Università degli Studi di Messina
Dipartimento di Scienze Matematiche e Informatiche,
Scienze Fisiche e Scienze della Terra
Corso di Laurea Triennale in Informatica

Analysis of entity encoding techniques, design and implementation of a multithreaded compile-time Entity-Component-System C++14 library

Laureando:
Vittorio Romeo

Relatore:
Prof. Giacomo Fiumara

Anno Accademico 2015/2016

Introduction

- Successful development of complex real-time applications and games requires a **flexible and efficient** entity management system
- It’s critical to find an elegant way to compose objects in order to:
  - Prevent code repetition *(DRY principle)*
  - Improve modularity, performance and flexibility
- The **Entity-Component-System** architectural pattern was designed to achieve the aforementioned benefits, by separating data from logic
- What is the Entity-Component-System pattern?
- How can it be implemented without sacrificing performance, safety and an high level of abstraction?

Outline

- **Entity encoding** techniques
  - Object-oriented inheritance
  - Object-oriented composition
  - Data-oriented composition
- Overview and implementation of **ECST**
  - Design and core values
  - Features/limitations
- Example case study: **particle simulation**

Entity encoding techniques

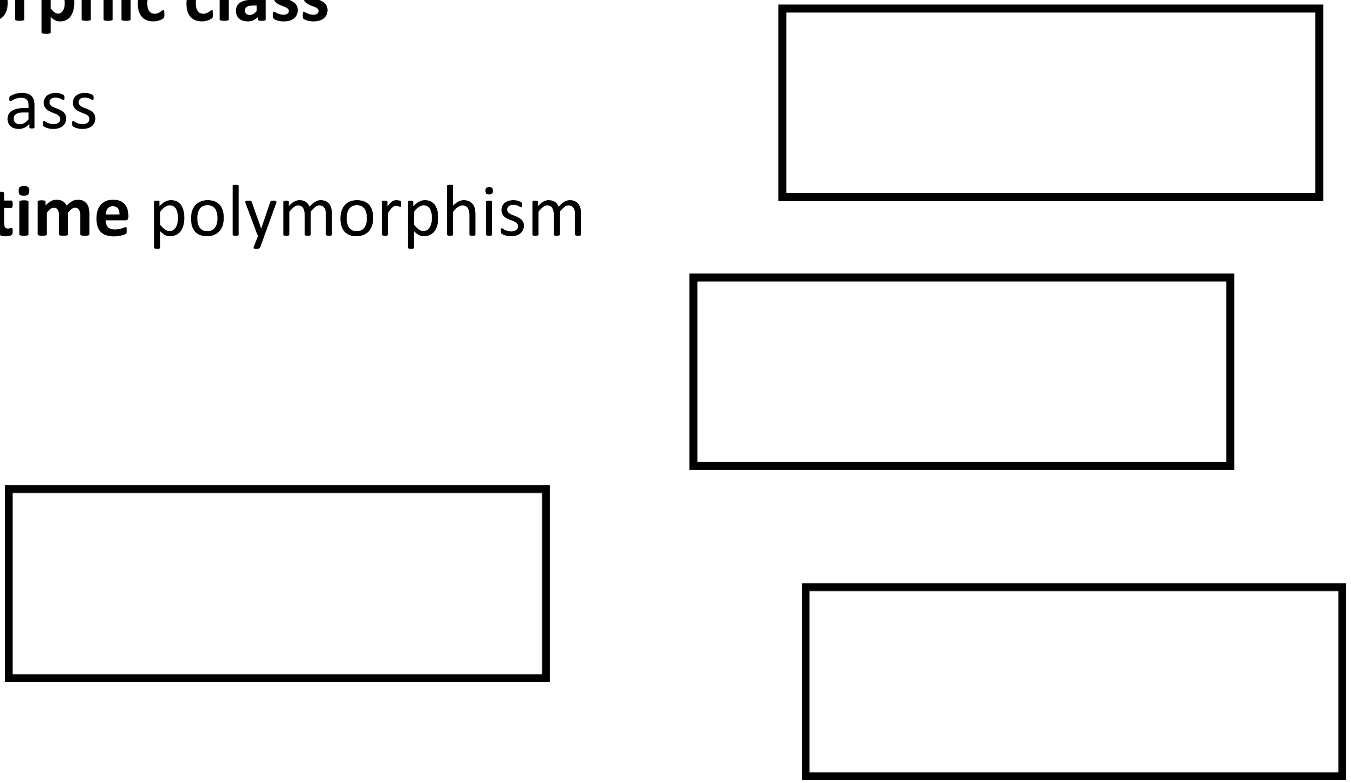
Concepts, benefits and drawbacks of entity encoding architectural patterns

