GROUP DESKTOP GAME

Version 1.0
BSc Computing for Games

Dr Edward Powley

Introduction

In this assignment, you are required to design and implement a 2D game for Windows PC. Your game will be written in C++ using the Simple DirectMedia Layer (SDL) library. You will work in small groups (3–4 students) to design, plan and develop your game.

This assignment is formed of **four** parts: A, B, C, and D. Final summative submission is via LearningSpace.

A. *Pitch* your game design

As an **individual** effort, prepare and deliver a pitch for a game concept. You will be given creativity cards to aid in coming up with your concept. The only hard restriction on your concept is that it must use **2D graphics**, i.e. it must be implementable in SDL without the use of OpenGL or other 3D graphics libraries.

Your pitch should address the following questions:

- What is the title and high concept of the game?
- What is the intended aesthetic?
- What is the core mechanic?
- What makes the game fun?
- Is there a market for this type of game? Who is the target audience?
- What are the unique selling points?
- Is the scope appropriate for the time-frame of this assignment?

Prepare a **10-minute** presentation. Use appropriate aids to help communicate your design. This may or may not include g a ludic sketch and other visual aids. A formal presentation (e.g. PowerPoint slides) is **neither required nor recommended**, but is permitted.

Formative submission: Participate in the pitch session in class. The quality of your pitch is **not formally assessed**, however you will receive formative feedback from your tutors and peers. **Non-participation** will be penalised on a threshold basis.

In the session, the class will vote on which concepts will be taken forwards to parts B, C and D.

B. *Plan* the game

As a **group** effort, break your game concept down into user stories. Create a Trello task board and populate it with your user stories.

Formative submission: Discuss your task board with your tutor.

C. *Implement* the game

As a **group** effort, implement your assigned game concept. You will implement the game across **four** sprints; use your Trello board from part B to plan your sprints. Each sprint will end with a sprint review, where you should aim to have a playable build of your game.

One member of the group should fork the GitHub project at the following URL, and add the other members as contributors. Use this forked repository for all source code and other assets.

`https://github.com/Falmouth-Games-Academy/comp150-desktop-game`

Formative submission: Participate in the sprint review sessions.

D. *Demonstrate* the game

Bring your finished game to the demo session at the end of the semester, and be prepared to discuss it with tutors and peers.

Formative submission: Participate in the demo session. There is no formal submission for this part.

"A game is a series of interesting choices."

— Sid Meier

Summative submission

Create a zip file containing screenshots of your Trello task board from part B, and the source code and other assets for your game from part C. The recommended way do this is using the "Download Zip" button on the GitHub website. Upload the zip file to the appropriate submission queue on LearningSpace.

"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

You do **not** need to include a compiled executable in your submission. Please ensure that the markers will be able to compile and run your code. In particular you are advised to include any third-party libraries within your repository, or provide clear and comprehensive instructions on how to set up an appropriate build environment. If in doubt, please ask your tutor to verify that they can compile your code before you submit it.

All students must make an individual submission of the group's work. Late or non-submission will be subject to the usual penalties, even if the other members of your group have submitted on time.

Additional Guidance

A common mistake in this project is poor planning and poor time management. By now this will be a familiar phrase, but it is no less true of this project, and the usual guidance applies. It is particularly common for students to vastly underestimate the effort required to implement even a simple game, and thus vastly overscope their games. From the pitch stage, you should consider carefully what is feasible within the scope of this project.

For the most part, your work will be marked as a group effort. However it is important for us to avoid the situation where students are able to "coast" through the assignment, and receive undue credit for their fellow group members' work. It is also important to avoid the reverse situation, where one member of the group takes it upon themselves to do the lion's share of the work and prevent the others from contributing. Marks will be weighted by a multiplier for **individual contribution**, which aims to penalise both of these behaviours. We will assess this by several means, including 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 100% for this, so any student who is putting in the appropriate level of effort has no need to worry. Note that "fair share" is judged on effort rather than productivity, so for example do not worry that you cannot produce as much code as someone with more programming experience than you.

The first step in planning your implementation should be to break your concept down into **user stories**. Your user stories should be **distinguishable** (i.e. there should be little overlap between them) and **easily measured** (i.e. it should be easy to tell when each user story has been implemented). They should also be **comprehensive**, i.e. the user stories should completely capture the desired functionality of the game, with no gaps. Imagine giving your user stories to a developer who has never seen a game of your target genre. Would they be able to implement the game correctly, or would they miss key features?

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.

