SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

50.017 Graphics and Visualization

Assignment 3 – Texture Mapping

Handout date: 2024.02.29

Submission deadline: 2024.03.11, 11:59 pm

Late submissions are not accepted

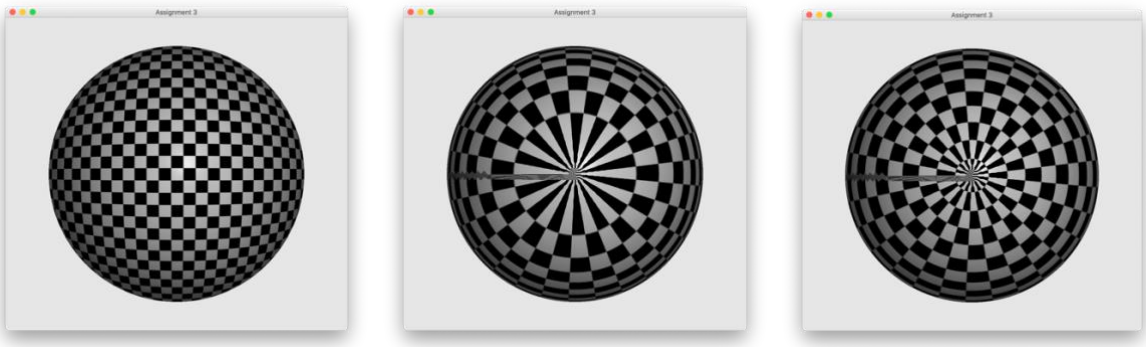

Figure 1. Expected program output on sphere.obj. From left to right: planar, cylindrical, and spherical parameterizations to generate texture coordinates.

In this assignment, you will learn how to use simple parameterizations to generate texture coordinates for a given mesh model and to draw a textured mesh model with OpenGL. "TODO" comments have been inserted in main.cpp to indicate where you need to add your implementations.

1. User Interface

After running the starter code, it should prompt you to enter filename.obj. Once you input an obj file name (e.g., sphere.obj), the starter code will load and render the 3D model for you; see Figure 2. The starter code also does some pre-processing on the 3D model: scale the 3D model such that the maximum edge length of the 3D model's bounding box is equal to 1.

You can interact with the rendered 3D model in the same way as in Assignment 1:

- Left mouse drag will rotate the model around a mapped axis based on the mouse motion.
- Shift + left mouse drag will translate the model in the screen space based on the mouse motion.
- Scrolling the mouse will scale the model to make it either smaller or bigger depending on the mouse scrolling direction.

Once you have implemented the tasks in Section 2. You can use the following keys to switch among different renderings:

- Press key 'p' to show 3D model with planer texture mapping
- Press key 'c' to show 3D model with cylindrical texture mapping
- Press key 's' to show 3D model with spherical texture mapping

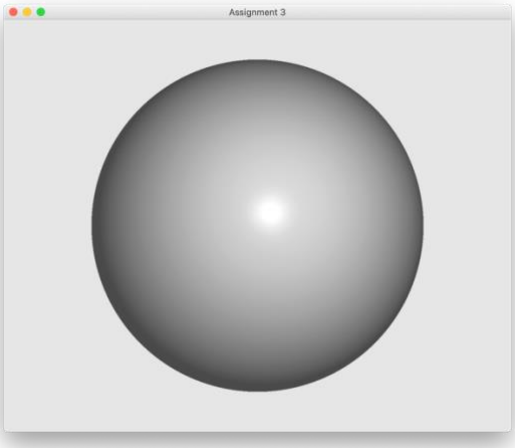

Figure 2. User interface of the starter code after loading sphere.obj

2. Texture Mapping

To do texture mapping, you need to have a 2D texture image as well as texture coordinates for each vertex of the 3D mesh model. The starter code will load the texture image saved at `data/texture.png` and store it in an array of the program. Your task is to generate texture coordinates for each mesh vertex using three simple parameterizations.

2.1 Planar Parameterization

In this task, you need to generate texture coordinate for each mesh vertex using planar parameterization. This task should be implemented in function:

```
void calcPlanarMapping()
```

The 3D coordinate of ith vertex is stored in `objModel.vertices[i].v[3]`. Your goal is to compute (u, v) texture coordinate for each vertex based on its 3D coordinate using planar parameterization, and store this (u, v) texture coordinate in `objModel.vertices[i].t[2]`.

You should use the following equations to compute the texture coordinate (u, v) of a vertex with position $(x, , y, z)$:

$$u = \frac{x - x_{\min}}{x_{\max} - x_{\min}}, \qquad v = \frac{y - y_{\min}}{y_{\max} - y_{\min}}$$

where $x_{\min}$, $y_{\min}$, $x_{\max}$, and $y_{\max}$ are stored in `objModel. bBox.minP[0]`, `objModel. bBox.minP[1]`, `objModel. bBox.maxP[0]`, and `objModel. bBox.maxP[1]`, respectively.