What is an **entity**?

- Something **tied to a concept**
- Has related **data** and/or **logic**
- **Many instances** of the same entity type
    - **Specific instances** must be **trackable**
- Can be **accessed** and **modified**
    - Creation, destruction, mutation, reading
- Examples:
    - **Game objects:** *player, bullet, car*
    - **GUI widgets:** *window, textbox, button*

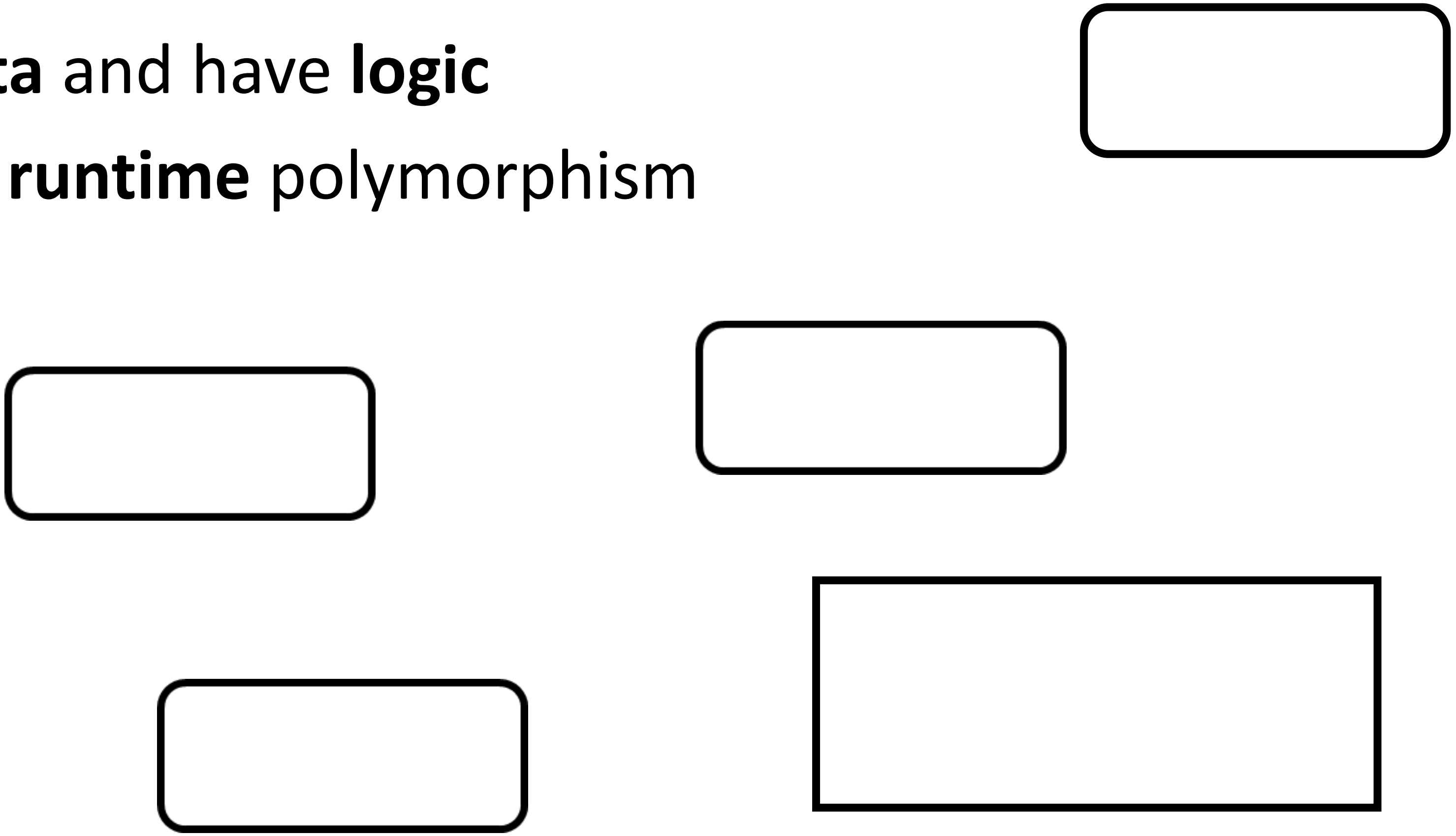
Encoding entities – **OOP inheritance**

- An **entity type** is a **polymorphic class**
- **Data** is stored inside the class
- **Logic** is handled using **runtime** polymorphism
- **Very easy** to implement
- **Cache-unfriendly**
- **Runtime overhead**
- **Lack of flexibility**



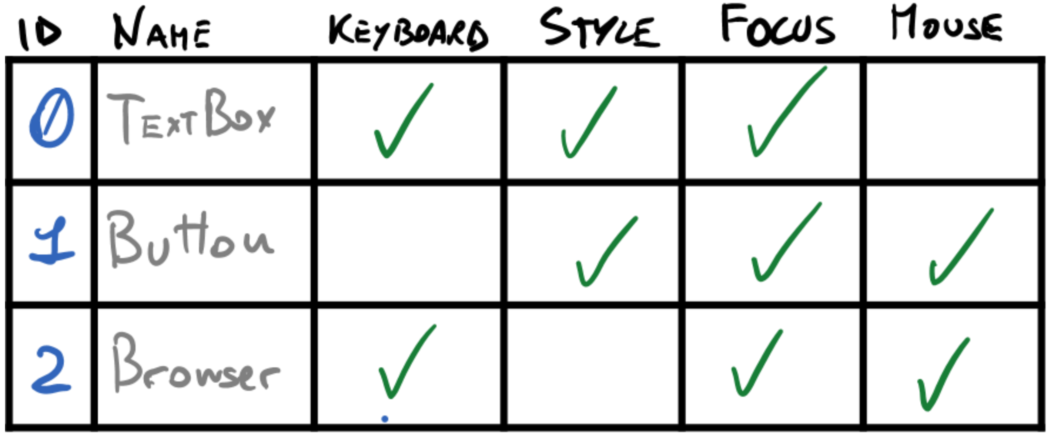
Encoding entities – **OOP composition**

- An **entity** is an aggregate of **components**
- Components **store data** and have **logic**
- **Logic** is handled using **runtime** polymorphism
- **Easy** to implement
- More **flexible**
- **Cache-unfriendly**
- **Runtime overhead**



Encoding entities – **DOD composition**

- An **entity** is a numerical **ID**
- Components only store **data** (logicless)
- **Logic** is handled using **systems**
- Potentially **cache-friendly**
- Minimal **runtime overhead**
- Great **flexibility**
- **Hard** to implement



ID	NAME	KEYBOARD	STYLE	FOCUS	MOUSE
0	TextBox	✓	✓	✓	
1	Button		✓	✓	✓
2	Browser	✓		✓	✓

Overview and implementation of **ECST**

Design, features, limitations and examples

Core values and concepts

- **«Compile-time»** ECS
- Customizable **settings/policy-based** design
- **Transparent** user syntax
- **Multithreading** support
  - Avoid explicit locking
  - **«Dataflow»** programming
- Modern **«type-value encoding»** metaprogramming
- Clean, modern and safe C++14

