# CSE4431: Lab 1

Burton Ma

Mon 16 Jan, 2012

## 1 Introduction

In the first part of this lab you will modify a small GLUT program to retrieve the modelling transformation of the modelview transformation. This should be a useful review of simple OpenGL and GLUT programming.

The second part of the lab introduces you to the `glman` tool used in the textbook. You will modify a fragment shader that applies a brick pattern to rendered geometry.

## 2 Retrieving the Modelling Transformation

Recall that OpenGL maintains a modelview matrix that is used to transform every vertex sent down the rendering pipeline. The modelview matrix is the product of the viewing transformation and the modelling transformation:

$$\mathbf{T}_{\text{modelview}} = \mathbf{T}_{\text{viewing}} * \mathbf{T}_{\text{model}} \tag{1}$$

The viewing transformation is (almost) always a rotation followed translation:

$$\mathbf{T}_{\text{viewing}} = \mathbf{D}_{\text{viewing}} * \mathbf{R}_{\text{viewing}} \tag{2}$$

OpenGL allows you to query the state of the modelview matrix using code such as

```
// Get the modelview matrix and store it in Tmodelview
GLdouble Tmodelview[16];
glGetDoublev(GL_MODELVIEW_MATRIX, Tmodelview);
```

If the viewing transformation is known, then the modelling transformation can be computed as:

$$\begin{aligned}
\mathbf{T}_{\text{model}} &= \mathbf{T}_{\text{viewing}}^{-1} * \mathbf{T}_{\text{modelview}} && (3)\\
&= (\mathbf{D}_{\text{viewing}} * \mathbf{R}_{\text{viewing}})^{-1} * \mathbf{T}_{\text{modelview}} && (4)\\
&= \mathbf{R}_{\text{viewing}}^{-1} * \mathbf{D}_{\text{viewing}}^{-1} * \mathbf{T}_{\text{modelview}} && (5)\\
&= \mathbf{R}_{\text{viewing}}^{T} * \mathbf{D}_{\text{viewing}}^{-1} * \mathbf{T}_{\text{modelview}} && (6)
\end{aligned}$$

## 2.1 Lab Exercise 1

Run the program /cse/course/4431/labs/01/lab01.exe (found under the Y: network drive). When you do so, move the window that renders the cube so that you can see the console (where some text will be output). If you right click in the cube window, a menu of various modelling transformations will appear. When you select a menu entry, the modelview transformation, $\mathbf{T}_{\text{modelview}}$, will be applied to the original cube (centered on the origin) and the modelling transformation, $\mathbf{T}_{\text{model}}$, will be output to the console. Your task is to reproduce the code that computes and ouputs $\mathbf{T}_{\text{model}}$ (i.e., implement Equation 6).

1. Create a new Console Application in Visual Studio 2010
2. Add the GLUT program /cse/course/4431/labs/01/lab01.cpp to the project
3. Modify the program so that it correctly prints the current modelling transformation to the console; you will need to modify the functions printModellingTransformation which is called by the function processMenuEvent.

## 3 Introduction to glman and Shaders

glman is the tool that is used in the textbook that lets you prototype shaders without having to write any OpenGL code. glman reads in shaders stored as plain text and generates a simple user interface that allows you to adjust parameters used by your shaders. The glman is documented in Chapter 4 of the textbook.

glman can be found as /cse/course/4431/glman/Executables/glman.exe
Copy the files brick.frag, brick.glib, and brick.vert from the folder /cse/course/4431/labs/01/ into a folder owned by you. Start glman and use it to load your copy of the GLIB file brick.glib

Use the interface to adjust the values of uBrickColor, uMortarColor, uBrickSizeX, and uBrickSizeY and observe what happens to the rendered image.

### 3.1 Modifying the Fragment Shader

The current fragment shader uses a constant value to control the width of the mortar lines between the bricks. It is easy to modify the fragment shader so that the width of the mortar lines becomes an adjustable parameter.

Edit your copy of the fragment shader file brick.frag:

1. Add two new uniform variables after the line uniform float uBrickSizeY;

   ```
   uniform float uBrickPctX;
   uniform float uBrickPctY;
   ```

   These variables represent what fraction of a brick should be shaded with the brick color; a value of 0.95 means that 95% of a brick should be shaded with the brick color and the remaining 5% should be shaded using the mortar color. There are two variable so that we can control both the horizontal and vertical fractions of the brick that should be shaded with the brick color.

2. The shader currently uses a value of `0.85` for both the horizontal and vertical fractions. Find the line

   ```
   vec2  BrickPct = vec2(0.85, 0.85);
   ```

   and change it to

   ```
   vec2  BrickPct = vec2(uBrickPctX, uBrickPctY);
   ```

3. Save your edits.

### 3.2 Modifying the GLIB File

The GLIB file read by =glman= specifies the camera setup, the shader files, the input parameters to the shaders, and the geometry to be rendered. Now that our fragment shader is expecting two additional inputs, we have to modify the GLIB file if we want to adjust these inputs.

Edit your copy of the fragment shader file `brick.glib`:

The part of the file that specifies the inputs to the shaders begins with the tag `Program`. Modify this part of the file so that it reads:

```
Program  Brick  \
  uBrickSizeX <0. 0.3 1.> \
  uBrickSizeY <0. 0.1 1.> \
  uBrickPctX <0. 0.8 1.> \
  uBrickPctY <0. 0.8 1.> \
  uBrickColor {1. 0. 0.} \
  uMortarColor {0.5 0.5 0.5}
```

The notation `uBrickPctX <0. 0.8 1.>` creates a slider that can be used to adjust the value of `uBrickPctX` to be between `0.` and `1.` and sets the initial value to `0.8`

Save your edits.

### 3.3 Reload the GLIB File

Once you have saved your edits, you can use =glman= to reload the GLIB file without restarting =glman= (reloading the GLIB file has the side effect of reloading all of the shaders). Reload your copy of =brick.glib= and confirm that there are now two new sliders that allow you to adjust the brick fractions.

## 4 Understanding the Shaders

You will need to read the last two pages of Chapter 2, all of Chapter 3, and skim Chapter 5 of the textbook to fully understand the shader implementations used in this lab. The handout is the source of the shaders used in this lab (modified to work with =glman=).