Module Specification

More Games in C++

Version: 2021-22, v2.0, 12 Jul 2021

Contents

Part 1: Information

Module title: More Games in C++

Module code: UFCFXG-30-2

Level: Level 5

For implementation from: 2021-22

UWE credit rating: 30

ECTS credit rating: 15

Faculty: Faculty of Environment & Technology

Department: FET Dept of Computer Sci & Creative Tech

Partner institutions: None

Delivery locations: Frenchay Campus

Field: Computer Science and Creative Technologies

Module type: Standard

Pre-requisites: Games in C++ 2021-22

Excluded combinations: None

Co-requisites: None

Continuing professional development: No

Professional, statutory or regulatory body requirements: None

Part 2: Description

Overview: 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.

Features: Not applicable

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

Part 3: Teaching and learning methods

Teaching and learning methods: Contact Hours: 3 hours per week.

Students should approach their sessions as an opportunity to work with the module team and to ask questions. Whilst the sessions will be structured, the types of content delivered will be partially driven by the students. A typical session will involve discussion around the the topic of C++ game development accompanied with live coding and time assigned specifically for questions and answers. These sessions are primarily a chance for students to enhance their knowledge to aid them with work on their ongoing portfolios.

As programming is a practical subject matter that requires practice and understanding 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.

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

Hours to be allocated: 300

Contact hours:

Independent study/self-guided study = 228 hours

Face-to-face learning = 72 hours

Total = 300

Reading list: The reading list for this module can be accessed at readinglists.uwe.ac.uk via the following link https://uwe.rl.talis.com/modules/ufcfxg-30-2.html

Part 4: Assessment

Assessment strategy: Formative assessment:
Throughout the year, students will be able to reach out and discuss their code with the module team. Any comments given will be designed to provide positive reinforcement and should help to guide students in the correct direction.

Summative assessment:
A number of tasks will be set across the teaching year to be completed individually or in small groups (scope-dependent). These will be summative, though formative

feedback is available through in-class discussions and demo sessions. Students will be expected to explore multiple programming techniques to complete their games. The tasks given will aim to align with the specific topics covered throughout the year and directly relate to the learning outcomes.

Students will need to complete a Viva or equivalent (developer diary, video etc.) for each game task they work on. This will be used to confirm that their work is their own, but also to allow them the opportunity to reflect on what they've done. Typically a post-mortem is used for this purpose.

Resits:
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.

Assessment components:

Portfolio - Component A (First Sit)
Description: Portfolio
Weighting: 100 %
Final assessment: Yes
Group work: No
Learning outcomes tested: MO1, MO2, MO3, MO4, MO5

Practical Skills Assessment - Component A (Resit)
Description: Game with Evidence of Reflection
Weighting: 100 %
Final assessment: Yes
Group work: No
Learning outcomes tested: MO1, MO2, MO3, MO4, MO5

Part 5: Contributes towards

This module contributes towards the following programmes of study:

Games Technology [Sep][SW][Frenchay][4yrs] BSc (Hons) 2020-21

Games Technology [Sep][FT][Frenchay][3yrs] BSc (Hons) 2020-21

Games Technology {Foundation} [Sep][FT][Frenchay][4yrs] BSc (Hons) 2019-20

Games Technology {Foundation} [Sep][SW][Frenchay][5yrs] BSc (Hons) 2019-20