REFERRAL BRIEF PORTFOLIO OF GAME ENGINE COMPONENTS — GRAPHICS

Version 1.0
BSc Computing for Games
COMP220

Brian McDonald

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

For this assignment you have to carry out the following:

- Implement an application framework using SDL
- Initialise OpenGL
- Render multiple 3D Object to the screen
- Scene lit with at lease one light
- FPS style Camera which can be controlled by the user

You have to submit a zip file which contains your application source code and all assets to the learning space by **4pm on 26th of July 2018**

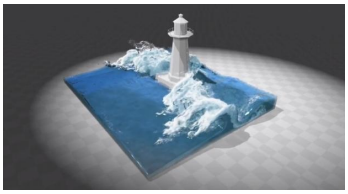
Assignment Setup

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`.



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.

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
Application Framework	10%	Empty application	Window displayed using SDL, Window and SDL shutdown correctly	Achieve Basic Proficiency, events such as close and input handled	Achieve Novice Competency, the initialisation of SDL and the Window is moved into functions	Achieve Novice Competency, A Game Application class is created to initialise SDL, Window and handle events	Achieve Professional Competency, the Game Application class serves as a base class for a 'Game Application'
OpenGL Initialisation	15%	OpenGL not initialised	OpenGL and GLEW initialised and destroyed correctly	Achieve Basic Competency, the initialisation of OpenGL is handled by a function in your application	Achieve Novice Competency, GLEW selects the best version of OpenGL that the graphics card supports	Achieve Novice Proficiency, OpenGL functionality moved into a Renderer class	Achieve Professional Competency, you have created an Interface class for a Renderer and the OpenGL Renderer inherits from this
Rendering of 3D Objects	20%	No objects renderer to the screen	One object rendered to the screen	Multiple objects rendered to the screen	A Game Object class (or similar) is created to render objects to the screen	Achieve Novice Competency and add a Component based system to the Game Object	Achieve Professional Competency and implement some Scene Management to the application
Lights	20%	No lights in scene	One Directional light is in the scene, which has a Diffuse component calculated. The scene also includes an Ambient component.	Achieve basic proficiency and the light has Specular component calculated	Achieve Novice Competency and implement a point light	Achieve Novice Proficiency and support multiple point lights	Achieve Professional Competency and support multiple point lights, one directional light and a sport light
FPS Style Camera	15%	No FPS Camera implemented	FPS Camera that can be controlled using WASD	Achieve Basic Competency and the mouse movement controls the viewpoint of the Camera	Achieve Novice Competency and move the camera functionality into a Camera class	Achieve Novice Proficiency, provide support for multiple cameras and the ability to switch viewpoints	Achieve Professional Competency and camera effects like head bob and screen shake
Maintainability	15%	There are no comments in the source code, or comments are misleading. Most variable names are unclear or inappropriate. Code formatting hinders readability.	The source 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 source code is somewhat well commented. Some identifier names are descriptive and appropriate. An attempt has been made to adhere to thhe PEP-8 formatting style. There is little obvious duplication of code or of literal values.	The source code is reasonably well commented. Most identifier names are descriptive and appropriate. Most code adheres to the PEP-8 formatting style. There is almost no obvious duplication of code or of literal values.	The source code is reasonably well commented, with Python doc-strings. Almost all identifier names are descriptive and appropriate. Almost all code adheres to the PEP-8 formatting style. There is no obvious duplication of code or of literal values. Some literal values can be easily "tinkered" in the source code.	The source code is very well commented, with Python doc-strings. All identifier names are descriptive and appropriate. All source code adheres to the PEP-8 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 code.
Use of Version Control	5%	GitHub has not been used.	Source code has rarely been checked into GitHub.	Source code has been checked into GitHub at least once per week. Commit messages are present. There is evidence of engagement with peers (e.g. code review).	Source code has been checked into GitHub several times per week. Commit messages are clear, concise and relevant. There is evidence of somewhat meaningful engagement with peers (e.g. code review).	Source code has been checked into GitHub several times per week. Commit messages are clear, concise and relevant. There is evidence of meaningful engagement with peers (e.g. code review).	Source code has been checked into GitHub several times per week. Commit messages are clear, concise and relevant. There is evidence of effective engagement with peers (e.g. code review).