PORTFOLIO OF GAME ENGINE COMPONENTS — GRAPHICS

Version 1.0
BSc Computing for Games
COMP220

Dr Ed Powley

Introduction

In this assignment, you are required to **design** and **implement** a C++ program using SDL and OpenGL which demonstrates the type of 3D computer graphics techniques that appear in a modern game engine.

Graphics technology is one of the most obvious areas in which innovation has driven gaming technology in recent years. Modern gaming PCs and consoles contain powerful graphics processing units (GPUs), and gamers expect modern games to push this hardware to its full potential. In this assignment you will practice the use of advanced graphical effects. Your final product will be a portfolio piece, which you can use in future to demonstrate your mastery of these techniques.

"Because of the nature of Moore's law, anything that an extremely clever graphics programmer can do at one point can be replicated by a merely competent programmer some number of years later."

— John Carmack

"Currently computer graphics are used a great deal, but it can be excessive."

— Hayao Miyazaki

This assignment is formed of several parts:

(A) **Write** a 2-page handout that will:
  (i) **outline** the concept of your demo;
  (ii) **explain** how your demo satisfies the requirements of the contract (provided as an appendix at the end of this document);
  (iii) **describe** at least **two** graphical or simulation effects your demo will include;

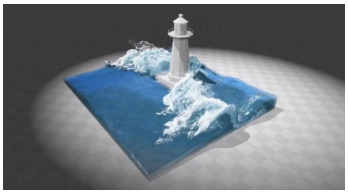
(B) **Populate** a Trello board with the user stories for your demo.

(C) **Implement**, over a series of **three** sprints, draft versions of your program that will:
  (i) **address** the requirements in the contract;
  (ii) **implement** at least **two** algorithms for graphical or simulation effects.

(D) **Implement** a final version of your program, which will:
  (i) **revise** any issues raised by your tutor and/or your peers.

(E) **Present** a practical demo of the computer program to your tutor that will:
  (i) **demonstrate** your academic integrity;
  (ii) **demonstrate** your individual programming knowledge and communication skills.

Assignment Setup



A demo of fluid simulation with NVIDIA's PhysX. Recent advances in GPU technology have enabled a wide range of high-fidelity real-time rendering and simulation effects.

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

`https://github.com/Falmouth-Games-Academy/comp220-portfolio`

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 **individual** and will be assessed on a **threshold** basis. Answer the following questions to pass:

- What is the title and high concept of the demo?
- What is the intended aesthetic?
- For each of two graphical or simulation effects:
  - What is the effect?
  - How will the effect contribute to the aesthetic?
- Is the scope appropriate for the product development time-frame?

To complete Part A, prepare the handout using any word processing tool. To help illustrate your intended aesthetic, your handout may include images and/or links to online videos. In addition, create a Trello board and populate it with your user stories.

Show the handout to your **tutor** for immediate **informal feedback**.

Part B

Part B consists of a **single formative submission**. This work is **individual** and will be assessed on a **threshold** basis. Answer the following questions to pass:

- What are the user stories for your demo?
- Which user stories are required for a minimum viable product, and which are stretch goals?
- Is the scope appropriate for the product development time-frame?

To complete Part B, populate a Trello board with your user stories.

Show the Trello board to your **tutor** for immediate **informal feedback**.

Part C

Part C is a **single formative 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) Enough work is available to conduct a meaningful review;
(c) A broadly appropriate review of a peer's work is submitted.

To complete Part C, prepare draft versions of the computer programs. Ensure that the source code and related assets are pushed to GitHub and a pull request is made prior to the scheduled sprint review sessions. Then, attend the scheduled sprint review sessions.

Part D

Part D is a **single summative submission**. This work is **individual** and will be assessed on a **criterion-referenced** basis. Please refer to the marking rubric at the end of this document for further detail.

To complete Part D, revise the computer program based on the feedback you have received. Then, upload it to the LearningSpace. Please note, the LearningSpace will only accept a single `.zip` file.

