CS3241 Computer Graphics (2020/2021 Semester 1)

Lab Assignment 3

Release Date: 16 October 2020, Friday

Submission Deadline: 1 November 2020, Sunday, 11:59 PM

LEARNING OBJECTIVES

Writing OpenGL program to simulate planar reflection using texture-mapping and a multi-pass rendering technique. After completing the programming assignment, you should have learned how to

- set up texture mapping in OpenGL,
- model and draw texture-mapped objects,
- set up off-center view frustum,
- read back image in framebuffer for texture mapping, and
- simulate planar reflection using a multi-pass rendering technique.

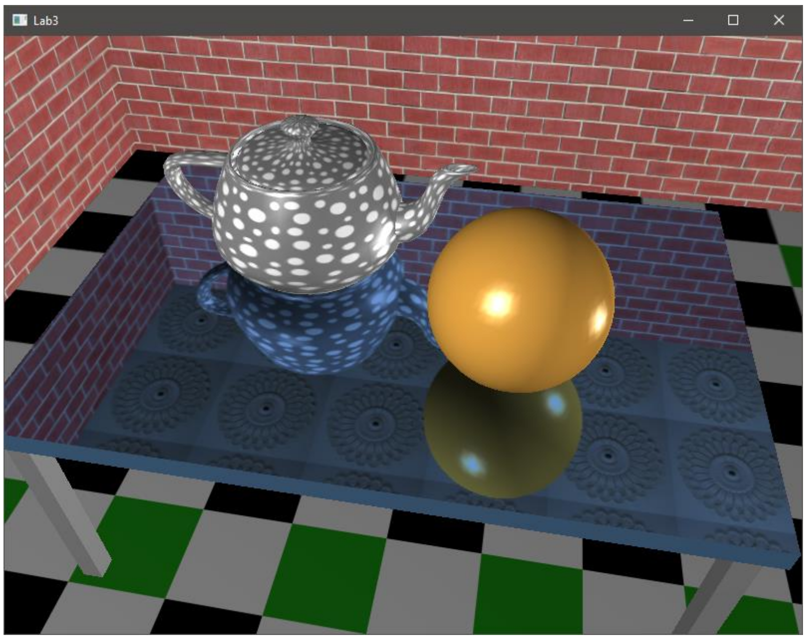
TASKS

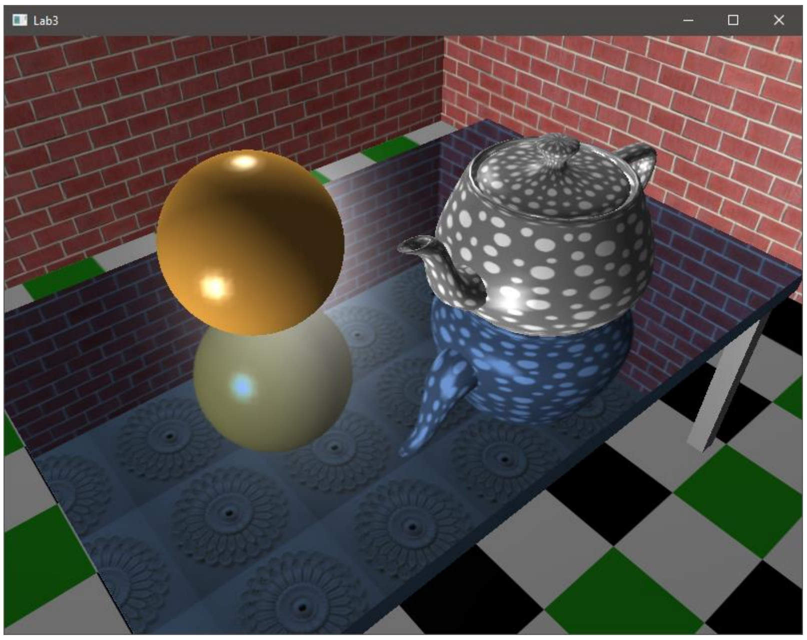
Please download the ZIP file **Lab3_todo.zip** from the **Lab Assignments** folder in LumiNUS Files.