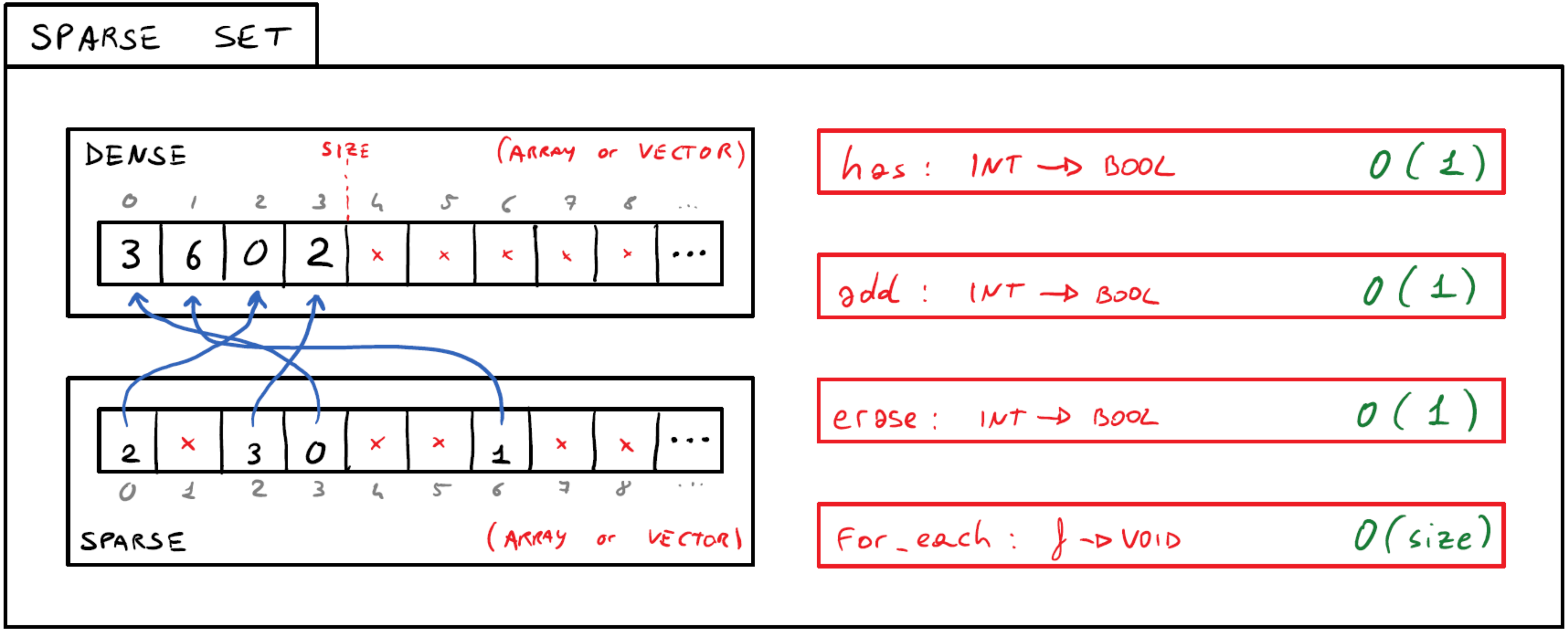
Multithreading support

- Enabled by default, customizable and can be optionally disabled
- Two levels of parallelism:
  - **Outer**: independent system chains can run in separate parallel tasks
  - **Inner**: system logic can be independently performed on entity subsets in separate tasks

