

Liverpool John Moores University

Title: DATA STRUCTURES AND ALGORITHMS FOR GAMES
Status: Definitive
Code: **5107COMP** (121229)
Version Start Date: 01-08-2021

Owning School/Faculty: Computer Science and Mathematics
Teaching School/Faculty: Computer Science and Mathematics

Team	Leader
Yann Savoye	Y

Academic Level: FHEQ5 **Credit Value:** 20 **Total Delivered Hours:** 56.5

Total Learning Hours: 200 **Private Study:** 143.5

Delivery Options

Course typically offered: Semester 1

Component	Contact Hours
Lecture	22
Practical	33

Grading Basis: 40 %

Assessment Details

Category	Short Description	Description	Weighting (%)	Exam Duration
Artefacts	AS1	Extending a Game Application Using Data Structures and Algorithms	60	
Exam	AS2	Examination	40	1.5

Aims

To study abstract data types (ADTs) and common implementations of these data types.

To gain an understanding of how a game application utilises both Parallel and Sequential Algorithms.

To develop applications using the Boost/STL C++ containers in order to implement games applications.

To understand the application of the data-oriented design and Object-Oriented Design paradigms on Data Structures.

To build on programming skills through implementation of algorithms and use of appropriate data structures in problem solving for games development.

To recognise and specify how complexity of operations on these ADTs and their overall performance characteristics are affected by both the ADT in question and its implementation strategy

Learning Outcomes

After completing the module the student should be able to:

- 1 Appraise a range of fundamental data structures and their operations.
- 2 Analyse and differentiate fundamental algorithms' (such as insert, remove, search, sort) time and space complexity characteristics as applied to a range of ADT implementations.
- 3 Evaluate Sequential vs Parallel Algorithm Strategies and their relationship to CPU / GPU architectures.
- 4 Apply appropriate data structures in order to solve specific sub problems related to game and game engine development.
- 5 Construct user-defined ADTs in C++ and utilise the Boost/STL APIs in order to store data related to both game mechanics and game engine architecture.
- 6 Develop software that utilises appropriate algorithms and data structures to satisfy a technical game design specification.

Learning Outcomes of Assessments

The assessment item list is assessed via the learning outcomes listed:

Extending Game Using DSA Examination	4	5	6
	1	2	3

Outline Syllabus

Algorithms for structure operations; insert, remove, retrieval

Algorithms for structure navigation; search and sort

Algorithm types: iterative and recursive

Relationship between ADTs and computing fundamentals (e.g. Stack, Queue)

Use of Big O notation to specify time complexity for simple algorithms

Debug / Profiling tools for Internal State Examination.

Debug / Profiling tools for algorithmic execution analysis (e.g. Frame Analysis)

Memory Management: Pointers, References, Heap, Stack.

STL and Boost: algorithms, iterators, and containers.

Introduction to Parallel Programming

- Goals of parallelism (e.g., throughput) versus concurrency (e.g., controlling access to shared resources)

- Parallelism, communication, and coordination

- Programming constructs for coordinating multiple simultaneous computations

- Need for synchronization
- Data races (simultaneous read/write or write/write of shared state)
- Higher-level races (interleavings violating program intention, undesired non-determinism)
- Lack of liveness/progress (deadlock, starvation)
- Game Specific Structures:
 - Scene Graphs
 - BSP/Quad Tree/kd-tree/OctTree
 - Frustums.
 - Linear Data Managers and Object Aliasing
 - Broadphase Collision Containers
 - Component-Based Game Objects.
 - Buffers and Queues – CPU vs GPU Representation.
 - BitSets and Enumerations
 - Enumerated Dictionaries.
 - Pipelines and Caches
 - Type Semantics, SISD vs SIMD Access Traits.
 - Review of physical memory and memory management hardware

Data Oriented Programming vs. Object Object Programming solutions for Data Structure Representation.

Brute Force/Greedy/Divide and Conquer

C++ Templates

C++ Class and Pointer Memory Layouts

C++ Specific Optimisation Strategies.

Sequential vs. parallel processing

Parallel programming vs. concurrent programming

Request parallelism vs. Task parallelism

Abstract Data Types and common implementation strategies:

- Linear ADTs:
 - Lists (Arrays, Linked Lists)
 - Stacks, Queues
- Non-Linear ADTs:
 - Trees, Binary Trees, BSTs
 - Maps (ListMaps, BSTMaps, HashMaps)
- Graphs
 - Undirected
 - Directed
 - Weighted
 - Graph Theory Fundamentals (Nodes, Edges etc).

Learning Activities

Lectures – to deliver the theoretical concepts on data structures and algorithms applied to computer games development.

Practical – Tutor-led practical session in the computer laboratory.

Further exercises – additional exercises for students to work on in their own time.

Directed learning – provides additional reading to enable practical work to be completed.

Learning materials can be accessed digitally via University Virtual Learning Environment (VLE).

Notes

This module is a technical, skills-focused module. Students will be introduced to Data Structures as Abstract Data Types and cover the fundamentals of discrete mathematics which provide the foundations for data structure design and implementation. Students will gain exposure of how to create their own implementation in C++ and the C++ Standard Template Library Implementations. Each structure will be covered from a games programming perspective, demonstrating real-world usage within game engines and game applications. The module will culminate with a look into parallel algorithms and how games apply parallelisation across CPU and GPU hardware using data structures.

It will require previous experience in programming; It will build on existing programming-based skills such as problem / functional decomposition and the use of an IDE to develop and test programs.