Intro to Graphics Spring 2017 – Lab 3

Due Date: February 1st

Lab Objective

- Experiment with geometric transformations using transformation matrices.

Lab Setup

For Qt Creator in the labs, you can use the following instructions:

1. Download the lab3_qtproj.zip file from the Connex assignment.
2. Unzip and open lab3.qbs inside it using Qt Creator.
3. Compile and run! ☺

If you're using Visual Studio, you can do the same thing as above with lab3_vs2015.zip.

If you want to build it yourself, download lab3_custom.zip. Compile main.cpp and the MiniFB files for your operating system:

- For Windows: WinMiniFB.c
- For Mac: MacMiniFB.m, OSXWindow.m, OSXWindowFrameView.m. Link with Cocoa.
- For Linux: X11MiniFB.c. Link with X11.

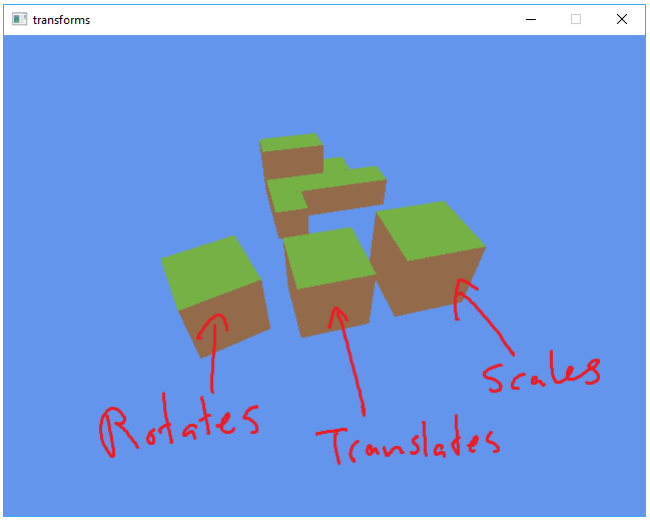
Submission Guidelines

Implement the missing code in main.cpp. Submit your code and a screenshot of your results to Connex.

You only need to submit main.cpp and the screenshot, not any project files or executables.

Expected Result

You should have something like the following image:



Lab Instructions

You should render a scene with three cubes:

- One that demonstrates rotation.
- One that demonstrates translation.
- One that demonstrates scale.

Render the three cubes side-by-side, similarly to the screenshot in the example results.

In the code, you will find three blocks of code with TODOs, one for each of the three cubes. You must implement their transforms by setting their respective "model" matrices.

Additionally, implement a transform that rotates the whole scene. Do this by setting the "scene" matrix. The scene matrix has a TODO above it in the code.

Once you've finished these requirements, feel free to play around and improve the scene (just for fun).

Transformation Matrices with GLM

To create a translation/rotation/scale matrix, you can use the following functions from GLM:

```
mat4 translate(vec3 offset);

mat4 rotate(float angle, vec3 axis);

mat4 scale(vec3 factor);
```

You can combine matrices using the * operator. For example:

```
mat4 a = …;
mat4 b = …;
mat4 c = a * b;
```

To create an identity matrix, just call the constructor of mat4. For example:

```
mat4 id = mat4();
```

Animation

There is a variable `curr_time` available in `render_scene()`. This function stores the time (in seconds) since the program started. You can use this to make the program animate. Try making effects using `sin(curr_time)`.