CSC3224 Project 1

Welcome to the practical classes for CSC3224. In the context of this module, we are considering the design of engineering solutions to a given challenge, rather than the design of those challenges themselves – put another way, you are primarily assessed on the quality of your software engineering, rather than the gameplay experience. This is reflected in the weighting of the assessed coursework elements. You will first create a game engine suited to the deployment of a game designed to your own specification (**Project 1**); you will then submit the game itself for assessment (**Project 2**).

Module Assessment

There are 2 pieces of assessment for this module:

1. **Project 1 which is submitted to NESS. (75)**
2. Project 2 which is submitted to NESS. (25)

Project 1 (Deadline: 11:45pm, Wednesday 20th April, 2016)

Aims

Produce a game engine suited to deploying a game of your own design and specification. This engine is expected to employ middleware, and assessment is based upon the integration and, where appropriate, extension of that middleware. The engine itself will be data driven, either in terms of read-in files containing game information, or through a clearly demarcated procedural generation subsystem.

Deliverable Software Specification (Up to 75 Marks)

The game engine is expected to include, as a minimum, the following subsystems:

- Graphics – renders the game on screen
- Physics – manages movement and interface detections
- Resource Management – controls the loading and unloading of assets
- Human Interface – permits the user to actually interact with the game
- Audio – allows the engine to give audio cues and responses, and play background music
- Initialise/Shutdown – loads the game engine in a managed fashion, and closes cleanly
- Profiling – provides information regarding performance of individual subsystems
- File I/O – interprets data from either external data files (e.g. text files containing scripted instructions), or an independent Procedural Generation Subsystem, to construct environments ('levels') of the game

The submitted software will include complete functionality appropriate to the eventual game which will be deployed using the engine (**Project 2**). Again, it is expected that many of the above subsystems will employ middleware to underpin core functionality (e.g. the nclgl framework, SFML, OpenAL, etc.), and you are encouraged to make appropriate use of such middleware.

The submitted software will include a tutorial test scene (either a single interpreted file, or a mocked-up output from the Procedural Generation Subsystem) which takes a player through elementary controls of the game. For example, in a platform game, this might entail introducing the player to which keys control their avatar as they navigate a simple environment.

Marks are awarded for the overall submitted engine on the following bases:

- Integration of middleware within subsystems to which it is relevant
- Any extensions to such middleware made to suit the game design goals
- Structure of the engine
- Reliability of each subsystem (as demonstrated through the test scene)
- Complexity of each subsystem (as assessed through the code base and test scene)
- Comprehensive commenting of the code base

The submitted code must compile and execute on any machine. This means that libraries employed should be included in the submitted ZIP file, and relative paths linking those libraries must be employed. Relative paths should also be used for all **`#include`** statements. It is recommended that you test the contents of your ZIP file on a machine other than your own before submitting a final version to NESS.

Notes:

- If pursuing procedural generation, this subsystem **does not** need to be complete when **Project 1** is submitted. It is enough that you have a mocked-up output from such a system which your File I/O subsystem can then interpret to create the test scene.
- You are only marked for original code produced in this module; this includes code written to integrate existing libraries with your engine, but does not include code you have submitted for assessment in other modules or the code within third party libraries you have installed. For example, you are free to include as your physics subsystem the code you are developing for the CSC3222 coursework hand-in, but will be marked on the integration of that code with the engine, and any extensions you have made, and not on the 'duplicate' code.
- Where a file contains code which has not been written by you for this module, it should be stated at the top of that file in comments. Where a file contains a mixture of code written for this module, and other code, blocks of code written for this module should be preceded with **`// CSC3224 NCODE [Name] [Student Number]`** and end with **`// CSC3224 NCODE BLOCK ENDS`**.

Deliverables

Zipped Microsoft Visual Studio Project, Source Code and Executable (submitted via NESS)
Demonstration of Working Software (to take place on Thursday 21st April at 1PM in the Rack, or earlier by individual arrangement)

Mark Scheme

Submitted Software:	75
Total:	75