Australian National University

COMP4610/8610
Computer Graphics

Computer Lab Homework Assignment #2, S1 2024

Topic: Rasterization and Shading
Date Issued: see Wattle page
Due Date: see Wattle page
Weighting: 12 %

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.

Assignment Project Exam Help
https://tutorcs.com

The homework assignments involve a significant amount of C/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.

WeChat: cstutorcs

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 CMakeLists.txt. Please include all needed source codes for successful compilation. Please also remove all intermediate files and folders (such as `.vscode/` and `build/`) that are not needed for the compilation – Failing to do so will lead to penalty to the marks.

(2) A written CLab2-Report (minimum 10-point font size, single column A4, in PDF format)

Your ZIP file must be named as "COMPX610_2024_HW2_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.

Tasks for CLab-2:

Task-1: Rasterization (30/100 marks)

In the previous assignment (CLab-1), you have drawn a wireframe house on screen. In this assignment, we will paint the model with suitable colour. In other words, we are going to rasterize those triangles in the mesh.

You need to complete following tasks:

1. You need to write a function `rasterize_triangle(const Triangle &, bool)` in `rasterizer.cpp` to replace previous `rasterize_wireframe (const Triangle &t)`. This function generally work as follows:
   (a) Find the 2D bounding box of the triangle.
   (b) Traverse each pixel inside the bounding box (using integer index). Then, use the center point of each pixel to test whether this centre point is inside the triangle or not.
   (c) If it is inside the triangle, apply depth value interpolation using its barycentric coordinates, and compare its depth with the corresponding value in the depth-buffer.
   (d) Use Z-buffer algorithm to refresh the Z value in the depth-buffer and set corresponding pixel color.
2. You may find that when you zoom-in the screen, you will find some zigzags (aliasing) along the boundaries of the triangles. Please implement a 2x2 super-sampling method for anti-aliasing. If your code is properly implemented, the boundary of your triangles should appear smooth, i.e. not having zigzag edges.
3. There are always many triangles that are not visible from current viewport. Exclude these triangles in the early stage of rendering will save us much labour. Please implement the Back-face culling algorithm in the `rst::rasterizer::draw` function in `rasterizer.cpp` to remove some of the invisible triangles.

In your report:

- Put a screenshot of the `rasterize_triangle(const Triangle &, bool)` function with anti-aliasing and a screenshot of the code snippet used for back-face culling.
- Briefly explain how you implement back-face culling algorithm.
- Rotate the house around y-axis by 140 degree and render. Put two results (w/ and w/o anti-aliasing) in the report.

Figure1. Expected house rendering w/o anti-aliasing.

Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs

Task-2: Local Shading and Texture Mapping (40/100 marks)

After rasterization, you will find even though the mesh is painted, it still doesn't look realistic. In this task, you are going to write some shaders to colorize the mesh to make it fancier.

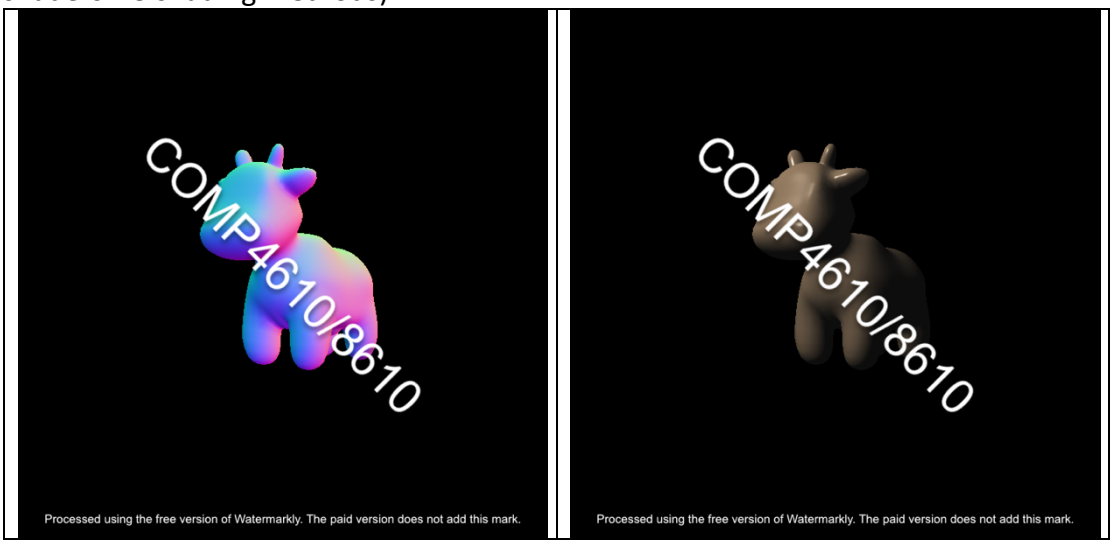
To do this, you need to complete the following tasks:

1. Implement some shaders in `task2.cpp`:
   a. `normal_fragment_shader` to assign the fragment color as the normal direction.
   b. `blinn_phong_fragment_shader` to calculate the fragment color according to the Blinn-Phong reflection model.
   c. `texture_fragment_shader` to calculate the fragment color by texture mapping.
   d. `bump_fragment_shader` to calculate the fragment color by bump mapping.
2. Apart from Phong shading that is used in the provided code, there are other shading techniques including Flat shading and Gouraud shading. Please implement those three shading methods in `rst::rasterizer::rasterize_triangle(const Triangle &, const std::array<Eigen::Vector3f, 3> &, const std::vector<light> &, rst::Shading, bool)` function in `rasterizer.cpp` and compare their difference.
   Hint: Read Wikipedia page on Shading:
   Shading - Wikipedia

Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs

In your report:

- Put four screenshots of the four shaders.
- Put three screenshots of the code in `rasterize_triangle` function, each for one shading method. And talk about the difference in the implementation among those 3 shading methods.
- Put the rendering results in your report. There should be 12 images in total (4 shaders x 3 shading methods).