2.2 Cylindrical Parameterization

In this task, you need to generate texture coordinate for each mesh vertex using cylindrical parameterization. This task should be implemented in function:

```
void calcCylindricalMapping()
```

The goal of this task is the same as Section 2.2, except that you are using cylindrical parameterization.

2.3 Spherical Parameterization

In this task, you need to generate texture coordinate for each mesh vertex using spherical parameterization. This task should be implemented in function:

```
void calcSphericalMapping()
```

The goal of this task is the same as Section 2.2, except that you are using spherical parameterization.

To check whether your texture mapping is correct or not, you can compare your renderings with those in Figures 1, 3, and 4.

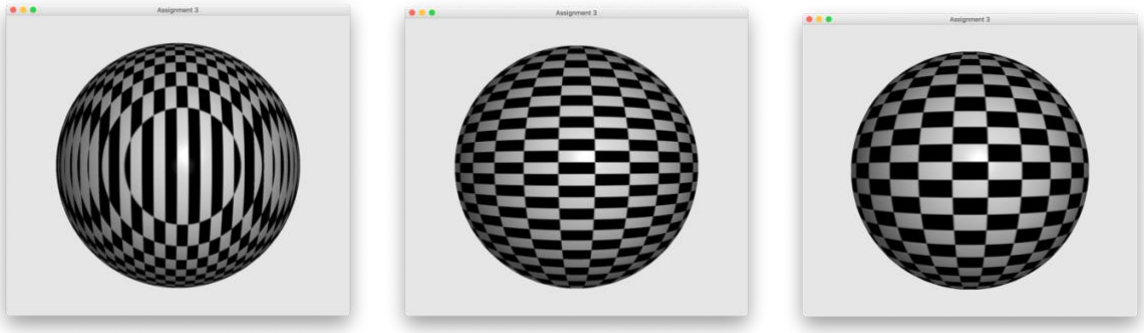

Figure 3. Another view of textured sphere.obj. From left to right: results with planar, cylindrical, and spherical parameterizations.

Figure 4. Textured bunny.obj. From left to right: results with planar, cylindrical, and spherical parameterizations.

You can also generate renderings using different texture map images. To this end, you need to prepare a 1024x1024 PNG image (with only RGB channels) and use it to replace the image at `data/texture.png`. We have prepared another three texture map images for you in the `data` folder (`texture_wood.png`, `texture_stone.png`, and `texture_scene.png`). Figure 5 shows the rendering results of garg.obj with these three texture maps.

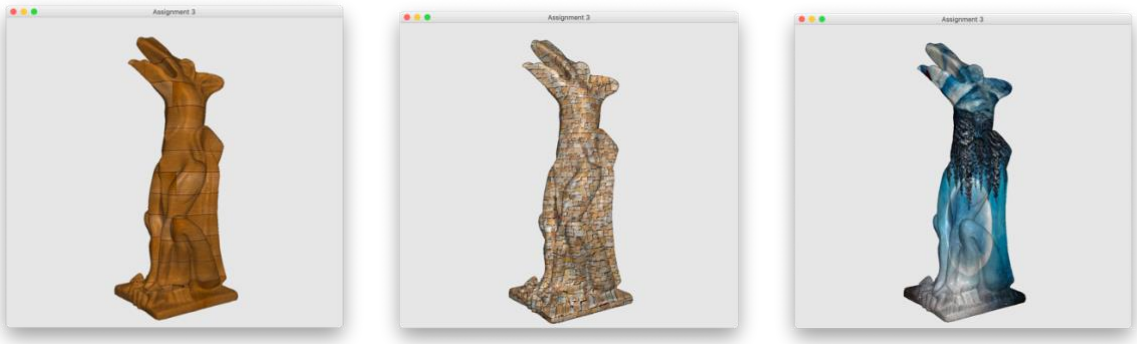

Figure 5. Textured garg.obj with planar parameterization. From left to right: results with texture_wood.png, texture_stone.png, texture_scene.png as the texture map respectively.

3. Grading

Each part of this assignment is weighted as follows:

- Planer Parameterization: 30%
- Cylindrical Parameterization: 30%
- Spherical Parameterization: 40%

4. Submission

A .zip compressed file renamed to AssignmentN_Name_I.zip, where N is the number of the current assignment, Name is your first name, and I is the number of your student ID. It should contain only:

- The **source code** project folder (the entire thing).
- A **readme.txt file** containing a description of how you solved each part of the assignment (use the same titles) and whatever problems you encountered. If you know there are bugs in your code, please provide a list of them, and describe what do you think caused it if possible. This is very important as we can give you partial credit if you help us understand your code better.
- A couple of **screenshots** clearly showing rendered images of textured 3D models.

Upload your zipped assignment to e-dimension. Late submissions receive 0 points!