You are to complete an **incomplete OpenGL program**, so that it can simulate **planar reflection**, using a multi-pass rendering technique and the texture-mapping capabilities provided by OpenGL. You have to complete the program according to the following requirements.

<u>Task 1</u>

You have to complete **main.cpp** to produce the planar reflection that you see in the following example images:





You may try the **completed executables: Lab3_done_win.exe** (for Windows), **Lab3_done_mac** (for macOS), and **Lab3_done_linux** (for Linux) found in the same ZIP file to see the required result. (On macOS and Linux, use the command `sudo chmod +x Lab3_done_*` to give the file execute permission before running it.)

Please read the instructions shown in the console/terminal window to learn how to operate the program.

The 3D scene contains a table with a flat rectangular semi-reflective table-top. The scene is also populated with other objects, at least some resting on the table-top. The table-top must reflect the scene. Here are some additional requirements:

- The reflection on the table-top is created by texture mapping a reflection image onto the table-top rectangle. The reflection image is generated by drawing the scene seen from an **imaginary viewpoint**, which looks through the table-top from under the table. This rendered image is then copied from the color buffer to a texture object, to be used for texture mapping the table-top rectangle.

- The reflection on the table-top should not be 100% (it is not a perfect mirror), and the underlying diffuse color and lighting on the table-top must still be visible. (Hint: use the correct texture function/environment.)

- **Mipmapping** must be used for all texture mapping, including for the reflection texture mapping. For the texture object that contains the texture image copied from the color buffer, you have to set the texture object using

  ```
  glTexParameteri(GL_TEXTURE_2D, GL_GENERATE_MIPMAP, GL_TRUE);
  ```

- You are not allowed to use the **stencil buffer** for this assignment.

- Write your code immediately below the locations marked "`WRITE YOUR CODE HERE`". There are **three** of such locations.

- You are allowed to modify only **main.cpp**. You are not required and must not change any other source files.

Task 2

In addition to the given teapot and sphere, you have to **add at least one new texture-mapped object** into the scene. The new object(s) should be positioned above the table-top, and can float in the air. You may not use the 3D models in the GLUT library (e.g. `glutSolidTorus()`) for your new object(s), and must write your own function(s) to provide the polygons, vertex normal, texture coordinates, and material. Each new object must have at least 4 polygons.

You are allowed to modify only **main.cpp**. You should use your own new image(s) to texture-map your new object(s). As before, **mipmapping** must be used for all texture mapping.

Besides your completed **main.cpp**, you also need to submit the **new texture image(s)**.

This task will be assessed based on the fulfillment of the basic requirements, on the technical difficulty and object's complexity, and on the aesthetics and creativity.

DO NOT HARD-CODE VALUES. You should write your code in such a way that when the values of the constants (defined in the beginning of the program) are changed to other valid values, your program should function accordingly. For example, if the table's height is changed, the reflection from the table-top should still look correct.

Besides what is provided in the ZIP file, you should not use any other third-party libraries. Your code must compile with either the MSVC++ 2017/2019 compiler on Windows, or gcc/clang on a macOS or Linux environment.

Instructions on how to **build the project** may be found at the end of the document.

GRADING

The maximum marks for this programming assignment is **100**, and it constitutes **8%** of your total marks for the module. The marks are allocated as follows:

- **Task 1 — 60 marks**

- **Task 2 — 40 marks**
  - 20 marks — basic requirements,
  - 10 marks — technical difficulty and object's complexity,
  - 10 marks — aesthetics and creativity.

Note that marks will be deducted for bad coding style. If your program cannot be compiled and linked, you get 0 (zero) mark.

Good coding style. Comment your code adequately, use meaningful names for functions and variables, and indent your code properly. You must fill in your **Name**, **Student Number**, and **NUS email address** in the **header comment**.

SUBMISSION

For this assignment, you need to **submit only**

- Your completed **main.cpp** that contains code for both Task 1 and Task 2;
- File(s) of your **new texture image(s)** for Task 2. They must be in the **images** subfolder. Total **image files' size** must not exceed **5 MB**.