You will receive **formal feedback** from your **tutor** three weeks after the final submission deadline.

Part E

Part E 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) Enough work is available to hold a meaningful discussion;
(b) Clear evidence of programming knowledge **and** communication skills;
(c) No breaches of academic integrity.

To complete Part E, prepare a practical demonstration of the computer program. Ensure that the source code and related assets are pushed to GitHub and a pull request is made prior to the scheduled viva session. Then, attend the scheduled viva session.

Additional Guidance

As always, avoid underestimating the effort required to implement even simple software; always consider scope. From the proposal stage, you should consider very carefully what is feasible.

Your code will be assessed on **functional coherence**: how well the finished product corresponds to the user stories, and whether it 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.

Unlike your previous assignments, you will be assessed on the **performance** of your solution. Real-time graphics and simulation are not just about creating aesthetically pleasing effects, but doing so whilst maintaining a smooth and consistent framerate free of any lag or glitches that might frustrate the player. It may be necessary to trade-off the complexity or fidelity of an effect in order to achieve acceptable performance.

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 at startup.

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. One approach to promoting creativity is **divergent thinking**: generating ideas by exploring many possible solutions. Often the most interesting ideas are **subversive**: they deliberately go against convention or obvious solutions.

You will **not** be judged on the quality of your art assets. It is fine to use meshes and textures found online, as long as they are available under an appropriate license and are properly attributed.

FAQ

- **What is the deadline for this assignment?**
  Falmouth University policy states that deadlines must only be specified on the MyFalmouth system.
- **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

- `http://www.opengl-tutorial.org`
- `http://gamedev.stackexchange.com/questions/32876/good-resources-for-learning-modern-opengl-3-0-or-later`
- `https://google.github.io/styleguide/cppguide.html`

Marking Rubric

Criterion	Weight	Refer for Resubmission	Basic Proficiency	Novice Competency	Novice Proficiency	Professional Competency	Professional Proficiency
Basic Competency Threshold	40%	At least one part, or at least one sprint review, is missing or is unsatisfactory.	Submission is timely. Enough work is available to hold a meaningful discussion. Clear evidence of programming knowledge and communication skills. No breaches of academic integrity. The student participates in all sprint reviews				
Appropriateness of User Stories and Sprint Plans	5%	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 product design. Sprint plans provide much support for the project.	Nearly all user stories are distinguishable and easily measured. User stories clearly correspond to the product 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 product design. Sprint plans provide significant support for the project.	All user stories are distinguishable and easily measured. User stories clearly and comprehensively correspond to the product design. Sprint plans provide extensive support for the project.
Functional Coherence	5%	Few user stories have been implemented and/or the code fails to compile or run. Many obvious and serious bugs are detected. The executable's performance on lab PCs is unacceptable.	Some user stories have been implemented. Some obvious bugs are detected. The executable has many performance issues on lab PCs.	Many user stories have been implemented. There is some evidence of feature creep. Few obvious bugs are detected. The executable has some performance issues on lab PCs.	Almost all user stories have been implemented. There is little evidence of feature creep. Some minor bugs are detected. The executable has very few performance issues on lab PCs.	All user stories have been implemented. There is almost no evidence of feature creep. Some bugs, purely cosmetic and/or superficial in nature, are detected. The executable has almost no performance issues on lab PCs. Some consideration for graceful degradation on less powerful hardware is evident.	All user stories have been implemented. There is no evidence of feature creep. Few to no bugs are detected. The executable has no performance issues on lab PCs. Much consideration for graceful degradation on less powerful hardware is evident.
Sophistication	10%	Little insight into the appropriate use of programming constructs is evident from the source code. The program structure is poor or non-existant.	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.	Extensive insight into the appropriate use of programming constructs is evident from the source code. The program structure is extremely effective. There is very high cohesion and very low coupling.

