COSC422 Advanced Computer Graphics

Programming Exercise 10

Non-Photorealistic Rendering

This programming exercise introduces basic techniques for 3-tone rendering and pencil shading of three-dimensional objects using only vertex and fragment shaders.

TeapotNPR.cpp:

Low polygon count models are always preferred for non-photorealistic rendering of objects. The program `TeapotNPR.cpp` loads a coarse polygonal model of a teapot consisting of 1452 triangles. The vertex shader, as usual, performs transformation and lighting, and the fragment shader outputs the interpolated colour values received from the vertex shader (Fig. 1.).

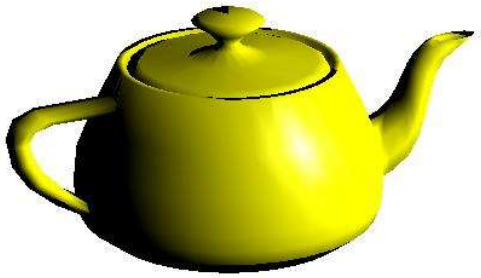

Fig. 1.

The fragment shader also receives the value of ***l.n*** in the variable "`diffTerm`". Modify the fragment shader to output colours according to the following rule:

l.n < 0 : output ambient colour (0.2, 0.2, 0.2, 1)

l.n ∈ [0, 0.7] : output mid-tone colour: (0.5, 0.5, 0, 1)

l.n > 0.7 : output material colour (1, 1, 0, 1)

The variable '`phongColor`' is no longer used, and may be removed from the shaders.

Use the two-pass rendering approach shown on Slide [6]-17 for drawing silhouette edges. The uniform variable location `flagLoc` in the application may be used to distinguish between the two passes in the fragment shader. During the second pass, the fragment shader should output only black colour. For the two-pass method to work, polygon culling must be enabled (un-comment the statement in the initialise() function). The output of the program is shown in Fig. 2.

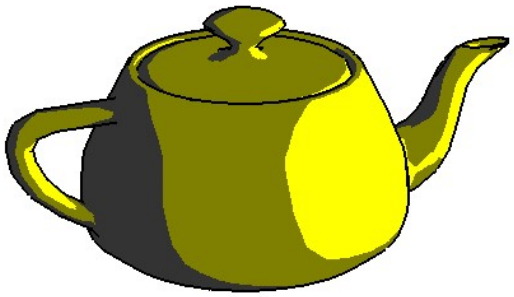

Fig. 2

A function "loadTextures()" that loads 3 pencil stroke textures is provided. Add a statement to call this function from within the initialise() function. Also, add the following declaration statement to the fragment shader:

```
uniform sampler2D pencilStroke[3];
```

The corresponding uniform variable locations and their values have already been defined in the application. Please note that the texture coordinates are not included as part of the model definition. We will generate the texture coordinates in the vertex shader.

The rendering command used for the model is "glDrawArrays()". The general format of this command is

```
glDrawArrays(GL_TRIANGLES, 0, numVerts);
```

where *numVerts* is the number of vertices that are output. A built-in input variable **`gl_VertexID`** available in the vertex shader will have a value between 0 and *numVerts*-1, representing the index of the current vertex. Use this variable to assign texture coordinates (0, 0), (0.5, 0), (0.25, 0.5) to every set of three consecutive vertices processed by the vertex shader. Pass the assigned texture coordinates as an 'out' variable's value to the fragment shader.

In the fragment shader, select the first texture (`pencilStroke[0]`) to get the output colour if $\boldsymbol{l.n} < 0$, second texture if $\boldsymbol{l.n} \in [0, 0.7]$, and the third texture if $\boldsymbol{l.n} > 0.7$. The output of the program should look similar to that shown in Fig. 3.

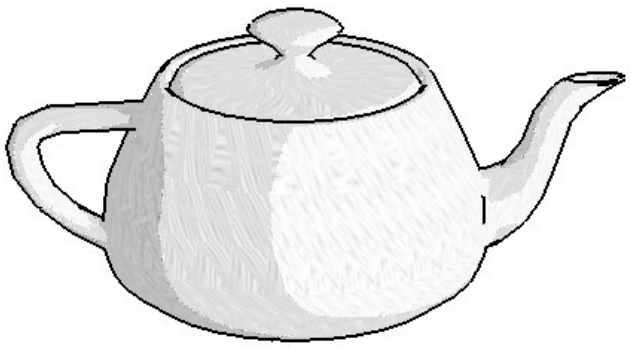

Fig. 3.

Use a weighted combination of two textures (multi-texturing) at points close to shade boundaries ($\boldsymbol{l.n} = 0$, $\boldsymbol{l.n} = 0.7$) to blend the textures as shown in Fig. 4.

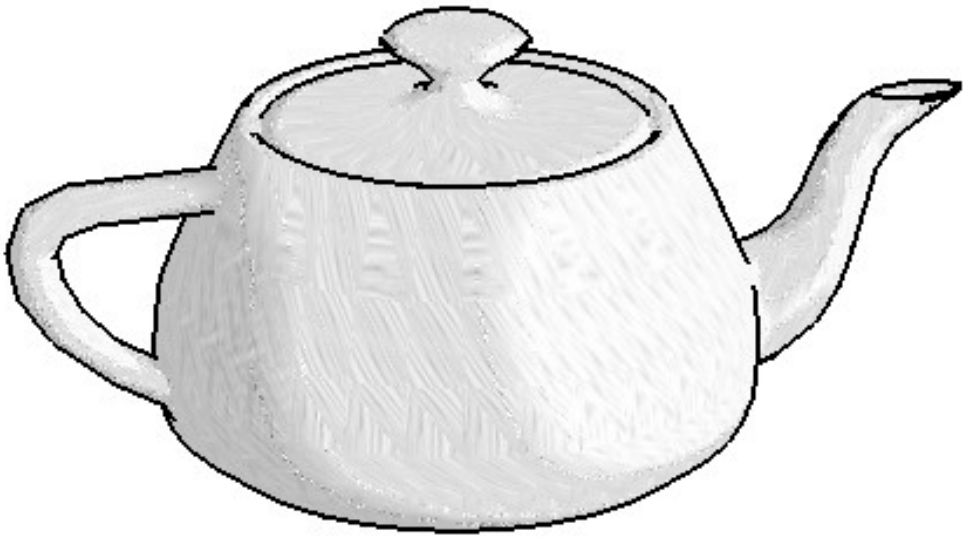

Fig. 4.

[6]: COSC422 Lecture slides, “Non-Photorealistic Rendering”.