Challenges

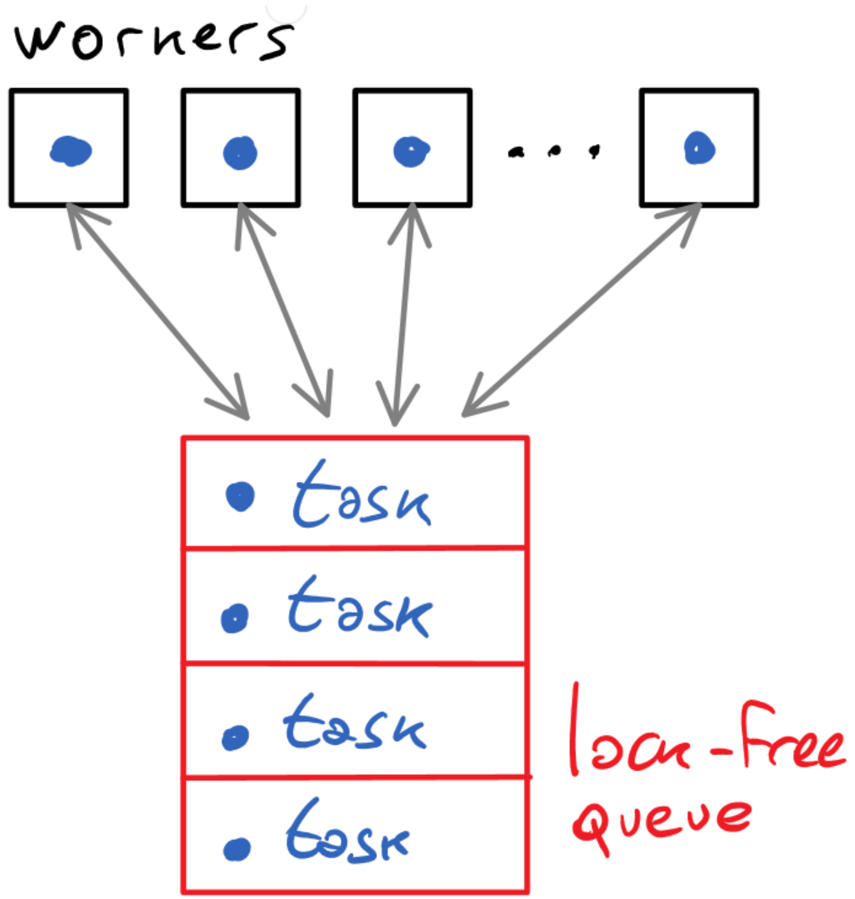
- Efficient management of **entity IDs**
- Exploiting **compile-time knowledge** to increase performance and safety
- Executing systems **respecting dependencies** between them and using **parallelism** when possible
- Processing entities subsets of the same system in different threads
- Dealing with entity/component addition/removal during system execution
- Providing a clean and safe interface to the user

Sparse Integer Sets



Multithreading details

- Avoid **busy waiting**, use **condition variables**
- Use **thread pooling** with a fast **lock-free concurrent queue**

