DigiPen Institute Of Technology, Singapore

CSD2150/CSD2151 Introduction to Real-Time Rendering

Assignment 8 (NightSky App)

Topics and references

- Multi-pass rendering
- Skybox and Cube mapping
- Gamma correction
- Checkerboard texture
- Physically-Based Rendering
- Fog effect
- Cartoon style rendering
- Discard rendering

Disclaimer

This document provides additional guidelines to the assignment specifications discussed in classes. The structure and content of the document are very similar to that given for assignment 1, with some important differences for successful completion. Some of such differences are *in italics* for your convenience. Use the time between classes to read this document and begin the project.

If you find the information in this document insufficient, review the first class recordings in Team and read the Q&A on the Moodle. Send me your thoughts and questions (if any), so we can discuss the most interesting points in the second class this week.

1. Introduction

This is the only submission for assignment 8 that assess your ability to build an application using the graphics framework CGFW with the Skybox and texture you have developed in the previous assignment and a few new tricks you learn this week.

2. Development guideline

1. First things first: Make sure you've downloaded the latest framework from the Moodle in General section. This will be the starting point for your project. *It is not recommended to use the project from the previous assignment as a base. There isn't much in common that you can reuse, apart from the framework itself.*
2. Create a new project folder with the following naming convention: `<module id>_a8_<student DigiPen id>`. Example: `csd2150_a8_leekuan.yew`.
3. Copy the contents of the `CGFW` folder (the only folder within the downloaded zip file) into this newly created project folder.
4. Locate and open the `main.cpp` file in the project folder. This file, along with the shader files, are the only files that require modification for assignments within this module. If you modify a file, ensure that a file-level documentation block is present at the beginning of the file and that it is updated with accurate information. This block is a mandatory requirement and must contain your name, assignment number, and DigiPen ID.

```
/!*****************************************************************************
\file name.cpp
\author Vadim Surov (vsurov\@digipen.edu)
\co-author YOUR NAME (DIGIPEN EMAIL)
\par Course: CSD2151
\par Assignment: 8
\date 03/02/2025 (MM/DD/YYYY)
\brief The purpose ...
*****************************************************************************/
```

This documentation serves the purpose of providing a reader the purpose of this source file at some later point of time. Also, the header is required to submit the file as evidence of a violation of rules, such as plagiarism.

5. *In `main.cpp` file, create a multiple-pass scene with with the skybox and a mesh rendering code. Define all required shader's parameters as uniforms (ex: screen size, timer, mouse position, all what you need). See an example of a such implementation in the demo app.*
6. *Create a vertex shader that calculate the vertex position in clipping space for both passes. Then for the first pass it calculates vectors for skybox mapping. For the second pass it also calculates normals and positions in view space. Do not forget to pass throw all the texture coordinates.*
7. *Create a fragment shader that makes the rendering similar to what was demonstrated in the class this week with skybox, PBR, fog, cartoon and discard effects. Everything is demonstrated with related codes for scenes and shaders separately in different demo apps this week. So you need to combine all of them together in one app with a few modifications. For example, you need to add few cartoon function calls to the PBR implementation.*
8. Move forward with the project development, making sure to meet all requirements from the next section.

3. Requirements

- (**Q01**) Assignments must be submitted by the deadline. Late submissions within 7 hours will be penalized by one letter grade down. Late submissions exceeding 7 hours will only be permitted with prior approval and a valid excuse, such as a medical certificate issued within the same week. In such cases, the maximum extension allowed is 3 days. Submissions received more than three days after the deadline will not be accepted and graded.
- (**Q02**) The assignment submission in Moodle must be a zip file with the correct name and internal file structure. Refer to sections 2 and 6 for specific instructions. Failure to meet this requirement will result in a failing grade for your submission.
- (**Q03**) The submitted zip file must contain a project that can be successfully opened and built in VS2022 without any errors to produce an executable file. Graders are not permitted to modify the source and configuration files to facilitate a successful build. Failure to build an executable will result in a failing grade for your submission.
- (**Q04**) The compilation process must not generate any warning messages originating from the student's code within the submitted files. (Warning messages from OpenGL during program execution are not subject to this requirement.) Maintain the Warning Level at Level 3 (/W3) during compilation. Do not modify this setting in the project. Failure to meet this requirement will result in a one-letter grade deduction from your submission.

