Homework Assignment #1, S2 2023

Topic: Mesh representation and transformations

Date Issued: see Wattle page
Due Date : see Wattle page

Weighting: 10%

Instruction:

All homework assignments must be completed individually. We encourage you to discuss the assignments with other students. However, you should not share any of your codes with anyone else. Each student is responsible for implementing the assignment on their own. You may assist other in debugging their codes, but you should not copy and paste. ANU is using TurnItIn to detect possible duplications. Consulting with previous year students who enrolled in this course on specific assignment is also not allowed. You may use the internet as a resource for learning the materials, but you should not borrow any existing codes found online.

The homework assignments involve a significant amount of C++ programming. However, for most cases, a skeletal code base is provided, and you only need to fill in the missing parts, and/or fix bugs if any.

You will submit a single ZIP file as your submission, which must contain the following files:

(1) All source codes (ending in .h, or .hpp, or .cpp), and your Makefile, CMakeLists.txt. Please include all needed source codes for successful compilation. Please also remove all intermediate files (such as ./vs. . /build.) that are not needed for the compilation – Failing to do so will lead to penalty to the marks.

(2) A written HW1 Report (of between 2~ 3 pages A4, 10 point font size, in PDF format, with task statement, methods used, any new features that you have added, any known bugs in your code, answer any questions that have been asked in the task, instruction for the tutor to use your code, example results.)

Your ZIP file must be named as "COMPX610_2023_HW1_UID.zip". Replace 'X' with 4 or 8. Replace the UID with your Uxxxxxxxx; Please submit your ZIP file to Wattle before the deadline. Late submission will lead to penalty as per the ANU policy. Later-than-one-week submission will not be accepted, which may result zero mark, unless a pre-approval for special consideration is obtained in written before the submission deadline.

There are two tasks in this HW1 assignment:

Task-1: OBJ mesh file manipulation:

Get yourself familiar with Wavefront OBJ file format by reading the Wikipedia page:
https://en.wikipedia.org/wiki/Wavefront_.obj_file

You are required to create, load and display 3D meshes stored in Wavefront's OBJ file format. Specifically, you need to complete the following two sub-tasks, and report your results in your HW1 report.

(a) Create by hand an OBJ file of a 3D cuboid mesh consists of 12 triangles, and display it on screen via the MeshLab software.

(b) Use MeshLab to draw the two provided meshes of kangaroo (v1, and v2) , and compare their OBJ files by reading the raw text files of the OBJs, summarize the key differences in your Hw1 Report.

Task-2: Transformations for graphics rendering

From the first two weeks' lectures, you have learned how to use transformation matrices to prepare an object for graphics rendering. Now it is time to put them into practice.

In this task, you will learn how to use matrices to implement geometrical transformations, such as 3D rotation, and perspective projection. Together with the next two homework assignments (i.e., Hw2 and Hw3), you will learn how to implement a simple and effective **rasterizer** for computer graphics rendering, using C++ on a CPU.

We have provided an almost-ready-to-go skeleton code for your to fill in any missing functions etc. Specifically, your task in Hw1 is to produce and fill in a suitable **3D rotation matrix** and a **perspective projective matrix**.

Given a triangle patch in the 3-space, whose coordinates are v0(2.0, 0.0, -2.0), v1(0.0, 2.0, 2.0), and v2(-2.0, 0.0, -2.0). Your task is to display these 3 points on the screen, and draw the wireframe for the triangle.

The function called *draw_triangle* () in the skeleton code is where you should work on. You only need to fill in some suitable transformation matrices. Specifically, you will need to complete the following steps: configuring the object model in the world space, set up the camera view transformation, apply perspective projection and viewport clipping etc.

Please complete the skeleton codes, and in particular complete the following functions:

- Get_model_matrix(float rotation_angle) : fill in the model transformation matrix and return this matrix. In this function, your task is to implement a 3D rotation about the Z axis.

- Get_projection_matrix (float, eye_fov, float aspect_ratio, float zNear, float zFar): construct a projection matrix and return it.

- Main(): add any other necessary codes.

After you have successfully finished the above tasks, the rasteriser will create a window on your screen, and display the triangle. If you press A , or D, the triangle will rotate about XZ axis in different directions. If you click "ESC", the program will exit.

You can also execute the program in command line, in this case there will be no window pop up, and your result will be saved in 'output.png' by default.

./Rasterizer -r 20

You can also test: ./Rasterizer -r 20 my_rendered_img.png.

[Bonus points] You may change the code so that it can also rotate around any axis passing through the origin. Namely, you will need to add a function *Eigen::Matrix4f get_rotation(Vector3f axis, float angle)* to main.cpp.

How to compile and run your code ?

Library dependencies: Eigen, and OpenCV ;
Please use the following commands in order, to compile and test your code.

(1) mkdir build
(2) cd build (3) cmake ..
(4) make -j4
(5) ./Rasterizer.

Note: In doing this homework, you do not need to use triangle.class. Rather, the only files that you will need to modify are: main.cpp, and rasterizer.hpp. The latter is responsible for the creation of the renderer and GUI interface, and the draw functionality.

Submission requirement and Marking criteria:

Please refer to "Homework#1 submission template and marking criteria" posted on Wattle. (Ignore the information below in the shaded region.)

~~Task 1 : (10 marks) : display all the meshes on screen, and include the results in your HW1 report.~~

~~Task 2: (total 80 marks)~~

- ~~Correctly set 3 modelling matrices and projection matrices (20 marks) , explain the steps that you used to get these matrices. List the matrices in your report as well.~~
- ~~Code can be compiled successfully without any errors. (10 marks)~~
- ~~Pressing 'A', 'D', and 'ESC' can function correctly. (20 marks) Include the results in your report.~~
- ~~The Rasterizer class plays a key role in this project. It has the following members and functions: Matrix4f model, view, projection; and set_model(), set_view(), set_projection() ; Please comment on these members and functions. (10 marks).~~
- ~~Overall presentation of quality your report (10 marks).~~
- ~~[bonus feature] Successfully implement the bonus-point feature (10 marks). You need to include examples to demonstrate this advanced feature is implemented correctly. Add explanations about *Eigen::Matrix4f get_rotation(Vector3f axis, float angle) to your HW1 report.*~~
- ~~Make sure that your ZIP can be extracted successfully. If your ZIP is corrupted, a minus-5 marks as a penalty will be applied. In addition, longer-than-3-page report will incur minus-5 penalty.~~

==END==