"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

"Hofstadter's Law:

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

— Douglas Hofstadter

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, library features, and object oriented programming concepts. Appropriateness to the task at hand is key, however: you will **not** receive credit for shoehorning a complicated solution into your program where a simpler one would have sufficed.

Maintainability is important in all programming projects, but doubly so when working in a team. Use **comments** liberally to improve comprehension of your code, and choose carefully the **names** for your files, classes, functions and variables. For high marks you should use a well-established commenting convention for **high-level documentation** of your files, classes and functions. The open-source tool Doxygen supports several such conventions, and can optionally be used to generate documentation from your source code in HTML and other formats. Also ensure that all code corresponds to a sensible and consistent **formatting style**: indentation, whitespace, placement of curly braces, etc.

As with all assignments on this course, you are expected to display a level of **innovation and creative flair** befitting Falmouth University's reputation as a world-leading arts institution. We are particularly looking for creativity in the design of your **game mechanics**; your game will **not** be judged on aesthetic criteria such as the quality of your art assets. One approach to promoting creativity is **divergent thinking**: generation of ideas by exploring many possible solutions. Often the most interesting ideas are **subversive**: they deliberately go against accepted conventions or obvious solutions.

Additional Resources

- Mitchell, S. (2013) SDL Game Development. Packt Publishing.
- 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`
- `http://www.doxygen.org`

Marking Rubric

Criterion	Weight	F (0 – 39)	D (40 – 49)	C (50 – 59)	B (60 – 69)	A (70 – 79)	A* (80 – 100)
Individual pitch	Threshold 5%	No individual pitch is delivered, or the pitch is inappropriate.					An appropriate individual pitch is delivered.
Sprint reviews	Threshold 2.5% + 2.5%	The student fails to participate in at least one sprint review.		The student participates in all four sprint reviews. At least one sprint does not result in a playable build.		The student participates in all four sprint reviews. A playable build is delivered at the end of every sprint.	
Appropriateness of user stories and sprint planning	10% †	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 effective 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 exemplary support for the project.
Functional coherence	10% †	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	20% †	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.	Significant 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.	Exemplary 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. Some identifier names are unclear or inappropriate. Code formatting is inconsistent or does not aid readability.	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 a consistent formatting style. There is almost no obvious duplication of code or of literal values.	The code is reasonably well commented, with some Doxygen-compatible module documentation. Almost all identifier names are descriptive and appropriate. Almost all code adheres to a consistent 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 a consistent 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.
Innovation and creative flair	10% †	No innovation and/or creativity. The game concept is a clone of existing works with only cosmetic alterations.	Little innovation and/or creativity. The game concept is derivative of existing works, with only minor gameplay alterations.	Some innovation and/or creativity. The game concept is derivative of existing works, but shows emerging divergent and/or subversive thinking in terms of gameplay.	Much innovation and/or creativity. The game concept is somewhat original, with an attempt at divergent and/or subversive thinking in terms of gameplay. The gameplay shows promise of fun and engagement.	Significant innovation and/or creativity. The game concept is original, with evidence of divergent and/or subversive thinking in terms of gameplay. The gameplay is somewhat fun and engaging.	Exemplary innovation and/or creativity. The game concept is highly original, with strong evidence of divergent and/or subversive thinking in terms of gameplay. The gameplay is fun and engaging.

Criterion	Weight	F (0 – 39)	D (40 – 49)	C (50 – 59)	B (60 – 69)	A (70 – 79)	A* (80 – 100)
Portability and Project Structure	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 provided template has not been followed.	Several portability issues are present. The provided template has mostly been followed.	Some portability issues are present. The provided template has mostly been followed exactly.	Few portability issues are present. The provided template has mostly been followed exactly.	Almost no portability issues are present. The provided template has mostly been followed exactly.
Professionalism	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 an emerging level of professionalism. Agile working practices have provided little support for the project.	The group has demonstrated a progressing level of professionalism, functioning adequately as a team. Agile working practices have provided some support for the project.	The group has demonstrated an appropriate level of professionalism, functioning somewhat effectively as a team. Agile working practices have provided much support for the project.	The group has demonstrated a high level of professionalism, functioning effectively as a cohesive team. Agile working practices have provided significant support for the project.	The group has demonstrated an exemplary level of professionalism, functioning highly effectively as a cohesive team. Agile working practices have provided exemplary support for the project.
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.	All group members have checked material into GitHub at least once per sprint. An attempt has been made to use branches.	All group members have checked material into GitHub several times per sprint. Commit messages are clear, concise and relevant. Branches are used sensibly.	All group members have checked material 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).	All group members have checked material 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.