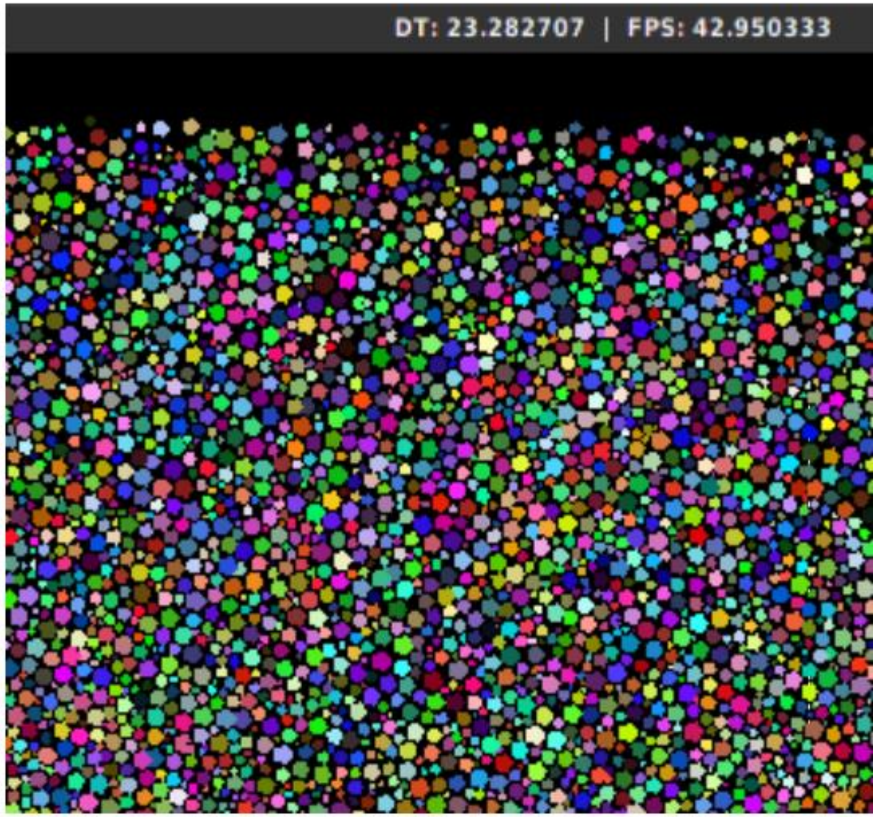


Example case study: **particle simulation**

Description, screenshots and benchmarks

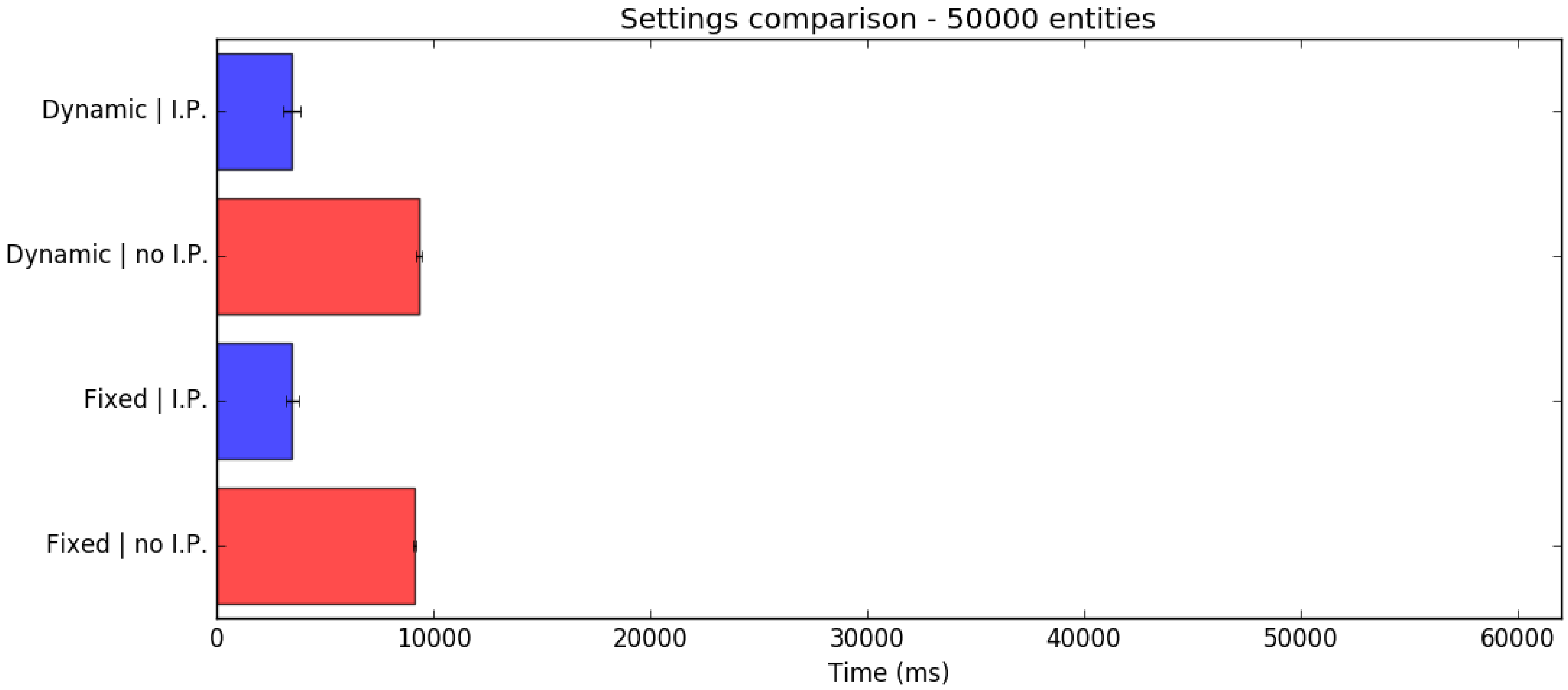
Overview

- Huge number of particles
- Circle versus circle collisions
- Completely elastic collisions
- 2D grid broad spatial partitioning
- Particles slowly decay
- 6 component types
- 9 system types

