COMP3811 Coursework 2

Contents

In Coursework 2, you will work in pairs to create an interactive animated scene using OpenGL. You must use the groups defined in Minerva. Coursework 2 determines 70% of the total mark for COMP3811.

Before starting work on the CW, make sure to study this document in its entirety and plan your work. Pay particular attention to Section 1, which contains information about submission and marking.

1 Submission Instructions

Your submission will consist of source code, a short video ($\lesssim 2$ minutes) and a report ($\lesssim 5$ pages). Submissions are made through Gradescope (do *not* send your solutions by email!). You can use any of Gradescope's mechanisms for uploading the complete solution and report. In particular, Gradescope accepts `.zip` archives (you should see the contents of them when uploading to Gradescope). Do not use other archive formats (Gradescope must be able to unpack them!). *The report is the basis for assessment. The source code and the video are supporting evidence for assertions made in the report.*

The source code must compile and run as submitted on the standard SoC machines found in the UG teaching lab (2.05 in Bragg). Your code must be based on the code provided for this coursework. You must use the included premake build system (which has also been used in the module's exercises and assignments). Your code must compile cleanly, i.e., it should not produce any warnings. If there are warnings that you cannot resolve or believe are in error, you must list these in your report and provide an explanation of what the warning means and why it is acceptable in your case. *Do not change the warning level defined in the handed-out code. Disabling individual warnings through various means will still require documenting the warning in the report.*

The video must be single file called `video.mp4` or `video.avi`. It should use a resolution of 1280×720 and H.264 or H.265 compression. It can use either then standard Y'CbCr color space or sRGB (no HDR). The video should be no more than 2 minutes. It should show the tasks that you have completed. Instructions for recording videos in the UG teaching lab will be provided on Minerva. The videos should not use sound / voice overs, as these might be ignored during marking. *Use proper screen recording software. Do* ***not*** *use a camera or your phone to film the screen.*

The report must be a single PDF file called `report.pdf`. In this report, you must list all tasks that you have attempted and describe your solutions for each task. You may refer to your code in the descriptions, but descriptions that just say "see source code" are not sufficient. Include screenshots from your program where relevant (however, do *not* include screenshots/images of code - if you wish to include code, make sure it is rendered as text in the PDF using appropriate formatting and layout).

In the end of the report (appendix), *include a table that lists each group member's individual contributions to each of the completed tasks*. Document who wrote what code. Make sure that both group members contribute substantially to the code. If both members worked on a certain feature, include rough proportions.

Apply good report writing practices. Structure your report appropriately. Use whole English sentences. Use appropriate grammar, punctuation and spelling. Provide figure captions to figures/screenshots, explaining what the figure/screenshot is showing and what the reader should pay attention to. Refer to figures from your main text. Cite external references appropriately.

Your submission must not include any "extra" files that are not required to build or run your submission (aside from the report). In particular, you must *not* include build artifacts (e.g. final binaries, `.o` files, ...), temporary files generated by your IDE or other tools (e.g. `.vs` directory and contents) or files used by version control (e.g. `.git` directory and related files). Note that some of these files may be hidden by default, but they are almost always visible when inspecting the archive (.zip or similar). Do not submit unused code (e.g. created for testing). Submitting unnecessary files may result in a deduction of marks.

*While you are encouraged to use version control software/source code management software (such as git or subversion), you must **not** make your solutions publicly available. In particular, if you wish to use Github, you must use a private repository. The two members of your group should be the only users with access to that repository.*

Note on plagiarism

You are allowed to discuss ideas with colleagues outside of your group. However do not share code with anybody outside your group. You must program independently and not base your submission on any code other than what has been provided with the coursework and/or in the exercises for COMP3811. As a special exception, you may reuse code from COMP3811 exercises that *you are the sole author of*.

