MODULE SPECIFICATION

Part 1: Information			
Module Title	Games in C++		
Module Code	UFCFWA-30-1	Level	Level 4
For implementation from	2020-21		
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	None		
Excluded Combinations	None		
Co- requisites	None		
Module Entry requirements	None		

Part 2: Description

Overview: Introducing students to C++ software development and making games using the C++ language.

Educational Aims: The aim of this module is to introduce students to fundamental concepts underpinning computer games programming, using the C++ programming language and fundamental software development practices, problem solving techniques and mathematics which, together, will allow students to confidently write code that solves typical games programming problems.

Outline Syllabus: Introduction to the C++ programming language:
- Variables and operators
- Control structures and execution flow
- Functions
- Classes and object orientation
- IDEs and compilation / execution / debugging processes

Software development process:
- Problem solving with code / functional decomposition, from planning to implementation
- Object oriented decomposition and UML notation
- Coding style considerations and documentation practices

- Understanding of appropriate development techniques and processes to use when implementing game code

Mathematics:
- Set theory and logic: operators, truth tables, simple propositional / predicate logic.
- Computer arithmetic: binary and decimal representations.
- Algebra: basic manipulation, Cartesian coordinates, lines, curves and linear equations.
- Trigonometry, functions, tangents, and normals as applied to geometry.

Teaching and Learning Methods: The syllabus will be explored through a combination of lectures and studio sessions. Students will also be expected to learn independently and carry out directed study and/or reading outside of taught classes.

Lectures will introduce programming concepts whilst being practically explored within supervised studio sessions guided by tutorial tasks.

Students will be set tutorial tasks to form a lab logbook. In addition, coursework tasks will be set to explore further taught concepts. It is expected that the majority of this coursework will be carried out independently, outside of taught sessions, however there will be assessment specific sessions available to provide targeted help prior to hand-in.

Part 3: Assessment

A set number of the tutorial tasks are to be completed to form individual lab logbooks. The tutorial tasks set for the module will be peer and/or tutor reviewed regularly in studio/practical sessions. Completed tasks will contribute to a logbook, which forms part of the students' portfolios. While this logbook contributes to the summative assessment, it also allows the students formative feedback from tutors in studio sessions.

In addition to the tutorial tasks, coursework tasks will be set. These coursework tasks will also form part of the portfolio for the module, and will be set in order of increasing complexity. The reason behind this strategy is to align assessed tasks with the topics being taught, and distribute workload for the module across the year.

A final examination for the module will assess detailed understanding of taught material and assess students individual understanding of module content.

First Sit Components	**Final Assessment**	**Element weighting**	**Description**
Portfolio - Component B		75 %	Portfolio of practical exercises and lab logbook
Examination (Online) - Component A	✓	25 %	Online Examination (2 hours) 24 hour window
Resit Components	**Final Assessment**	**Element weighting**	**Description**
Portfolio - Component B		75 %	Portfolio of practical exercises and lab logbook
Examination (Online) - Component A	✓	25 %	Online Examination (2 hours) 24 hour window

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**

Write, compile and run high-level computer programs demonstrating appropriate use of the C++ language syntax	MO1
Utilise the debugging facilities of an IDE (Integrated Development Environment) to identify, analyse and resolve run-time errors	MO2
Use methodical processes to analyse and decompose typical games programming problems in order to design, implement and evaluate their algorithmic solutions	MO3
Employ software engineering techniques and associated notation to illustrate and interpret small-scale software designs	MO4
Apply fundamental mathematical concepts from algebra, trigonometry, computational arithmetic, logic and set theory, to solve games programming problems	MO5

Contact Hours

Independent Study Hours:	
Independent study/self-guided study	228
Total Independent Study Hours:	228
Scheduled Learning and Teaching Hours:	
Face-to-face learning	72
Total Scheduled Learning and Teaching Hours:	72
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/ufcfwa-30-1.html

Part 5: Contributes Towards

This module contributes towards the following programmes of study: