PRODUCTION TASKS & TECHNICAL DEMO

Version 1.0
BSc Computing for Games

Dr Michael Scott

Introduction

In this assignment, you will prepare a production prototype of your game. This version of your game will be written in C++ using the Simple DirectMedia Layer (SDL). You will continue to work in small groups (typically, 3–4 students).

The sucess of a digital game is not just determined by the fun of its design, but also by the soundness of its architecture. Put simply, games with serious bugs do not sell. As such, the value of software engineering principles cannot be emphasised enough! Reflecting upon these principles, through the lens of a development project that mirrors industry practice, is therefore important.

This assignment is formed of several parts:

(A) **Prepare**, as a **group**, a plan which will:

i. **identify** the skills and time available to complete the final prototype;
ii. **describe** the key user stories that will comprise the final prototype;
iii. and **highlight** the stories most critical to the technical demo.

(B) **Implement**, as a **group**, an initial production prototype in SDL which will:

i. **illustrate** the core game mechanic;
ii. and **apply** at least **three** non-trivial algorithms.

(C) **Implement**, as a **group**, a final production prototype in SDL which will:

i. **revise** any issues raised by your tutor and/or your peers.

(D) **Prepare**, as an **individual**, an A3 research-style poster that:

i. **outlines and discusses** the engineering of the final prototype;
ii. and **describes one** algorithm that you have implemented.

(E) **Present**, as a **group**, a `technical demo` which will:

i. **clarify** the technical content of your own poster;
ii. and **show** the final production prototype of the game.

"It seems that perfection is attained not when there is nothing more to add, but when there is nothing more to remove."

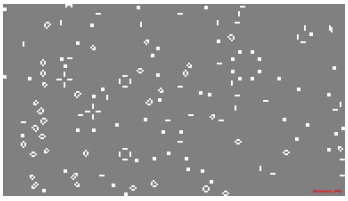
— Antoine de Saint-Exupéry

"Good judgment comes from experience and experience comes from bad judgment!"

— Fred Brooks Jr

"Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it."

— Brian Kernighan



Mathematician, John Conway, devised the Game of Life to simplify concepts presented by mathematician John von Neumann, who sought to devise a hypothetical machine that could replicate itself and evolve.

Assignment Setup

This assignment is a **product development task**. Fork the GitHub repository at the following URL:

`https://github.com/Falmouth-Games-Academy/comp160-game`

Use the existing directory structure and, as required, extend this structure with sub-directories. Ensure that you maintain the `readme.md` file.

Modify the `.gitignore` to the defaults for **Visual Studio**. Please, also ensure that you add editor-specific files and folders to `.gitignore`.

Part A

Part A consists of a **single formative submission**. This work is **collaborative** and will be assessed on a **threshold** basis. The following criteria are used to determine a pass or fail:

(a) Only well-formed user stories are included;

(b) The user stories form a comprehensive plan;

(c) The plan has reasonable scope and is feasible.

To complete Part A, setup and populate the team Trello board. Ensure that all members of the team are added to the board. Show it to your tutor in the scheduled CPD catch-up tutorial.

You will receive immediate **informal feedback** from your **tutor**.

Part B

Part B is a **multiple formative submissions**. This work is **collaborative** and will be assessed on a **threshold** basis. The following criteria are used to determine a pass or fail:

(a) Submission is timely;

(b) Enough work is available to conduct a meaningful review;

(c) A broadly appropriate review of another team's work is submitted.

To complete Part B, prepare a draft version of the production prototype. Ensure that the source code and related assets are pushed to GitHub and a pull request is made prior to *each* scheduled sprint review session. Ensure that you attend *each* scheduled sprint review session.

You will receive immediate **informal feedback** from your **tutor** and **peers**.

Part C

Part C is a **single summative submission**. This work is **collaborative** and will be assessed on a **criterion-referenced** basis. Please refer to the marking rubric at the end of the brief for details on the criteria.

To complete Part C, revise the production prototype based on the feedback you have received and tidy up any incomplete features. Please also ensure that you include appropriate screen-shots of the Trello board in a separate folder. Then, upload the source code to the LearningSpace. Please note, the LearningSpace will only accept a single `.zip` file. The recommended way of generating the `.zip` file is using the *(Download Zip)* button on the GitHub website.

You will receive **formal feedback** three weeks after the final deadline.

