STUDENT AND ACADEMIC SERVICES

MODULE SPECIFICATION

Part 1: Information

Module Title	More Games in C++		
Module Code	UFCFXG-30-2	Level	Level 5
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		
Contributes towards	Games Technology {Foundation} [Sep][SW][Frenchay][5yrs] BSc (Hons) 2018-19 Games Technology {Foundation} [Sep][FT][Frenchay][4yrs] BSc (Hons) 2018-19		
Module type:	Standard		
Pre-requisites	Games in C++ 2020-21		
Excluded Combinations	None		
Co- requisites	None		
Module Entry requirements	None		

Part 2: Description

More Games in C++ is the follow-on module to Games in C++ in which students are to create a selection of more advanced games using the industry standard C++ programming language.

Educational Aims: The aim of this module is to build upon the foundations set out in the 1st year. The games developed in this module will be more advanced and encourage students to implement a wider range of technical solutions and gameplay mechanics, all whilst retaining creative control of their visions.

Outline Syllabus: The following outlines a typical syllabus applicable to this module, however, it should be noted that the syllabus may change in response to industry shifts and practices.

The module is designed to expose students to the following topics:

C++ Standard Library:
Rationale for using C++ in Games Development
Modern approaches to managing memory within a C++ code-base
The production of C++ standards compliant code
Development of games that can run across multiple platforms
Defensive code design and their appropriate programming techniques

Templates
Function and Class Templates
Smart Pointers
The C++ template utilities library

Software development:
Cross platform build practices
Professional practices and their application within the games industry
Practical considerations – APIs, IDEs, libraries and SDKs
Compiler directives and representation of language features

Object Orientation: composition vs inheritance
The use of Design Patterns within C++, origins and implementation specifics of particular interest to games development.

Threading and networking:
Threading: Concepts, libraries and implementation approaches
Networking: Concepts, libraries and implementation approaches

STL:
Standard Template Library and its implementation of data structures
Be able to identify the appropriate STL container for a given task
Correctly make use of the various algorithms included within the STL

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

Students should approach their sessions as studio time, engaging in the same practices used within the industry. These sessions are primarily a chance for students to work on their ongoing portfolios.

During this time a member of the module team will be available to provide guidance and support. When deemed necessary, topics may be introduced in a more formal manner within the allocated time slot. As programming is a practical subject matter that requires practice to obtain competency, students should endeavour to engage and attend theses sessions as well as look to make the most of the expert knowledge the team possesses.

Starter code will be provided for their games and any additional API's or frameworks will be introduced where appropriate. At times in order to allow more expansive game design and wider scope, students will be expected to work within small groups. This allows them to develop important communication skills and engage in professional practices that reflect the often communal nature of games development.

Part 3: Assessment
Formative assessment: As each game is produced, students will be graded accordingly and feedback provided. This attempts to utilise a continual feedback loop in which they can take on board constructive comments that can help evolve them into proficient C++ practitioners. These comments aim to provide positive reinforcement and should help to guide students in the correct direction. In addition to these formalised feedback points, the module team is present at every studio session and will engage with students to give formative feedback and guidance on their ongoing projects.

Summative assessment:
A number of challenging 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 sessions. Students will be expected to explore multiple programming techniques to complete their games 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 specific topics, and distribute workload for the module across the year.

An individual component will be used to detail student contributions to, and reflection on, each of the games worked on.

Combined Formative and Summative Assessment:
A significant proportion of the portfolio tasks will be carried out under controlled conditions. This will take place under observation and with discussion with the module team.

Resists:
Students will be provided a brief for a video game and will be expected to create a game that adheres to it. This will require both a design stage and implementation stage. Assessment will be carried out by reviewing their technical implementation and accompanying documentation. The work in this resit is individual and not group.

First Sit Components	**Final Assessment**	**Element weighting**	**Description**
Portfolio - Component A	✓	75 %	Portfolio of Games developed throughout the year.
In-class test - Component A		25 %	Throughout the year the student's will be expected to read around the subject and engage with the reading list. As a result in-class tests will be used to guage their understanding of common programming techniques and paradigms that are relevant to the module. The tests will not be focused directly on their portfolio work, but those skills being developled will also contribute to their understanding.
Resit Components	**Final Assessment**	**Element weighting**	**Description**
Practical Skills Assessment - Component A	✓	100 %	A typical game development brief will be handed out to the students and they will be need to design and implement a game based directly from the brief. Accompanying this will be a short reflective write-up on the game's development cycle.

Part 4: Teaching and Learning Methods

Learning Outcomes

On successful completion of this module students will be able to:

	Module Learning Outcomes
MO1	Be able to develop cross-platform games using recent C++ standards, best practices and appropriate APIs/Frameworks
MO2	Implement simple threaded and networked applications that avoid typical race / synchronisation issues.

	MO3	Utilise the STL and its related algorithms to produce stable and consistent functionality across differing development environments
	MO4	Design and implement object orientated applications that make appropriate use of mechanisms such as polymorphism and composition
	MO5	Act as a reflective practitioner and develop an awareness of the technologies used within the industry
Contact Hours	**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/ufcfxg-30-2.html	