You must put it/them in a ZIP file and name your ZIP file ***your-student-number*_lab3.zip**. For example, **A0123456X_lab3.zip**. All letters in your student number must be capitalized.

Submit your ZIP file to the **Lab 3 Submissions / Group T0*x*** folder in LumiNUS Files, where **T0*x*** is your officially allocated Tutorial group number. Before the submission deadline, you may upload your ZIP file as many times as you want to the correct folder. **We will take only your latest submission.** Once you have uploaded a new version to the folder, you **must delete the old versions**.

DEADLINE

Late submissions will NOT be accepted. The submission folder will automatically close at the deadline.

BUILDING THE PROJECT

For Windows:

You can just open the given Visual Studio 2017 solution file **Lab3.sln**. The executable **Lab3.exe** will be produced in the **Debug** or **Release** folder, depending on whether you have used the **Debug** or **Release** configuration when you built your project. If you are to run the executable **Lab3.exe** directly (not from within Visual Studio), you need to copy it to the parent folder to run it, so that it can find **freeglut.dll** and **glew32.dll**.

Installing Visual Studio

If you do not have Visual Studio already installed on your computer, you can download (from https://visualstudio.microsoft.com/downloads/) and install Visual Studio Community 2017 or 2019. For NUS students, you can also go to Microsoft's **Azure Dev Tools for Teaching** site (https://azureforeducation.microsoft.com/devtools) to download and install Visual Studio Enterprise 2017 or 2019. At the website, you need to sign in with your NUS email address.

When you are installing Visual Studio, make sure you include the **Desktop development with C++** module/workload. If you have already installed Visual Studio earlier, you can check and install this module by running Visual Studio Installer from the Windows Start Menu, and then clicking on "Modify" under your existing install of Visual Studio.

Note that if you are using the Community edition of Visual Studio, you have to sign in using a Microsoft account (you can create a free account if you do not already have one) before you can use it.

For macOS:

A **CMakeLists.txt** file is provided for you to configure a project to build your program. You will need to download and install CMake. In the following instructions, the `<...>` symbols in the commands are merely *placeholders* for your project directory, so you need to replace them with the actual directory.

1. Download CMake for macOS from the link below and drag the icon to the **Applications** folder. https://github.com/Kitware/CMake/releases/download/v3.18.1/cmake-3.18.1-Darwin-x86_64.dmg
2. Add **cmake** to the **PATH** variable, by executing the following command in Terminal:
```
sudo "/Applications/CMake.app/Contents/bin/cmake-gui" --install
```
3. If you are using Xcode, execute the following command to generate build files for XCode:
```
cmake "-H<path/to/Lab3_todo>" "-B<path/to/Lab3_todo/build>" -G "Xcode"
```
4. Navigate to the generated **build** subfolder and open the **Lab3.xcodeproj** file in Xcode.
5. If you are using another text editor, follow instruction Step **2** under 'For Linux' below to generate build files for **gcc**/**clang** and **make**.

For Linux:

A **CMakeLists.txt** file is provided for you to configure a project to build your program. You will need to download and install CMake. In the following instructions, the `<...>` symbols in the commands are merely *placeholders* for your project directory, so you need to replace them with the actual directory.

1. Download and install **freeglut**, **glew** and **cmake** from your distribution's package manager (e.g. **apt**, **yum**, **pacman**). Note that different distributions may name the **freeglut** and **glew** packages differently (for instance, **freeglut3-dev** on Ubuntu, and just **freeglut** on Arch).
2. To successfully compile the lab, you will need appropriate graphics drivers for your hardware and an active display server, such as Xorg or Wayland.
3. Execute the following command to generate Unix makefiles:
   ```
   cmake "-H<path/to/Lab3_todo>" "-B<path/to/Lab3_todo/build>" -G "Unix Makefiles"
   ```
4. Open the lab project directory in an editor/IDE of your choice.

——— **End of Document** ———