CODING TASK II: GAME COMPONENT

Version 1.0
BSc Computing for Games

Dr Michael Scott

Introduction

In this assignment, you are required to construct a game component and integrate it into an existing game. You will then present a demonstration of your game component.

Games are often comprised of a rich architecture incorporating many components. As such, coding tasks in the games industry may require you to develop bespoke code on a particular aspect of a game. Many components could be modelled. For example: level design; in-game items; characters; art or audio assets; in-game events; an AI director; and so on. Through this project, you will become acquainted in a practical way with the different techniques and methods that help you to work effectively to build computing solutions. You will gain a better knowledge of coding and its various aspects.

This assignment is formed of four parts: A, B, C, and D.

"Students come into programming classes with a broad range of backgrounds; some have experience in several programming languages, others have never programmed before in their life.

Being able to engage with the community and support each other is important. Upload your code to GitHub and receive feedback from experienced peers. Review your peers' work yourself and really consider what 'quality' actually means and what 'good' source code looks like. Debate, argue, and question others about it — an open and sustained discourse is an excellent way to learn — for both beginners and adepts!"

(a) Present a brief proposal that will:
   i. **identify** a component of game architecture that needs creating;
   ii. **describe** what will be created, using appropriate annotations and user stories;
   iii. and then **outline how** it will be integrated into an existing game project.

(b) Prepare one draft computer program that implements the game component according to the requirements highlighted in the proposal.

(c) Prepare one final computer program that will:
   i. **revise** any issues raised by your tutor or your peers;

(d) Prepare a practical demonstration of your work

Note: All submissions must be clearly distinctive. Members of the same COMP150 team must **not** target the same problem.

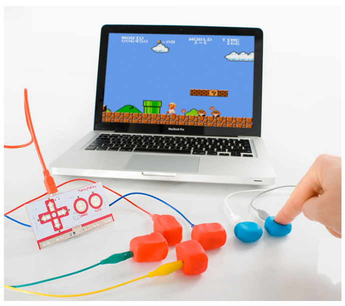
Part A consists of a **single formative submission** with a deadline set in LearningSpace. This work will be assessed on a **threshold** basis. The threshold is set at 5%. This means that 5% of the total marks available for the coursework overall are awarded on a pass or fail basis. In other words, satisfactory submissions will be awarded 5%. However, unsatisfactory submissions will receive 0%.

The following criteria are used to determine a pass or fail for each submission in Part A:

(a) Submission is timely;

(b) Proposed game component is relevant and feasible to implement;

(c) Coding task is non-trivial.


The *MaKey MaKey* allows a multitude of materials to be used to create videogame controllers.

Part B is a **single formative submission** with a deadline set in LearningSpace. This work will be assessed on a **threshold** basis. The threshold is set at 5%. This means that 5% of the total marks available for the coursework overall are awarded on a pass or fail basis. In other words, satisfactory submissions will be awarded 5%. However, unsatisfactory submissions will receive 0%.

Part C is a **single summative submission** and will be assessed on a criterion-referenced basis. This submission is expected to take students from the threshold of 10% (F) up to the maximum of 100% (A*). This means that 80% of the total marks available for the coursework overall will be awarded.

The following criteria are used to allocate marks:

(a) Appropriateness of Requirements;

(b) Appropriateness of Design;

(c) Functional Coherence of Executable Solution;

(d) Maintainability of Source Code;

(e) Sophistication of Source Code;

(f) Professional Practice;

Part D is not grade-bearing; however, attendance is mandatory. Absence will result in a grade capped at 40% (D-).

"Remember, learning to program can take a surprising amount of time & effort — students may get there at different rates, but all students who put in the time & effort get there eventually. Making good use of (reflection and deliberate practice) are an essential part of this process."

— Professor Quintin Cutts

Submission Instructions

Part A

Part A must be completed as a formative submission on GitHub. Fork the GitHub project at the following URL:

`https://github.com/Falmouth-Games-Academy/comp110-coding-task-2`

Write your proposal in the `readme.md` file. Also use this repository for any other digital assets you create (e.g. diagrams and pseudocode), checking them in regularly as you work on your projects. For the proposal, images should be embedded directly in the `readme.md` file.

You will need to show your proposal to your tutor to be signed-off before the deadline. This will need to be arranged in one of your fortnightly tutorials prior to the deadline.

Part B

Part B will be completed as an in-session code review during the appropriate workshop, as scheduled on the LearningSpace. The source code and related assets should be hosted on GitHub before the code review takes place. The single PDF document must be submitted to the LearningSpace by the final submission deadline as shown on LearningSpace. Please note that the LearningSpace will only accept a single PDF document.

You will receive formal feedback three weeks after the submission deadline shown on LearningSpace.

Rhythm games such as *Guitar Hero* and *Rock Band* are excellent examples of games which make use of unique input devices to enhance gameplay.

Part C

Part C must be submitted as a single zip file, containing the source code and assets from the relevant Git repository, on the LearningSpace. This must be submitted by the final submission deadline as shown on LearningSpace. Please note that the LearningSpace will only accept a single zip file (.rar format must not be used).

Part D

No formal submission is required. Please turn up to the demo session at the end of the Semester, with an executable demo of your work and be prepared to discuss it with others.

You will receive formal feedback three weeks after the final submission deadline shown on LearningSpace.

Additional Guidance

"The first 90 percent of the code accounts for the first 90 percent of the development time.

"The remaining 10 percent of the code accounts for the other 90 percent of the development time."

— Tom Cargill

A common mistake in this project is poor planning and poor time management. In such circumstances, it is often the case that students greatly underestimate how much work is involved in first learning programming concepts and then actually applying them in order to write a computer program. Programming is quite unlike other subjects in that it cannot be "crammed" into a last minute deluge just before a deadline, and so relying on knowledge-centric learning strategies (for which such an approach may be possible in other subject areas) will likely result in disaster in the programming context. It is, therefore, very important that students begin their work early and sustain a good pace across the duration of the course. Do a little programming every day, if you can!

"Hofstadter's Law:

"It always takes longer than you expect, even when you take into account Hofstadter's Law."

— Douglas Hofstadter

It is also important to consider that 50% of the work is pushing for that last 5% of the polish. That is, significant effort is needed to push the quality of the work up from a B/C grade up to an A or A* grade. It is, therefore, very important to anticipate this and accommodate enough slack-time for this long phase of polishing in your plan — especially with competing demands from your other assignments.

You may worry about the first deadline as it is quite soon after the assignment is set and not much material has been covered. Please rest assured. The formative submission is supposed to be an early proof-of-concept. It is expected to contain errors, be riddled with mistakes, fail to fulfil the requirements, and generally be of poor quality. However, it is very important to make a start on this project so you are actively using your programming skills across the entire duration of the course. Ideally, you should be programming every day. Considering this, producing implementing the first 3 algorithms within 3 weeks is feasible.

The peer-review component of this work does sometimes raise alarm. However, the only way to learn how to review code is by reviewing code. Your tutor will guide you through the process and provide advice. With practice, it will become clear what is not satisfactory by discussing the quality of work with your peers and your tutor during the review sessions. Ongoing reviewing and discussion on GitHub is advised.

The *Dreamcast Fishing Controller*, released as a peripheral for the game *Sega Bass Fishing*. Even peripherals which appeal to only a small audience can enjoy moderate commercial success.

Additional Resources

- As listed in the module guide.

Marking Rubric

Criterion	Weight	F (0 – 39)	D (40 – 49)	C (50 – 59)	B (60 – 69)	A (70 – 79)	A* (80 – 100)
Satisfactory Preparation of Proposal	5%	The proposal is inappropriate and/or is late.					The proposal has been signed-off by your tutor by the deadline.
Satisfactory Completion of Peer-Review Tasks	5%	No work was submitted for peer-review and/or no peer-review has been submitted and/or either is late.					Work submitted for peer-review on time and reviews of peers' work submitted on time.
Appropriateness of Requirements	5%	No user stories are provided.	Few user stories are appropriately formatted, distinguishable, and easily measured.	Some user stories are appropriately formatted, distinguishable, and easily measured.	Most user stories are appropriately formatted, distinguishable, and easily measured. The scope and relevance of all requirements is appropriate.	Nearly all user stories are appropriately formatted, distinguishable, and easily measured. The scope and relevance of all requirements is appropriate.	All user stories are appropriately formatted, distinguishable, and easily measured. The scope and relevance of all requirements is appropriate.
Appropriateness of Design	10%	No design is presented.	The design is very flawed and/or very poorly described.	The design is flawed and/or poorly described.	The design is acceptable and adequately described.	The design is sound and well described.	The design is exceptional and very well described.
Functional Coherence	15%	The component is non-functional.	Few requirements have been met. There are many obvious bugs.	Some requirements have been met. There are some obvious bugs.	Many requirements have been met. There are few obvious bugs.	The game component is fit-for-purpose. There are almost no obvious bugs.	The game component is fit-for-purpose. There are no obvious bugs.
Sophistication	25%	No insight into the appropriate use of programming constructs is evident from the source code.	Little insight into the appropriate use of programming constructs is evident from the source code.	Some insight into the appropriate use of programming constructs is evident from the source code.	Much insight into the appropriate use of programming constructs is evident from the source code. The program is structured appropriately.	Significant insight into the appropriate use of programming constructs is evident from the source code. The program is structured effectively, such that there is high cohesion and low coupling.	Exemplary insight into the appropriate use of programming constructs is evident from the source code. The program is structured very effectively, such that there is very high cohesion and very low coupling.
Maintainability	25%	The source code cannot be maintained.	There are many problems which affect the maintainability of the source code.	There are some problems which affect the maintainability of the source code. Some clear and appropriate comments are present.	There are few problems which affect the maintainability of the source code. Many clear and appropriate comments are present.	There are almost no problems which affect the maintainability of the source code. Source code is well commented. Doc strings (or equivalent) are provided.	There are no problems which affect the maintainability of the source code. Source code is exceptionally well commented. Appropriate doc strings (or equivalent) are provided.
Professional Practice	10%	GitHub has not been used.	Source code and assets have been checked into the repository only just before a deadline.	Source code and assets have seldom been checked into the repository.	Source code and assets have regularly been checked into the repository. An attempt has been made to document the project using `readme.md` and `changelog.md`.	Source code and assets have regularly been checked into the repository. The first check-in to the repository is in the first half of the semester. The project is appropriately documented using `readme.md` and `changelog.md`. There is evidence of some engagement with the Falmouth Games Academy community (e.g. reviewing peers' pull requests).	Source code and assets have regularly been checked into the repository. The first check-in to the repository is in the first quarter of the semester. The project is exemplary documented using `readme.md` and `changelog.md`. There is evidence of much engagement with the Falmouth Games Academy community (e.g. reviewing peers' pull requests).