Part D

Part D is a **single summative submission**. This work is **individual** and will be assessed on a **threshold** basis. The following criteria are used to determine a pass or fail:

(a) Submission is timely;

(b) At least **one** non-trivial algorithm is described

(c) The algorithm was implemented by the person submitting the work

(d) The description of the algorithm is detailed and accurate

(e) There is sufficient detail to show how the algorithm fits into the overall architecture of the game

(f) There is little to no overlap with other members of the team

To complete Part D, prepare the poster using any word processing and/or presentation tool. Then, upload the relevant files to the LearningSpace. Please note, the LearningSpace will only accept a single `.zip` file.

You will receive **informal feedback** from your tutor three days prior to Part E.

Part E

Part E is a **single summative submission**. This work is **collaborative** and will be assessed on a **criterion-referenced** basis. Please refer to the marking rubric at the end of the brief for details on the criteria.

To complete Part E, look over your notes to prepare yourself to answer questions. Ensure that you are comfortable with the content of your poster and discuss any concerns with your tutor. Then, attend the scheduled technical demo session. Please ensure that you print your poster ahead of time and bring it with you. It should be printed on A3 paper and in portrait. This is your individual responsbility. Please, also ensure that one member of the team is able to setup a laptop with the production prototype ahead of time.

You will receive **formal feedback** three weeks after the final deadline.

Additional Guidance

Carefully select the algorithm that you will take ownership of and implement. This algorithm will need to interface with other game components, and therefore affected by work of your peers. Aim for high cohesion and low coupling! It is also important that the main requirements are firmly specified and are not too broad. This will ensure that there is little overlap with the work of your peers and help ensure that you do not overburden yourself with too much work.

Please remember to commit frequently and to push your source code and related assets to the GitHub repository. This will make it easier for you to maintain a backup of your work. It will also help you to measure your productivity. GitHub should be an essential part of your workflow, not merely a place to submit your work for feedback. You will be expected to maintain an archive of playable builds to demo your work at any time.

Poor planning and poor time management can have a significant impact on this assignment. As some of you may have already discovered, programming is quite unlike other subjects in that it cannot be "crammed" into a last minute deluge. Sustain a steady pace across the duration of the course. Do a little programming every day, if you can!

For the most part, your work will be marked as a group effort. However we want to avoid the situation where students try to "coast" through the assignment on their fellow group members' work, and equally the situation where one member of the group takes the lion's share of the work and prevents the others from contributing effectively. Marks will be weighted by a multiplier for **individual contribution**, which aims to penalise both of these behaviours. We assess this by several means, including but not limited to: sprint reviews; individual vivas; feedback from your peers; attribution in the source code; and

GitHub commit logs. Any student who has contributed their *fair share* of effort to the project will receive a fair % for their effort, so any student who is putting in the appropriate level of effort has no need to worry. Note that effort is not the same as productivity.

Your code will be assessed on **functional coherence**: how well the finished game corresponds to the user stories, and whether the game has any obvious bugs. Correspondence to user stories runs both ways: implementing features that were not present in the design ("feature creep") is just as bad as neglecting to implement features.

Your code will also be assessed on **sophistication**. To succeed on a project of this size and complexity, you will need to make use of appropriate algorithms, data structures, libraries, and object oriented programming concepts. Appropriateness to the task at hand is key: you will **not** receive credit for complexity where something simpler would have sufficed.

Maintainability is important in all programming projects, but doubly so when working in a team. Use **comments** liberally to improve code comprehension, and carefully choose the **names** for your files, classes, functions and variables. Use a well-established commenting convention for **high-level documentation**. The open-source tool Doxygen supports several such conventions. Also ensure that all code corresponds to a sensible and consistent **formatting style**: indentation, whitespace, placement of curly braces, etc. Hard-coded **literals** (numbers and strings) within the source should be avoided, with values instead defined as constants together in a single place. Consider allowing some literal values, where appropriate, to be "tinkered" without changing the source code, e.g. by defining them in an external file read by the game on startup.

FAQ

- **What is the deadline for this assignment?**
  Falmouth University policy states that deadlines must only be specified on LearningSpace. Please examine the assignment area where you located this document.
- **What should I do to seek help?**
  You can email your tutor for informal clarifications. For informal feedback, make a pull request on GitHub.
- **Is this a mistake?**
  If you have discovered an issue with the brief itself, the source files are available at:
  `https://github.com/Falmouth-Games-Academy/bsc-assignment-briefs`.
  Please make a pull request and comment accordingly.

Additional Resources

- `https://inventwithpython.com/makinggames.pdf`
- `https://inventwithpython.com/inventwithpython_3rd.pdf`
  (Chapters 17-20)
- Keith, C. (2010) Agile Game Development with Scrum. Pearson Education.
- Sims, C. and Johnson, H.L. (2012) SCRUM: A Breathtakingly Brief and Agile Introduction. Dymaxicon.
- `https://www.mountaingoatsoftware.com/agile/user-stories`
- `https://travis-ci.org`

Marking Rubric (Production Prototype)

Criteria marked with a † are weighted by individual contribution to a shared deliverable. All other criteria are individual.