You are encouraged to research solutions and use third-party resources. If you find such, you must provide a reference to them in your report (include information about the source and original author(s)). Never "copy-paste" code from elsewhere – all code must be written yourself. If the solution is based on third party code, make sure to indicate this in comments surrounding your implementation in your code, in addition to including a reference in your report. *It is expected that you fully understand all code that you hand in as part of your submission. You may be asked to explain any such code as part of the marking process. This also applies to code written by other members of the group. If deemed necessary, you may be asked to attend a short individual interview with the instructor(s), where you are asked to explain specific parts of your submission.*

If you wish to use a custom third party library, you must clear this with the instructors for COMP3811 in writing well ahead of your submission. You must provide a clear reason for using this library in your request. Third party libraries may not be used to solve substantial portions of any of the tasks. (Custom third party libraries are likely only necessary for higher bands of grades.)

2 Tasks

Using the provided framework you will create an application that demonstrates the ability to render visual scenes using OpenGL. You must use modern shader-based OpenGL with vertex buffer and array objects.

You are free, within some limits, to create a scene of your choice. Assessment will be based on the technical sophistication of the scene, the scene elements as well as on the possibility for user interaction. The scheme below sets out minimum requirements that must be met for grading in certain bands. Grading within each band is assessed based on: code quality (correctness, clarity, commenting, and efficiency); explanations of design choices and of the implementation in the report; and visual impact.

- **40% - 50%** You must create a visual scene that demonstrates reasonable complexity using several groups of objects. You must include at least one complex object constructed in code (e.g., not loaded from a file). You must implement a perspective projection that adapts to window size when resized. You must include a first-person style 3D camera with which a user can navigate the scene using the mouse and keyboard (WSAD+EQ to control position, mouse to look around, shift to speed up, and ctrl to slow down). Camera controls must be frame-rate independent. You must implement and use the vector and matrix classes from the base code (see `vmlib` folder). You must implement diffuse and ambient shading originating from a point light; the point light must be defined in the CPU code, i.e.., you may not "hardcode" it in the shader(s). The effects of the shading must be perceptible. Your scene must deviate substantially from the simple scenes used in the exercises and demos in COMP3811.
- **50% - 60%** You must fulfill all requirements from the previous band. Your scene must include at least one animated object. The animation must be frame-rate independent. The user must be able to control the animation (pause/resume and speed up/slow down) via keyboard interaction. You must include the full (and correct) Blinn-Phong lighting model *as presented in the course materials*, with at least one point light source. You must have at least one of each of the following in your scene: an object with a mainly diffuse

material, an object with a mainly specular material, and an object with an emissive material. (Include screenshots of these in your report and discuss the difference in appearance.)

- **60% - 70%** You must fulfill all requirements from the previous bands. You must implement texture mapping. At least one object must use one of the textures provided with the coursework (separate download on Minerva); the texture must be visible and recognizable in your 3D scene. You must use multiple light sources (three or more), and they must be distinguishable from each other. You must include at least one object loaded from an external Wavefront .obj file. The loaded object must include both texture coordinates and normals. You must include at least one "transparent" (alpha blended) object.
- **70% - 100%** You must fulfill all requirements from the previous bands. You must include one or more objects that require hierarchical modeling and transformations and that display motion in some of their parts. You must include an object that moves along a complex predefined path (e.g., through multiple line segments or control points). You must use multi-texturing, where an additional texture controls the emissive color or specular exponent of a material. You should integrate a third party UI library (e.g. ImGUI) and control one or more objects with the UI (e.g., adjust its material parameters). You must use a material with alpha masking. You should implement a function to take screen shots on request, by reading back the OpenGL framebuffer and storing it as a PNG file on disk. Finally, you may research a custom (not Blinn-Phong) shading model and implement it (describe the model in your report, along with relevant sources). You should use two or more different shader programs in your rendering. You should come up with additional advanced effects to add (e.g., particle systems, billboarding, image-space filters, environment maps / reflections, ...).

Make sure that you list each element that you have implemented in your report, and include a screenshot that shows this element as rendered by your application. Each element will be assessed and assigned a score based on its complexity, creativity and quality of implementation. Assessment considers primarily the technical aspects, but some marks may be assigned for particularly creative and/or good-looking results.

70 marks

Wrapping up

Please double-check the requirements in Section 1 and ensure that your submission conforms to these. In particular, pay attention to file types (archive format and report format) and ensure that you have not included any unnecessary files in the submission.