Politecnico di Torino

Ingegneria Informatica (Computer Engineering) LM-32(DM270)

A.a. 2025/2026

Graduation Session March 2026

Managing large hordes of NPCs in Unreal Engine 5

An analysis of Mass ECS system

Supervisors:
Francesco Strada
Corrado Raffaelli

Candidate:
Martina Plumari

Acknowledgements

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Tristique senectus et netus et malesuada fames ac turpis. Pretium nibh ipsum consequat nisl vel pretium lectus quam. Urna molestie at elementum eu facilisis sed odio morbi quis. Sed egestas egestas fringilla phasellus faucibus scelerisque eleifend. At in tellus integer feugiat. Mauris rhoncus aenean vel elit scelerisque mauris pellentesque pulvinar. Commodo sed egestas egestas fringilla. Nunc lobortis mattis aliquam faucibus purus in massa. Facilisis magna etiam tempor orci eu lobortis elementum nibh. Elementum curabitur vitae nunc sed velit dignissim. Neque volutpat ac tincidunt vitae. Massa id neque aliquam vestibulum morbi blandit cursus risus at. Porta non pulvinar neque laoreet suspendisse interdum consectetur. Turpis in eu mi bibendum. Ut tristique et egestas quis ipsum suspendisse. Integer quis auctor elit sed vulputate mi sit amet. Viverra nam libero justo laoreet sit amet cursus.

Table of Contents

List of Figures

Chapter 1

Introduzione

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Porttitor eget dolor morbi non arcu risus quis varius. Libero id faucibus nisl tincidunt. Neque laoreet suspendisse interdum consectetur libero id faucibus nisl tincidunt. Scelerisque in dictum non consectetur a erat. Leo a diam sollicitudin tempor id eu. Sodales ut eu sem integer vitae justo eget magna fermentum. A cras semper auctor neque vitae. Cursus euismod quis viverra nibh cras pulvinar. Mi tempus imperdiet nulla malesuada pellentesque elit eget gravida cum. Dictum at tempor commodo ullamcorper a lacus vestibulum sed. Ultricies mi eget mauris pharetra. In pellentesque massa placerat duis ultricies lacus. Fringilla phasellus faucibus scelerisque eleifend donec pretium vulputate.

Aliquam sem fringilla ut morbi tincidunt augue interdum. Risus at ultrices mi tempus imperdiet nulla malesuada pellentesque. Semper quis lectus nulla at volutpat. Nullam non nisi est sit amet facilisis. Eget velit aliquet sagittis id consectetur purus ut faucibus pulvinar. Ultricies tristique nulla aliquet enim tortor at auctor urna nunc. Pharetra diam sit amet nisl suscipit adipiscing bibendum est ultricies. Amet facilisis magna etiam tempor. Nunc non blandit massa enim nec dui nunc mattis. At ultrices mi tempus imperdiet nulla malesuada pellentesque. Cursus metus aliquam eleifend mi in nulla posuere. In eu mi bibendum neque egestas congue quisque. Augue eget arcu dictum varius duis at consectetur.

1.1 Goal

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Tristique senectus et netus et malesuada fames ac turpis. Pretium nibh ipsum consequat nisl vel pretium lectus quam. Urna molestie at elementum eu facilisis sed odio morbi quis. Sed egestas egestas fringilla phasellus faucibus scelerisque eleifend. At in tellus integer feugiat. Mauris rhoncus aenean vel elit scelerisque mauris pellentesque pulvinar. Commodo sed egestas egestas fringilla. Nunc lobortis mattis aliquam faucibus purus in massa. Facilisis magna etiam tempor orci eu lobortis elementum nibh. Elementum curabitur vitae nunc sed velit dignissim. Neque volutpat ac tincidunt vitae. Massa id neque aliquam vestibulum morbi blandit cursus risus at. Porta non pulvinar neque laoreet suspendisse interdum consectetur. Turpis in eu mi bibendum. Ut tristique et egestas quis ipsum suspendisse. Integer quis auctor elit sed vulputate mi sit amet. Viverra nam libero justo laoreet sit amet cursus.