Criterion	Weight	Refer for Resubmission	Basic Competency	Basic Proficiency	Novice Competency	Novice Proficiency	Professional Competency
Sprint Reviews	40%	The student fails to participate in at least one sprint review.	The student participates in all sprint reviews. All sprint reviews result in a playable build.				
Appropriateness of User Stories and Sprint Plans	5% †	No user stories and/or sprint plans are provided.	Few user stories are distinguishable and easily measured. Sprint plans provide little support for the project.	Some user stories are distinguishable and easily measured. Sprint plans provide some support for the project.	Most user stories are distinguishable and easily measured. User stories correspond to the game design. Sprint plans provide much support for the project.	Nearly all user stories are distinguishable and easily measured. User stories clearly correspond to the game design. Sprint plans provide considerable support for the project.	All user stories are distinguishable and easily measured. User stories clearly and comprehensively correspond to the game design. Sprint plans provide significant support for the project.
Functional Coherence	5% †	No gameplay elements have been implemented and/or the code fails to compile or run.	Few gameplay elements have been implemented. There are many obvious and serious bugs.	Some gameplay elements have been implemented. There are some obvious bugs.	Many gameplay elements have been implemented. There is some evidence of feature creep. There are few obvious bugs.	Almost all gameplay elements have been implemented. There is little evidence of feature creep. There are some minor bugs.	All gameplay elements have been implemented. There is no evidence of feature creep. Bugs, if any, are purely cosmetic and/or superficial.
Sophistication	10% †	No insight into the appropriate use of programming constructs is evident from the source code. No attempt to structure the program is evident (e.g. one monolithic source file).	Little insight into the appropriate use of programming constructs is evident from the source code. The program structure is poor.	Some insight into the appropriate use of programming constructs is evident from the source code. The program structure is adequate.	Much insight into the appropriate use of programming constructs is evident from the source code. The program structure is appropriate.	Considerable insight into the appropriate use of programming constructs is evident from the source code. The program structure is effective. There is high cohesion and low coupling.	Significant insight into the appropriate use of programming constructs is evident from the source code. The program structure is very effective. There is high cohesion and low coupling.
Maintainability	20% †	There are no comments, or comments are misleading. Most variable names are unclear or inappropriate. Code formatting hinders readability.	The code is only sporadically commented, or comments are unclear. Few identifier names are clear or inappropriate.	The code is well commented. Some identifier names are descriptive and appropriate. An attempt has been made to adhere to a consistent formatting style. There is little obvious duplication of code or of literal values.	The code is reasonably well commented. Most identifier names are descriptive and appropriate. Most code adheres to the Google C++ formatting style. There is almost no obvious duplication of code or of literal values.	The code is reasonably well commented, with Doxygen-compatible module documentation. Almost all identifier names are descriptive and appropriate. Almost all code adheres to the Google C++ formatting style. There is no obvious duplication of code or of literal values. Some literal values can be easily "tinkered".	The code is very well commented, with comprehensive Doxygen-compatible module documentation. All identifier names are descriptive and appropriate. All code adheres to the Google C++ formatting style. There is no obvious duplication of code or of literal values. Most literal values are, where appropriate, easily "tinkered" outside of the source.
Portability and Navigability	5% †	Game will not execute at all on another machine for reasons related to code portability which cannot be fixed easily due to its poor structure. The provided template has not been followed.	There were challenges executing the game, but these were resolvable. The directory structure inside the submitted zip file is unclear. The provided template has not been followed.	Several portability issues are present. The directory structure inside the submitted zip file is somewhat confusing. The provided template has mostly been followed.	Some portability issues are present. The directory structure inside the submitted zip file is adequate. The provided template has been followed.	Few portability issues are present. The directory structure inside the submitted zip file is mostly sensible. The provided template has been followed.	Almost no portability issues are present. The directory structure inside the submitted zip file is sensible. The provided template has been followed.