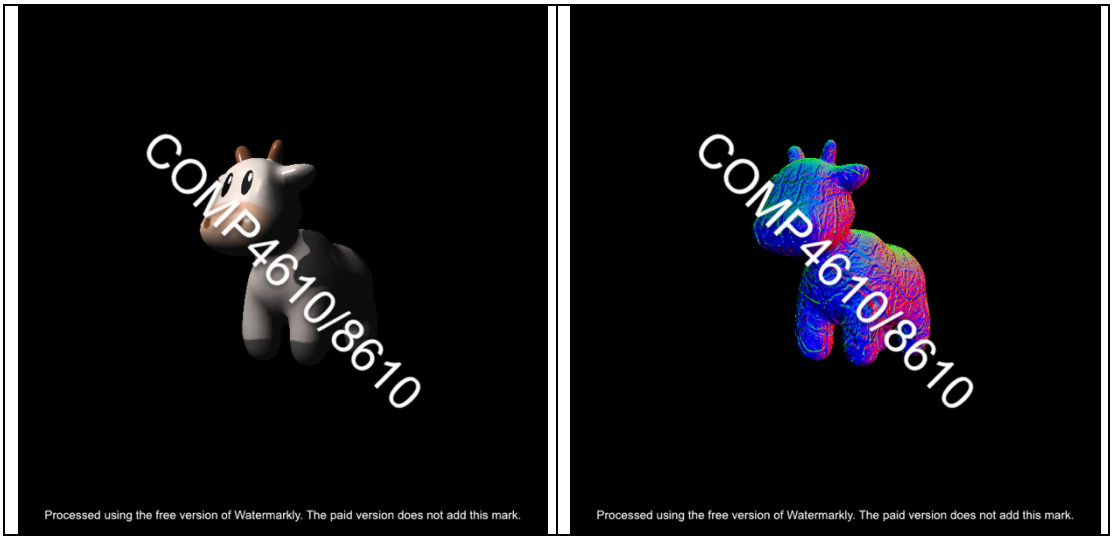


Figure2. Expected renderings of Phong shading

Task-3: MVPR: Modelling-Viewing-Projection and Rendering (30/100 marks)

From completing the above two tasks, you now have a much better grasp of rasterization technique often used for relative rendering in video games. Let us give another try to render the following complex scene of vividly colored cubes with shadowing effect.

Upon successful implementation of the previously mentioned shaders, you should observe visually pleasing results. However, these results may still be inaccurate, as they do not account for shadows. Shadow mapping is a well-established technique used to generate hard shadows. Please implement shadow mapping in your renderer to introduce the shadow effect. You need to model the scene by yourself and render it using rasterization with shadow-map.

Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs

Hint: Read Wikipedia page on Shadow mapping:
https://en.wikipedia.org/wiki/Shadow_mapping

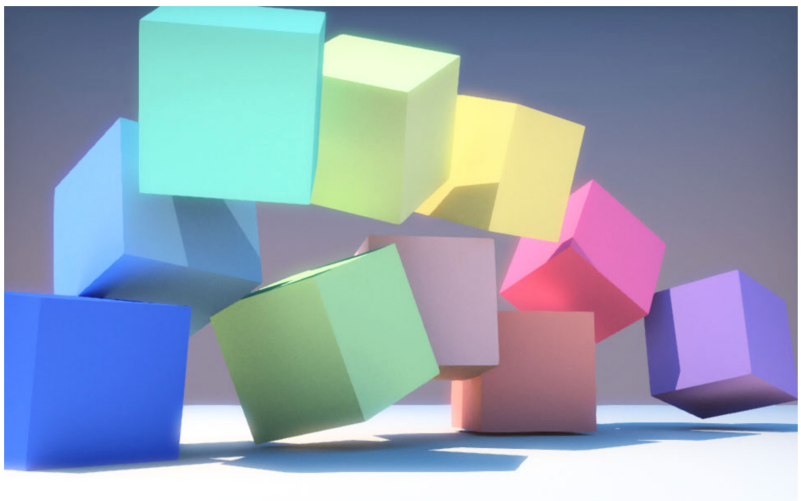


Figure3. Example of target scene.

You task is to model the scene, set up camera viewpoint and light, and rasterize the scene using your C++ code. To simplify the task, you can assume only one light in the scene will cast shadow. We don't provide specific code for task3. You can use any code from previous tasks if necessary to complete `task3.cpp`. Remember, you should implement task3 in our framework without any additional libraries.

In your report:

- Explain how you finish the task, mainly about how you implement the shadow mapping algorithm.
- Put the rendering of your scene in the report to show similar shadow effect as the example.
- Put the scene mesh namely `scene.obj` and texture map `scene.png` in the `models` folder for submission.

== END OF CLAB-2 ==

Assignment Project Exam Help

https://tutorcs.com

WeChat: cstutorcs