector between the light and the view vector,
- p is the material's shininess, and
- I_j is the intensity of the jth light.

The difference between the above equation and the standard Blinn-Phong lighting model is that we use the texture color T to modulate the diffuse component. Also, note that shading has to happen in the fragment shader, not the vertex shader.

After you are done with the above problems, remove all `.class` files, ZIP the whole `Pipeline2` directory, and submit the ZIP file to CMS.