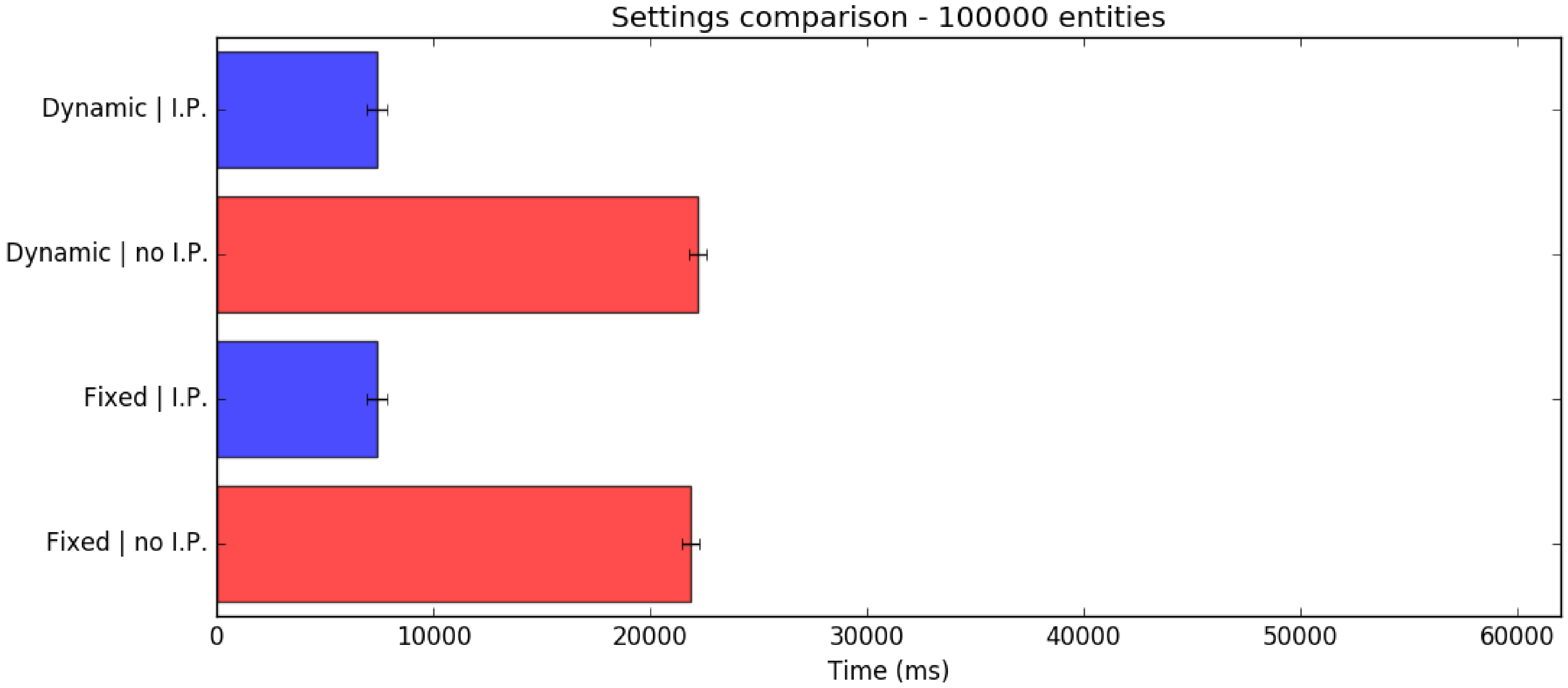


DT: 23.282707 | FPS: 42.950333

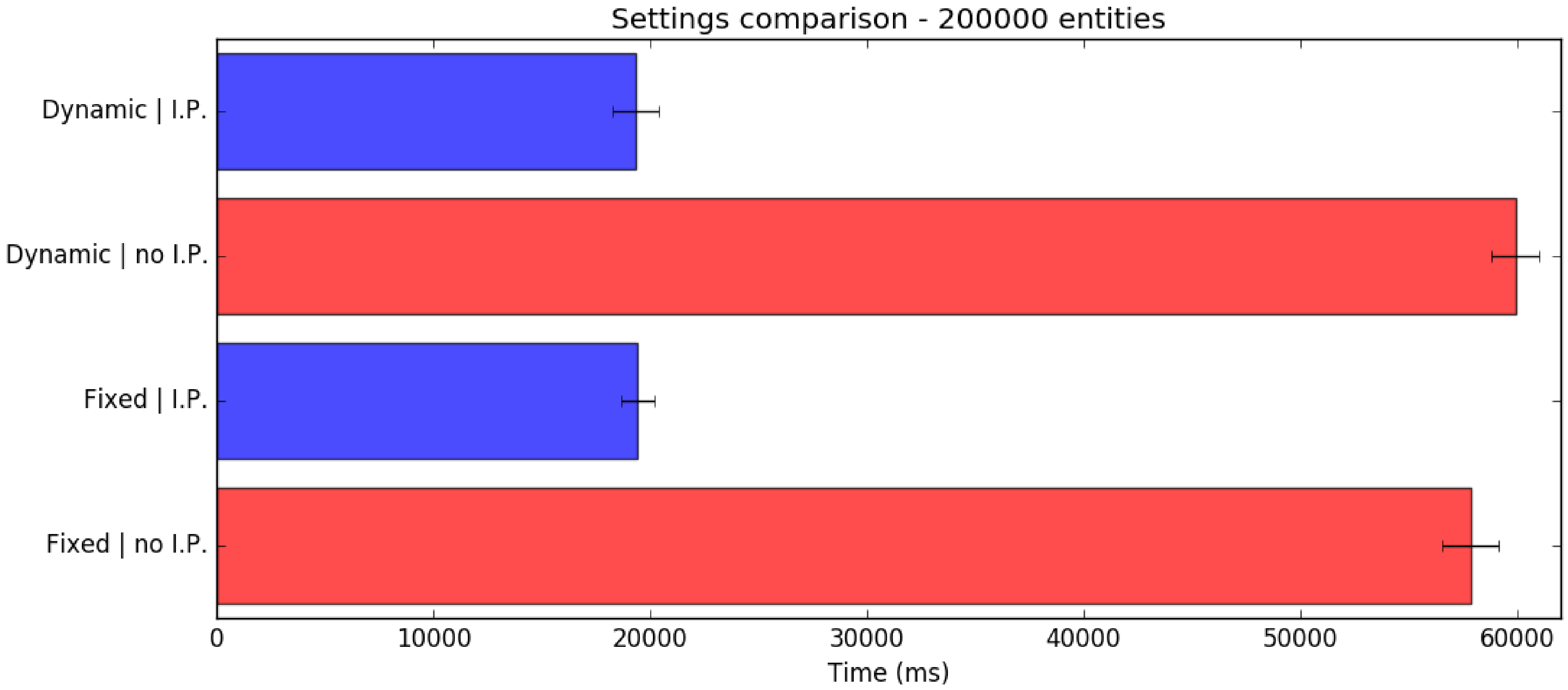
Results – 50000 entities



Results – 100000 entities



Results – 200000 entities



Suggested readings

- Open-source repositories:
  - ECST library: https://github.com/SuperV1234/ecst
  - Thesis PDF/sources: https://github.com/SuperV1234/bcs_thesis
  - C++Now 2016: https://github.com/SuperV1234/cppnow2016
- Other work:
  - Personal website: http://vittorioromeo.info/
  - YouTube channel: https://www.youtube.com/user/SuperVictorius
- Contacts:
  - vittorio.romeo@outlook.com