1.2 Struttura della tesi

Lorem ipsum dolor sit amet:

- onsectetur adipiscing elit,
- sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
- Tristique senectus et netus et malesuada fames ac turpis.

Chapter 2

Mass System

Mass is Unreal Engine's data-oriented Entity-Component-System (ECS) framework designed for large-scale, high-performance simulation. It provides an alternative to Unreal's traditional object-oriented Actor model by representing simulation elements as lightweight entities made of plain data components, and by executing behavior through batch-oriented processing systems rather than per-object logic.

This chapter first analyses the Entity–Component–System architectural pattern and the reasons it can offer performance advantages over the traditional object-oriented paradigm. It then examines the structure and functioning of Unreal Engine's Mass system.

2.1 Entity Component System

The Entity Component system (ECS) is a software architectural pattern specifically engineered to facilitate high-performance simulations. Unlike traditional game development frameworks that rely on deep object hierarchies, ECS is built upon the Data-Oriented Design (DOD) paradigm.

At its core, ECS shifts the focus from "what an object is" to "what data it possesses". By decoupling data from behavior, it enables the construction of complex simulations using granular, reusable modules rather than monolithic, rigid classes.

2.1.1 Object-Oriented vs Data-Oriented Design

To understand why ECS has become the industry standard for mass-scaled simulations, we must compare it to the traditional Object-Oriented Design (OOD).

In OOD, functionality is typically extended through inheritance, making the resulting objects inextricably bound to their ancestors through a hard-coded relationship. As a project grows, this can lead to deep and rigid class hierarchies. For instance, when a Soldier class inherits from a Human class, it does not merely get access to human traits, but becomes physically coupled to the entire memory footprint and logic chain of the parent. On the other hand, DOD favors composition; in this case, rather than defining what an entity is (a Soldier), we define what it has (Transform, Health, Ammo). This approach allows us to avoid being forced to choose between two unrelated classes and enables architectural flexibility since the entity is simply a dynamic collection of data components assigned at runtime.

The primary driver for DOD is data locality. Modern CPU speeds vastly outpace memory access times; consequently, performance is often throttled by "memory latency", the delaay incurred when fetching data from RAM. In Object-Oriented Design, objects are frequently scattered throughout memory. Accessing them requires "pointer-chasing" through class hierarchies, leading to frequent cache misses and significant performance penalties in contemporary CPU architectures. On the other hand, in Data-Oriented Design components of an identical type are stored contiguously in memory. When a system processes these components the CPU can accurately predict the next data address and pre-load it into the cache. This maximizes cache coherency, facilitating rapid, linear iteration over thousands of entities.

2.1.2 The Three Pillars of ECS

As the name suggests, the ECS pattern consists of three main functional layers:

- Entity: it is a unique identifier serving as a handle that associates a set of components. It contains no intrinsic data or behavior. Its sole purpose is to provide a unique index to which stateful data can be mapped;
- Component: granular data container that encapsulate state but contain no logic. As "plain old data" structures, components are stored in dense arrays to optimize memory throughput;
- System: it represents the logic layer of the architecture. Systems are independent units that operate over filtered collections of components by means of declarative queries. Because systems operate on dense arrays with explicit data dependencies, they are inherently suited for multi-threaded execution, allowing the engine to distribute workloads across CPU cores with minimal synchronization overhead.

2.1.3 Trade-offs

While ECS provides the performance requisite for systems like UE Mass, it introduces specific architectural trade-offs:

- Architectural Complexity: initial planning requires a more rigorous design of component interfaces and system boundaries compared to the more intuitive Object based approach of OOD;
- Decoupling Overhead: the radical separation of data can lead to "component bloat" if the granularity is not managed carefully, potentially resulting in repetitive or trivial data structures.
- Mentality Shift: transitioning from an object-centric logic to Data-Oriented paradigm requires a fundamental shift in how developers conceptualize program flow and state management. This shift takes time, and the steep learning curve may not be strategically feasible for all production timelines or team structures.

2.2 Mass Overview

Lorem ipsum dolor sit amet: