CSCE 441 - Computer Graphics

Programming Assignment 3 - Part 2

Deadline: Oct. 26th (11:59 pm)

1 Goal

The goal of this assignment (part 2) is to become familiar with texture mapping.

2 Starter Code

You will be starting with the code you have written for part 1 of this assignment.

3 Task 1

For this assignment, we will be only working on the "sphere.obj" object. Run the code with this object and press `t`. This key switches between the normal color and texture modes. You should be able to see a sphere with a texture of earth. Once you are in texture mode, pressing `n`, `l`, and `m` key switches between three modes of texture mapping.

- `n`: This mode uses nearest neighbors to perform the texture lookup everywhere, i.e., nearest neighbors are done for both minification and magnification.
- `l`: This mode uses bilinear interpolation to interpolate the four neighboring texels to obtain the value at the current texture coordinate. Again bilinear is done for both minification and magnification.
- `m`: This model performs mipmapping which uses performs trilinear interpolation based on the footprint of a pixel in screenspace into the texture domain.

In the next three tasks, you will be implementing these three different texture mapping modes.

4 Task 2

In this part, you will be implementing texture mapping using nearest neighbor. A few things to note:

- In part 1, `RenderCPU` would interpolate the color of three vertices and write the interpolated color into the color buffer. Here, once the texture model is activated by pressing `t`, the code should write the color of the texture into the color buffer instead. Note that your code should still operate as usual in the color mode. So you should have a flag that switches between the two modes in your `RenderCPU` function.
- In "main.cpp" there is vector array, called texture, which stores the texture image at multiple scales. The one you will be using for this part is the original image which is stored in the first element of the vector, i.e., `texture[0]`. We will be using the multi scale images for task 4 (mipmapping).
- The triangle class already comes with a variable, called `t`, which holds the texture coordinates of the three vertices.
- To obtain the texture coordinate at the current pixel, you need to interpolate the texture coordinates of the three vertices using barycentric coordinates. Note that you should perform perspective interpolation as discussed in the class.
- As discussed in the class, the texture space is between 0 and 1 in both dimensions. However, the texture coordinates could be less than 0 or greater than one. You need to map these values to the range [0, 1] using a wrap around function. This function, for example, maps 1.25 to 1.25 - 1 = 0.25. Similarly, it maps a negative value like -0.25 to 1 - 0.25 = 0.75. You should apply this function to every component of your interpolated texture coordinate.

- Once your interpolated texture coordinate is between 0 and 1, you need to map this space to image space before being able to use the interpolated texture coordinate. This can be simply done by multiplying the x component with `texWidth` and y component with `texHeight`.
- Once you have the scaled interpolated texture coordinate, you need to look up the color cooresponding to this coordinate in the texture image. However, the interpolated texture coordinate has floating point components. So it does not directly correspond to one of the pixels in the texture image. Since we will be doing nearest neighbor, simply compute floor of the x and y components of the interpolated texture coordinate and use those instead to look up the color in the texture image.

5 Task 3

Here, you will be implementing texture mapping using bilinear interpolation. A few things to note:

- Most of the steps are similar to the steps in Task 1. So you should have a modular implementation (writing different components as a function) to be able to reuse various stages here.
- The main difference is in the last step where you sample the texture based on the interpolated texture coordinate. Instead of just finding the nearest neighbor, you should use bilinear interpolation (as discussed in the class) to obtain the color of the interpolated texture coordinate, from the 4 neighboring texels.

6 Task 4

Here, you will be implementing mipmapping. A few things to note:

- So far you have been using the first element of the texture vector, i.e., `texture[0]` (the original texture), for texture mapping. For this part, you will be using the full multiscale texture. The `texture` vector contains 11 images (indicies from 0 to 10) corresponding to the texture image at different filtering levels (see slide 61 of the shading lecture). To make the implementation simpler, we have processed all the levels to have the same resolution, but the images at the higher levels are blurrier.
- To implement mipmapping, you need to first find the right scale ($L = \log_2 D$). This can be done by the gradient formula in slide 66.
- The calculated scale has a floating point value. You should first find the two closest levels to this. Then obtain the corresponding color for each level using bilinear interpolation. Finally, perform a linear interpolation between the two calculated colors to obtain the final color.

7 Deliverables

Please follow the instruction below or you may lose some points:

- You should include a README.txt file that includes the parts that you were not able to implement, any extra part that you have implemented, or anything else that is notable. Note that the README file should be in ”txt” file format and you should place it in the root folder next to the “src” folder and “CMakeLists.txt” file.
- Your submission should contain the “src” folder. DO NOT include the “build” and “resources” folders as well as the CMakeLists.txt.

- Zip up the whole package and call it “Firstname_Lastname.zip”. Note that the zip file should extract into a folder named “Firstname_Lastname”. **So you should first put your package in a folder called “Firstname_Lastname” and then zip it up.** The zip file structure should be exactly as follows:

  “`firstname_lastname.zip`”

  - “`firstname_lastname`”
    - “src”
    - README.txt

- Submit the package through Canvas.

Note: Once you upload your zip file, Canvas might adds extra information to the name of your zip file, which is fine. The most important thing is to put everything in a folder “Firstname_Lastname” and then zip this folder up. Canvas will not change this folder name.

8 Ruberic

Total credit: [100 points]

[05 points] - Be able to switch between the texture and color mode using “t”
[10 points] - Handle the wrap around boundary condition
[15 points] - Perspective correct interpolation
[10 points] - Nearest neighbor texture mapping
[15 points] - Bilinear texture mapping
[45 points] - Mipmapping