Criterion	Weight	Refer for Resubmission	Basic Proficiency	Novice Competency	Novice Proficiency	Professional Competency	Professional Proficiency
Maintainability	15%	The code is only sporadically commented, if at all, or comments are unclear. Few identifier names are clear or inappropriate. Code formatting hinders 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 the Google C++ formatting style. There is almost no obvious duplication of code or of literal values.	The code is reasonably well commented, with appropriate Doxygen-compatiable 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 appropriate Doxygen-compatiable 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.	The code is commented extremely well, with comprehensive appropriate Doxygen-compatiable documentation. All identifier names are descriptive and appropriate. All code adheres to the Google C++ formatting style. There is no duplication of code or of literal values. Nearly all literal values are, where appropriate, easily "tinkered" outside of the source.
Creative Flair	10%	Little or no creativity. The work is a clone of an existing work with mere cosmetic alterations. The work delivers little or no fun and/or engagement.	Some creativity. The work is derivative of existing works, with only minor alterations. The work delivers some fun and/or engagement.	Much creativity. The work is derivative of existing works, demonstrating little divergent and/or subversive thinking. The work delivers much fun and/or engagement.	Considerable creativity. The work is somewhat novel, demonstrating some divergent and/or subversive thinking. The work delivers considerable fun and/or engagement.	Significant creativity. The work is novel, demonstrating significant divergent and/or subversive thinking. The work delivers significant fun and/or engagement.	Extensive creativity. The work is highly original, with strong evidence of divergent and/or subversive thinking. The work delivers extensive fun and/or engagement.
Portability and Navigability	5%	Product will not execute at all on another machine, for reasons related to code portability, even if they are trivially resolvable. The directory structure inside the submitted zip file is unclear. Provided template has not been followed well, if at all.	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.	No portability issues are present. Appropriate cross-platform compatiability on Linux, Windows, and MacOS. The directory structure inside the submitted zip file is sensible. The provided template has been followed.
Use of Version Control	5%	Material has been checked into GitHub less frequently than once per sprint.	Code has been checked into GitHub at least once per sprint.	Code has been checked into GitHub several times per sprint. Commit messages are clear, concise and relevant. There is some 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. There is much 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. There is significant evidence of engagement with peers (e.g. code review).	Code has been checked into GitHub several times per week. Commit messages are clear, concise and relevant. There is extensive evidence of engagement with peers (e.g. code review).

Appendix: Contract

You must **design** and **implement** a technical demo making use of 3D graphics techniques. The demo must meet the following requirements:

- A scene containing **at least one** textured mesh **and at least one** light source.
- Standard **first-person movement controls**:
  - The player can use the mouse to look up/down and to rotate in place;
  - The player can use the WASD keys and/or the arrow keys to move around the environment;
  - Other controls (e.g. interact, jump, shoot) should be added **if and only if** they are required for your concept.
- At least **two** of the following graphics and simulation techniques:
  - Loading of meshes from a standard 3D object file format;
  - Procedural generation of complex meshes or terrain;
  - Rendering of semi-transparent materials;
  - Realistic rendering of rough surfaces (e.g. normal mapping);
  - Skeletal animation or deformation;
  - Collision detection or ray-casting;
  - Integration of a third-party physics engine (e.g. Bullet, PhysX);
  - Particle effects (e.g. fire, smoke, fluid);
  - An advanced real-time lighting effect (e.g. shadow casting, reflections, volumetric lighting, bloom);
  - Non-realistic rendering (e.g. cel shading);
  - Other advanced rendering or simulation techniques of your choice, subject to discussion with your tutor and demonstration that you have researched the feasibility of your chosen techniques.
- Some aspect intended to create **fun and/or engagement** for the user, such as:
  - Exploring an environment in a "walking simulator" style; or
  - Interacting with a simulated system; or
  - Achieving a **simple** gameplay objective (beware of overscoping!)