- (**Q05**) The program must be free of any run-time errors, including but not limited to: hangs, crashes, unhandled exceptions, and memory leaks. Failure to meet this requirement will result in a failing grade for your submission.
- (**Q06**) All modified by a student source code files must include a file-level documentation block. Refer to section 2 for specific instructions. Failure to meet this requirement will result in a failing grade for your submission.
- (**Q07**) To achieve the maximum score for this assignment, all required user interactions and outputs must be correctly implemented. Please refer to section 5 for detailed specifications.

4. Grading

Grading usually takes a week, depending on the availability of TAs and the number of submissions. We will strive to make grades available before the deadline of the next assignment, allowing you the opportunity to learn from your previous mistakes.

If you receive a grade or feedback on your submission that is confusing or puzzling, you can review and discuss it with your instructor during office hours.

Please note that we reserve the right to adjust grading rules based on specific situations. For example, if your code generates an excessive number of warnings, the penalty will increase proportionally.

5. User interactions and the output

In this assignment, you have doubled the number of elements to implement. Most of the code is provided in the demo applications from previous and this week. Therefore, the most challening aspect in this assigment for you is to combine them all together in one application.

The application must implement the following six distinct graphical elements:

- A skybox. You can use skybox images from Q&A this week or create your own.
- The ground or floor rendered with a checkerboard texture.
- At least one graphical element with PBR shading.
- At least one graphical element rendered with a cartoon style.
- Rendering with fog effect.
- Rendering with fragment discarding. Do not submit the discard code from the demo. Create something different and not too simple. For example, discard fragments within a circular region.

Four types of interactions must be implemented in the application:

- Orbiting the camera around the center of the world using the mouse.
- Walking the camera between objects using the mouse.
- Switching between camera control types by pressing the keys W (for walking) and O (for orbiting).
- Orbiting the light source around the center of the world using the mouse.

Refer to this week's demo as a reference for implementing some of these elements and interactions. You can also implement the change of shading from PBR to Blinn-Phong for comparison, as was done in the demo app, but it is not required.

Utilize the program's console output to clearly describe the interactions and effects you've implemented beyond the basic requirements. The output will be used for checking your submission by graders.

6. Submission guideline

1. **Clean:** To prepare your project for submission, ensure all temporary files and folders are deleted. This includes any files or folders created during development that are not essential for the project's functionality. To assist with this process, run the `_clear.bat` script located in the framework folder. This script is designed to remove many common temporary files that may have accumulated during development. Importantly, do not include any external libraries (such as `.lib` or `.dll` files) in the final submission package.
2. **Zip:** Create a zip archive of the project folder. (It's implied that the folder itself will be included into the zip.) Ensure the zip file's name matches the project folder's name. Verify the zip file size is under 1MB (maximum allowed size). Upload this zip file to the designated assignment submission page in the Moodle.
3. **Sanity check:** Verify the project's completeness and functionality by downloading from the Moodle, unzipping, compiling, and executing it in a fresh environment.
4. **Report:** Finally, complete the Questionnaire for Assignment. This is mandatory for all assignments. The link is available on Moodle after the link to this assignment. Submissions without the completed questionnaire will not be accepted and graded.

7. Honor code

By submitting your solutions and/or source code for this assignment, you acknowledge that

- you will not give or take answers to solutions and/or source code from your peers or from online sources;
- in addition, you also acknowledge that you will take an active part in ensuring that others will uphold the spirit and letter of this honor code.

If you see a typo or error in this document, please report it to the instructor by email vsurov@digipen.edu or send a message in the team.