CSC-30019 Games Computing Assessment 2019

Dr Adam Stanton

School of Computing and Mathematics, Keele University

a.stanton@keele.ac.uk

The deadline for submission of this assessment is **Friday 17th January 2020 10h00**.

Introduction

The objective of the practical assessment is to allow you to show your skills in, and understanding of:

- theory of games design;
- games programming in C++; and
- the use of the middleware components studied on the course, specifically the Open Dynamics Engine (ODE) and/or the openFrameworks library.

The assessment constitutes 65% of the overall mark for the module, and comprises three sections:

1. Use your understanding of games design to write a detailed proposal for a 3D computer game application. Justify your plans in terms your understanding of the theory of games design, and also in terms of the technologies and features available in ODE and/or openFrameworks. You should aim to write approximately 2500-3000 words. **[40%]**
2. Use your knowledge of C++, ODE and openFrameworks to implement a prototype of your proposal. **[50%]**
3. Outline clearly and concisely the structure and operation of your prototype (i.e. explain how it works), mention any issues you encountered and briefly discuss any shortfall from your original proposal. You should aim to write approximately 500-1000 words. **[10%]**

1 Proposal (2500-3000 words, 40%)

For the written component of part one, you are asked to write a proposal for a new game. This game could be a completely novel invention of your own, a new version of a popular genre of computer game, or a reimplementation of a classic game. The purpose of this section of the assessment is threefold: first, as a showcase for your creativity and an opportunity to develop a game idea that can be used later as part of a portfolio of work if you decide to apply for jobs in games. Second, to give you the chance to demonstrate your skills in researching the theoretical aspects of games design and arguing a case for a game from that perspective. Finally, it gives you the opportunity to show your technical understanding of the capabilities and limitations of some popular middleware components. Marks will be awarded approximately evenly between:

(a) novel, creative game ideas and uses of the game components;

(b) justifications of why the game will be fun to play, making reference to theoretical work; and

(c) sensible evaluations of the technical feasibility of your proposal, written in terms of the capabilities of the game components.

For this section of the assessment you must undertake your own research and reading. You should expect this to include finding and playing games, perhaps including old games (from sites such as myabandonware), in order to generate ideas for part (a). I suggest you start doing this early in the semester. At the same time, for part (b), you should be developing your own understanding of what a game is and what makes a fun game. We will discuss this in class, but you should take the time to read both of the required texts and to do your own research into other ideas on this topic as well. The texts are in the library in hard-copy, as well as accessible as an online resource. Use the module's reading list to find them. For reference, the two books are:

- Burgun, K. 2013. Game design theory: a new philosophy for understanding games.
- Koster, R. 2014. A theory of fun for game design.

For part (c), your submission can be developed as you gain experience with, and understanding of the game middleware presented during the module. However, the scope of your proposal (including the type of game you will be able to build) is ultimately limited by the capabilities of the middleware we are using–ODE and openFrameworks. Therefore, it is important that you think about the games you research and the ideas you develop for part (a) both in terms of the theory of part (b) and the technical feasibility of part (c). This will allow you to write a proposal that is robust, informed, and implementable, given the constraints of the module.

1.1 Grading guide

Your written work for this section of the assessment will be evaluated in line with the University's generic marking scheme. Where the task involves an element of creative work, i.e. part (a), your submission will again be evaluated making reference to the generic scheme, looking in particular at the coherence and detail of your concept, the depth of presentation and your ability to characterise the idea in concise but meaningful text.

2 Implementation (code, assets, and executable, 50%)

This part of the assessment requires you to program a prototype of your game or game component (e.g. a single level, a simplified version of the game, a demonstration of its key mechanics, or similar.) You are to use the C++ language under the Linux OS to accomplish this, making use of the ODE and/or openFrameworks libraries for 3D physics and graphics.

You are not required to create a new application from scratch, although you may do so if you wish. You can also choose one of the many demo applications distributed with the game libraries studied on the module (ODE or openFrameworks), or the examples given in class or for practical exercises, and make your application using this code as a starting point and adapting it as required. Of course, you can change the demo completely if you need to.

It's important that your program compiles and runs throughout your implementation process so make sure to regularly test this as you go through the iterations of development.