Criterion	Weight	Refer for Resubmission	Basic Competency	Basic Proficiency	Novice Competency	Novice Proficiency	Professional Competency
Team Cohesion	5% †	The group's professional conduct has been unacceptable, and/or the group has failed to function at all as a team. Agile working practices have not been used.	The group has demonstrated little professionalism. Agile working practices have provided little support for the project.	The group has demonstrated some professionalism, functioning adequately as a team. Agile working practices have provided some support for the project.	The group has demonstrated much professionalism, functioning somewhat effectively as a team. Agile working practices have provided much support for the project.	The group has demonstrated considerable professionalism, functioning effectively as a cohesive team. Agile working practices have provided considerable support for the project. There is evidence of some use of Travis CI to support a continuous integration approach.	The group has demonstrated significant professionalism, functioning highly effectively as a cohesive team. Agile working practices have provided significant support for the project. Travis CI has been used to effectively support a continuous integration approach.
Use of Version Control	10%	GitHub has not been used.	Material has been checked into GitHub less frequently than once per sprint. All code has been checked into the Master branch.	Code has been checked into GitHub at least once per sprint. An attempt has been made to use branches.	Code has been checked into GitHub several times per sprint. Commit messages are clear, concise and relevant. Branches are used sensibly.	Code has been checked into GitHub several times per sprint. Commit messages are clear, concise and relevant. Branches are used somewhat effectively. There is evidence of engagement with peers (e.g. code review).	Code has been checked into GitHub several times per sprint. Commit messages are clear, concise and relevant. Branches are used effectively. There is significant evidence of engagement with peers (e.g. code review).
Individual Contribution	Multiplier for criteria marked †	The student has failed to contribute their "fair share" to the project, or has actively prevented others from doing so.					The student has contributed their "fair share" to the project, and has facilitated others in doing so.

Marking Rubric (Technical Demo)

Criteria marked with a ‡ are shared by the group. All other criteria are individual.

Criterion	Weight	Refer for Resubmission	Basic Competency	Basic Proficiency	Novice Competency	Novice Proficiency	Professional Competency
Basic Competency Threshold	40%	No individual poster and/or production prototype is delivered, or either are inappropriate.	A broadly appropriate individual poster and group tech demo are delivered in a timely fashion. There is no evidence of academic misconduct.				
Communication Skills	20% ‡	Delivered with no enthusiasm. The technology behind the game has not been articulated with clarity.	Delivered with little enthusiasm. Little connection with the audience. The technology behind the game has been articulated with little clarity.	Delivered with some enthusiasm, conveying technical confidence. Some connection with the audience. The technology behind the game has been articulated with some clarity.	Delivered with much enthusiasm, conveying technical confidence. Much connection with the audience. The technology behind the game has been articulated with much clarity.	Delivered with considerable enthusiasm,conveying technical confidence. Considerable connection with the audience. The technology behind the game has been articulated with considerable clarity.	Delivered with significant enthusiasm, conveying technical confidence and passion. Significant connection with the audience. The technology behind the game has been articulated with significant clarity.
Poster Quality	10%	There is no poster or it does not describe the engineering of the software.	The engineering of the software (e.g., class designs) is described with little adequacy.	The engineering of the software (e.g., class designs) is described with some adequacy.	The engineering of the software (e.g., class designs) is concisely described with much adequacy. The use of UML diagrams and source code excerpts is somewhat effective.	The engineering of the software (e.g., class designs) is concisely described with considerable adequacy. The use of UML diagrams and source code excerpts is quite effective.	The engineering of the software (e.g., class designs) is concisely described with significant adequacy. The use of UML diagrams and source code excerpts is very effective.
Technical Insight	10%	No individual algorithm has been contributed or it is trivial.	Little insight into the technical qualities of the individual algorithm. Little ability to explain how the algorithm fits into the game's components and architecture.	Some insight into the technical qualities of the individual algorithm. Some ability to explain how the algorithm fits into the game's components and architecture.	Much insight into the technical qualities of the individual algorithm. Much ability to explain how the algorithm fits into the game's components and architecture. The relevance of the contribution is justified.	Considerable insight into the technical qualities of the individual algorithm. Considerable ability to explain how the algorithm fits into the game's components and architecture. The relevance and value of the individual algorithm are justified.	Significant insight into the technical qualities of the individual algorithm. Significant ability to explain how the algorithm fits into the game's components and architecture. The relevance and value of the individual algorithm are justified. The individual algorithm is somewhat important to the design of the game.
Demo Quality	20% ‡	There is no demo, or it is non-functional.	The demo demonstrates some mechanics and interfaces.	The demo demonstrates the key mechanics and interfaces.	The demo demonstrates the core game mechanic. Although there may be a backup video, at least some aspect of the demo is live using the production prototype.	The demo demonstrates the core game mechanic. Although there may be a backup video, much of the demo is live using the production prototype.	The demo demonstrates the core game mechanic. Although there may be a backup video, a considerable part of the demo is live using the production prototype. There is some innovation in terms of technology in the demo.