Computer Graphics 2018
Assignment 3 – Part 2: Final Project

1. Overview

For your final prac you will write a complete graphics application, with multiple models interacting in a 3D environment. You are free to write an application that contains many aspects of either a game or an animation.

The main aim of the prac is to get experience implementing a full graphics application.

The prac is worth 15% of your overall mark for the course. It is due on Wednesday, June 13 at 11.59pm.

This assignment relates to the following ACS CBOK areas: abstraction, design, hardware and software, data and information, HCI and programming.

2. Code Guidelines

Your computer graphics application will likely be a complex system, and it must be organised for ease of reading and maintenance. The marks allocated for each task are based not only on correctness but also the readability of the code that implements it.

In any case, your submission must provide a makefile that compiles your code and creates an executable called assign3_part2 (in Linux). As before, you can assume the GLFW, GLEW and GLM libraries are in standard locations, but you should include all other source files that your code relies on (including stb_image, and tiny_obj_loader). You will also need to include external texture and OBJ model files that are used by your application. Do not include compiled code of any kind (no executables or intermediate object files).

You MAY start from/or use code taken from the web or elsewhere. You MUST include any code you take from elsewhere in a separate folder labelled external_files. That is, the code in that folder should be the unmodified code you used/extracted parts from. Any code found in your other folders, which has portions that do not appear in any of the files in the external_files is expected to be code you wrote. You still also need to document in your code/report the extent to which you used the external code.

Your main program should run without command line parameters. For example ./assign3_part2 should run your application. Your code should be submitted by the due date with a short report (max of 2 pages) that explains the animation/game you developed and all options (from the list below) that you implemented.

You must use SVN version control software for this assignment. The repository path for this assignment is 2018/s1/cg/assignment3part2.

Hand in your C++ source and Makefile using the web based automark system: https://www.cs.adelaide.edu.au/services/websubmission/

The marking script will just check that it can "make" your submission on a Linux system.

3. What you have to do

Essentially, you need to implement the computer graphics application planned in the part 1 of your assignment 3 (see item 3.6: The plan for the second part of assignment 3). Your application will be marked based on the implementations present in your code, where each implementation is worth a certain number of marks, as detailed below:

- directional light [1 mark]
- point light [1 mark]
- spot light [1 mark]
- loading obj files [1 mark per object, max of 3 marks]
- multiple cameras [2 marks]
- texture mapping [1 mark]
- bump mapping [1 mark]
- height mapping [1 mark]
- light mapping [2 marks]
- parallax mapping [2 marks]
- skybox [1 mark]
- depth cue (e.g., fog) [1 mark]
- alpha blending [1 mark]
- multiple vertex/fragment shaders [2 marks if more than one shader is implemented and used]
- environment mapping (cube or sphere mapping) [2 marks]
- multi-pass method: reflective surface [3 marks]
- multi-pass method: shadow mapping [3 marks]
- procedural generation of geometry [2 marks]
- procedural generation of texture [2 marks]
- procedural generation of particles (e.g., simulation of fire or smoke) [3 to 5 marks, depending on implementation complexity],
- collision detection in animation [3 marks]
- simple sounds (e.g., a crashing sound for collision) [1 mark]
- extra features implemented by the student, but not listed above (number of marks will depend on implementation complexity)

For undergraduate students, if you are working alone, then you need to implement a subset of the points above, where the maximum you can have is 15 marks. If you are working in a group of two students, then the maximum mark is 20, and in a group of three students, the maximum mark is 24.

For postgraduate students, if you are working alone, then you need to implement a subset of the points above, where the maximum you can have is 17 marks. If you are working in a group of two students, then the maximum mark is 24, and in a group of three students, the maximum mark is 30.

Your mark will be proportional to the maximum mark depending on whether you are undergraduate/postgraduate and the number of students in your group. For example, if you are an undergraduate student, working in a group of three students, and your

implementation marks add up to 18, then your mark will be 100% times 18/24 (out of 100%).

Each member of the team must have a statement in the report explaining his/her role. In particular, I expect some distinction between what was done by each member - a statement "we shared all aspects equally" or other similar statements - is not acceptable. There has to be some clear delineation of contribution by feature and/or function/role of contribution. Everyone must write some substantial portions of code (even though a member may, for example, spend more time on producing models, concept, software architecture, for example). Every "chunk" (function/sourcefile - choose a reasonable way to interpret) must have an explicit declaration of authorship. The final mark for each team member will be a function of the mark given to the group and the proportion of work done by the member, as follows: (group mark) times (proportion of work done by team member)/(mean proportion of work done by all team members). For example, if the group (of three students) mark was 80%, and team member 1 worked on 30% of the tasks, team member 2 worked on 40% of the tasks, and team member 3 worked on the remaining 30% of the tasks, then team members 1 and 3 will have mark 80% x 30/33.3 (out of 100%) and team member 2 will have mark 80% x 40/33.3 (out of 100%).

Make sure the report has clear instructions on how to run the program (what keys to press, how to use the mouse, etc). Moreover, there **MUST BE** a designated key (hopefully 'ESC') that when pressed with cause the program to exit, at any stage of running.

It is important that YOU highlight in your report any design decisions you faced and the reason for the choices you took. You need to (without losing a sense of what is pretty trivial and what is worthy of explanation) tell the reader anything that is clever/challenging or not obvious in the task and how you solved it.

Finally, given the nature of this exercise - go about your solution incrementally. Make sure you have **something working** at all stages of development.

Gustavo Carneiro
May 2018