Remember that writing a game for modern hardware is an extremely involved process and you are not expected to produce a professional quality implementation; you are required only to successfully integrate some core components into a working program. Your work for part two should be considered "proof-of-concept" rather than a final, polished implementation. However for full credit you must demonstrate that a significant proportion of the key ideas for the game which you proposed in part one have been implemented. It is clear that the quality of your implementation may depend to some extent on the scope of your original plan, and this will be taken into account when marking. Finally, if you do base your application on existing demo software, then please ensure it is significantly different. Submissions that are demos with a few cosmetic changes count as plagiarism, and this is not difficult to spot.

2.1 Grading guide

Your practical work for part two will be marked according to the guide set out in Table 1. In addition, the University's generic marking scheme will be referred to where appropriate. Neither your source code nor coding style/quality will be marked directly; however, it will be both inspected visually and processed automatically in order to expose plagiarism. The primary mechanism for evaluating your program will be running the software, which is why it is imperative that your code compiles and runs and that you submit a description of the key features of your work in part three. In the case that you don't submit an executable file, and/or your program doesn't compile, then as a last resort your source code will also be used to judge your work. **Please note that in all cases you may also be asked to explain aspects of your program verbally before the final marks are delivered.**

Table 1. Grading guide

Mark	Mark band descriptor
0-39%	Some code has been written. Code does not compile / run. Little or no interaction / graphical display. Few or no aspects of proposal have been successfully implemented.
40-49%	Program compiles and displays on the screen when run. Some aspects of the written proposal have been implemented successfully. A minimal demonstration of the use of the features of the underlying game library or libraries. One or more game component used. Program exhibits instability or logical errors.
50-59%	Program compiles and runs. Many aspects of the written proposal have been implemented successfully. A satisfactory demonstration of the use of the features of the underlying game library or libraries. One or more game components integrated into program. Few or no instabilities evident when program runs.
60-69%	Program compiles and runs. Many or all aspects of the written proposal have been implemented successfully. A very good demonstration of the use of the features of the underlying game library or libraries. One or more game components integrated into program. No instabilities evident when program runs.
70-100%	Program compiles and runs. All aspects of the written proposal implemented successfully. An excellent demonstration of the use of the features of the underlying game library or libraries. No instabilities evident when program runs. Evidence of independent research beyond material delivered in lectures.

2.2 Development environment

This module uses the Linux OS for game development. Under Linux, C++ development, especially where programs use 3D graphics, is somewhat more straightforward than other operating systems. It also means

that you can undertake development using the PCs in the undergraduate labs in Colin Reeves. The lab PCs are pre-installed with the GNU C++ toolchain (including *make* and *g++*), QtCreator IDE and the Atom text editor.

3 Evaluation (500-1000 words, 10%)

For this part of the assessment, please submit a document divided into two parts. For the first part, describe the game's structure (how the computer's execution of your game code flows around the various functions you have written or used) and operation (how to actually play it). For the second part, reflect on any issues you encountered during development and how you might mitigate them if you were to continue the project.

3.1 Grading guide

Your written work for this section of the assessment will be evaluated in line with the University's generic marking scheme. Particular attention will be paid to the quality of your description for the first part, and the depth of the critical stance you adopt for the second.

Assessment Submission

Submit a single .zip file to the assessment drop box on Blackboard. **Name the zip file NNNNNNNN.zip**, where NNNNNNNN is your student ID. This is the first 8 digits of the number on your Keele Card. The zip should uncompress a single directory called NNNNNNNN, which contains three directories: part1, part2, and part3.

- part1 should contain a single .pdf file with your writing for the first part of the assessment. It should be properly referenced and include your student ID and a word count at the top of the document.
- part2 should contain: i) your source code; ii) either a Makefile or other instructions to compile your program written in a file called compile.txt; iii) all the assets (models, textures etc.) required by your game; and, **importantly, iv) a compiled, executable file**.
- part3 should contain a single .pdf file with your writing for the third part of the assessment. You do not need to reference things in this part, but you must include your student ID and a word count at the top of the document.

Please **do not** submit other types of archive (.tar.gz, .tar.xz, .rar), or other types of document (.doc, .docx, .odt). Due to the amount of time it takes to figure out what is where, I will not mark submissions that do not conform to the specification given above. An example layout for submission is available, and also uploaded to the module's Blackboard page.