MODULE SPECIFICATION

Part 1: Information			
Module Title	Low-Level Programming for Games		
Module Code	UFCFXG-30-2	Level	Level 5
For implementation from	2019-20		
UWE Credit Rating	30	ECTS Credit Rating	15
Faculty	Faculty of Environment & Technology	Field	Computer Science and Creative Technologies
Department	FET Dept of Computer Sci & Creative Tech		
Module type:	Standard		
Pre-requisites	Entertainment Software Development 2019-20		
Excluded Combinations	None		
Co- requisites	None		
Module Entry requirements	None		

Part 2: Description

Educational Aims: See Learning Outcomes

Outline Syllabus: The syllabus includes:

C++ language features:

Rationale for using C++ in Games Development

Memory allocation / deallocation

Object orientation: inheritance and polymorphism

Templates

Operator overloading

Delegate functions

Function Pointers (functors)

Software development using C++ for non-trivial projects:

Runtime behaviour / call-stack behaviour

Compiler directions and representation of language features

“Advanced” debugging facilities (dump files, expressions, exception handling, memory examination and tracing)

Practical considerations – APIs, IDEs, libraries and SDKs
API / SDK evaluation

Plug-ins / interfacing with existing applications

Unmanaged code:

Automatic vs dynamic memory handling

Measuring and analysing performance

Memory alignment, bit manipulation, packing, pooling

Custom memory management

Threading and networking:

Threading: Concepts, libraries and implementation approaches

Networking: Concepts, libraries and implementation approaches

Data Structures:

Standard Template Library and its implementation of such structures

Implementation of standard data structures
Linked lists, vector, stack, etc.

The use of Design Patterns within C++, origins and implementation specifics of particular interest to games development.

Efficiency:

Big-O notation and limitations of its analysis

Teaching and Learning Methods: Contact Hours: 3 hours of Lectorials per week.

Lectorials will blend the introduction of relevant programming concepts with practical exploration guided by worksheets.

It is expected that a significant proportion of the worksheet and portfolio tasks will be carried out during the extended Lectorial sessions. Under observation and with discussion with the module team. This will thus form the controlled conditions for the assessment.

A subset of the worksheet tasks will build to be components of a small number of more challenging portfolio tasks to implement taught concepts. These will use supplied designs / code /

libraries / SDKs / APIs where appropriate. Any base code will be provided via a read only code repository.

It is expected that both worksheet and portfolio tasks will involve working in small groups (2-4 students) to reflect the often communal nature of games development. Whilst the portfolio tasks will be largely carried out in students' own time, taught sessions will provide a space for progress discussions and portfolio help where appropriate.

Part 3: Assessment

Formative assessment:
Worksheet tasks set for the module will be subject to extended in session peer and tutor-led discussion.

Completed tasks will contribute to the more involved portfolio tasks.

Summative assessment:
In addition to worksheet tasks, a small number of more challenging portfolio tasks will be set across the teaching year, to be completed individually or in small groups. These will be summative, though some formative feedback on early work will be available through discussion in taught sessions.

Each task will have a research element, with the expectation students will explore multiple techniques to complete the task and explain their choice of methodology.

The reason behind this strategy is to expose students to the production of code as a group activity (the principal method of games development), to align assessed tasks with the topics being taught, and distribute workload for the module across the year.

An individual logbook will detail student contributions to, and reflection on, each of the portfolio tasks.

Combined Formative and Summative Assessment:
It is expected that a significant proportion of the worksheet and portfolio tasks will be carried out during the extended Lectorial sessions. Under observation and with discussion with the module team. This will thus form the controlled conditions for the assessment.

Furthermore, assessment will also be in part via presentation of the code produced.

First Sit Components	**Final Assessment**	**Element weighting**	**Description**
Reflective Piece - Component A		25 %	Individual Logbook
Portfolio - Component A	✓	75 %	Portfolio of Practical Tasks
Resit Components	**Final Assessment**	**Element weighting**	**Description**
Reflective Piece - Component A		25 %	Individual Logbook
Portfolio - Component A	✓	75 %	Portfolio of practical tasks

Part 4: Teaching and Learning Methods

Learning Outcomes

On successful completion of this module students will achieve the following learning outcomes:

Module Learning Outcomes	**Reference**
Demonstrate implementations of standard data structures commonly used in games development, as well as an understanding of their implementation within the STL.	MO1
Implement simple threaded and networked applications that avoid typical race / synchronisation issues.	MO2
In relationship to the wider use of C++ and their role in cross-platform games development; outline the role and significance of external libraries, Application Programme Interfaces (APIs) and Software Development Kits (SDKs).	MO3
Recognise issues related to efficiency and organisation of memory resources within unmanaged code and apply strategies to reduce their impact on run-time performance.	MO4
Apply this understanding of memory management issues within C++, to develop object oriented applications which avoid issues such as memory leaks, pointer errors and undefined behaviour. Up to and including the production of custom memory management systems, instancing etc.	MO5
Design and implement object orientated applications that make appropriate use of mechanisms such as polymorphism, templates and delegate functions.	MO6
Analyse the impact of using various C++ language features on the compilation process and run-time behaviour of non-trivial games development projects.	MO7
Act as a reflective practitioner.	MO8

Contact Hours

Independent Study Hours:	
Independent study/self-guided study	148
Total Independent Study Hours:	148
Scheduled Learning and Teaching Hours:	
Face-to-face learning	72
Project work (individual or group)	80
Total Scheduled Learning and Teaching Hours:	152
Hours to be allocated	300
Allocated Hours	300

Reading List

The reading list for this module can be accessed via the following link:

https://uwe.rl.talis.com/modules/ufcfxg-30-2.html

Part 5: Contributes Towards
This module contributes towards the following programmes of study: Games Technology [Sep][SW][Frenchay][4yrs] BSc (Hons) 2018-19 Games Technology [Sep][FT][Frenchay][3yrs] BSc (Hons) 2018-19