COSC422 Advanced Computer Graphics
Programming Exercise 02

This exercise shows the application of the fragment shader for multi-texturing a cube object. The structure of the program is similar to that of TorusDraw.cpp (Exer01).

CubeDraw.cpp:

The program `CubeDraw.cpp` provides the code for displaying the mesh model of a cube. It has a structure similar to `TorusDraw.cpp`.

- The file `Cube.h` contains the array definitions specifying vertex coordinates, normal components, texture coordinates and polygon indices.

- The vertex shader (`Cube.vert`) implements a simple lighting model using only the diffuse component. It also outputs the texture coordinates and the diffuse lighting term (***n•l***) to the fragment shader.

- The fragment shader (`Cube.frag`) defines a uniform variable "tSampler1" of type `Sampler2D`. The function `texture()` returns a colour value obtained by sampling the texture using the input texture coordinates. The fragment shader outputs this colour for each fragment, thus generating the display of a texture mapped primitive.

- The program `CubeDraw.cpp` loads the texture "Brick.tga" (Fig. (i)) and assigns it to texture unit 0. Using the function `glUniformi()` it assigns the same value 0 to the variable "tSampler1" in the fragment shader. The program generates the output of a texture mapped cube (Fig. (k)).

Fig. (i)

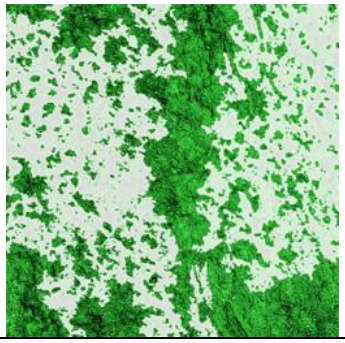

Fig. (j)

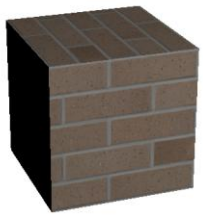

Fig. (k)

1. Modify the program `CubeDraw.cpp` to load a second texture “Moss.tga” (Fig. (j)). Select texture unit 1 for this texture, and assign the value 1 to a uniform variable “tSampler2” . Add this new uniform variable “tSampler2” in the fragment shader. Modify the shader’s output by combining the colour values obtained from the two textures. The result of multi-texturing should look similar to that given in Fig. (l).

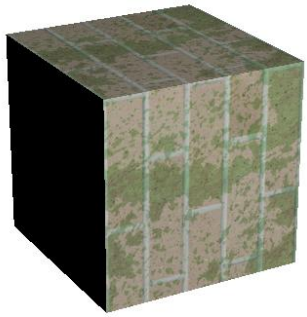

Fig. (l).