PCD Assignment 03 - Boids Simulator with Actors

A cura di

Alessandra Versari - alessandra.versari2@studio.unibo.it

Lorenzo Rigoni - lorenzo.rigoni2@studio.unibo.it

Riccardo Moretti - riccardo.moretti6@studio.unibo.it

Analisi del problema

In questo punto del terzo assignment, viene richiesto di implementare una versione concorrente della "*simulazione dei boid*" proposta da Craig Reynolds nel 1986. Nella simulazione, vengono create ***n*** entità chiamate "*boid*". Ogni *boid*, in un ciclo infinito, deve svolgere due azioni:

1. modificare la propria velocità in base ai pesi di separazione, allineamento e coesione;
2. modificare la propria posizione in base alla velocità calcolata precedentemente.

Oltre a ciò, durante la simulazione, l'utente può modificare i parametri dei boid (separazione, allineamento e coesione) e può sospendere/riprendere la simulazione, oltre a poterla fermare e avviarne una nuova.

Design, strategia ed architettura

Per risolvere il problema sopracitato, abbiamo deciso di usare un nuovo approccio di programmazione, la programmazione ad attori. In questo tipo di architettura, ogni entità chiamata *attore* ha un proprio stato e un proprio comportamento. In un sistema, gli attori comunicano tra loro scambiandosi messaggi, considerando che ogni attore ha la propria *mailbox* dove andrà a leggere i messaggi ricevuti.

L'implementazione è stata fatta in *Java* usando la libreria *Akka typed*.

Per la risoluzione del problema, abbiamo creato tre attori per il nostro sistema:

- ***GuiActor***
- ***BoidActor***
- ***SimulatorActor***

GuiActor è l'attore che interagisce con l'interfaccia grafica. Ha il compito di avvisare il *SimulatorActor* in caso di cambio dello stato di simulazione (*start*, *suspend*, *resume* e *stop*), di cambiare i parametri di simulazione (*alignment*, *cohesion* e *separation*) e di disegnare i boid aggiornati ricevuti da *SimulatorActor*.

BoidActor è l'attore che gestisce un sottogruppo di *boid* assegnato dal simulatore. Ha il compito di calcolare le nuove velocità e le nuove posizioni e restituirne il risultato.

SimulatorActor è l'attore che gestisce il ciclo di simulazione. Interagisce con gli altri due attori per iniziare la simulazione, creare i *BoidActor* per l'aggiornamento dei *boid*, fornirli al *GuiActor* per disegnarli e ricominciare i calcoli.

Per comunicare tra loro, dato che viene usata la versione *typed* di *Akka*, abbiamo creato la classe astratta ***Commands*** che contiene al suo interno tutti i tipi di messaggi che possono scambiarsi gli attori.

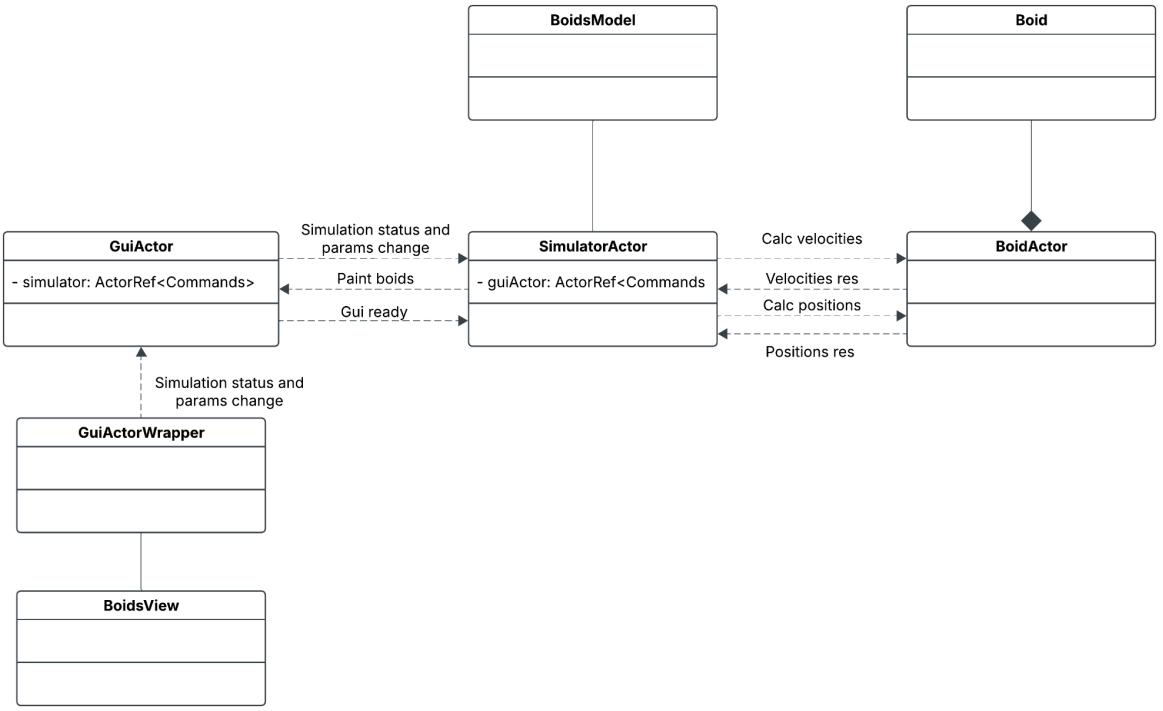

Figura 1: diagramma UML dell'architettura del sistema

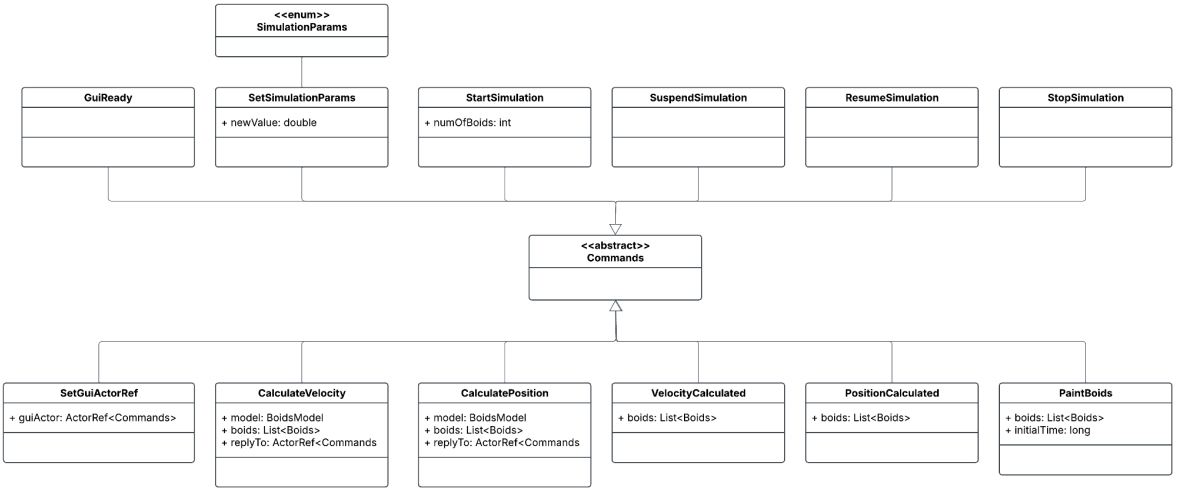

Figura 2: possibili messaggi che si scambiano gli attori

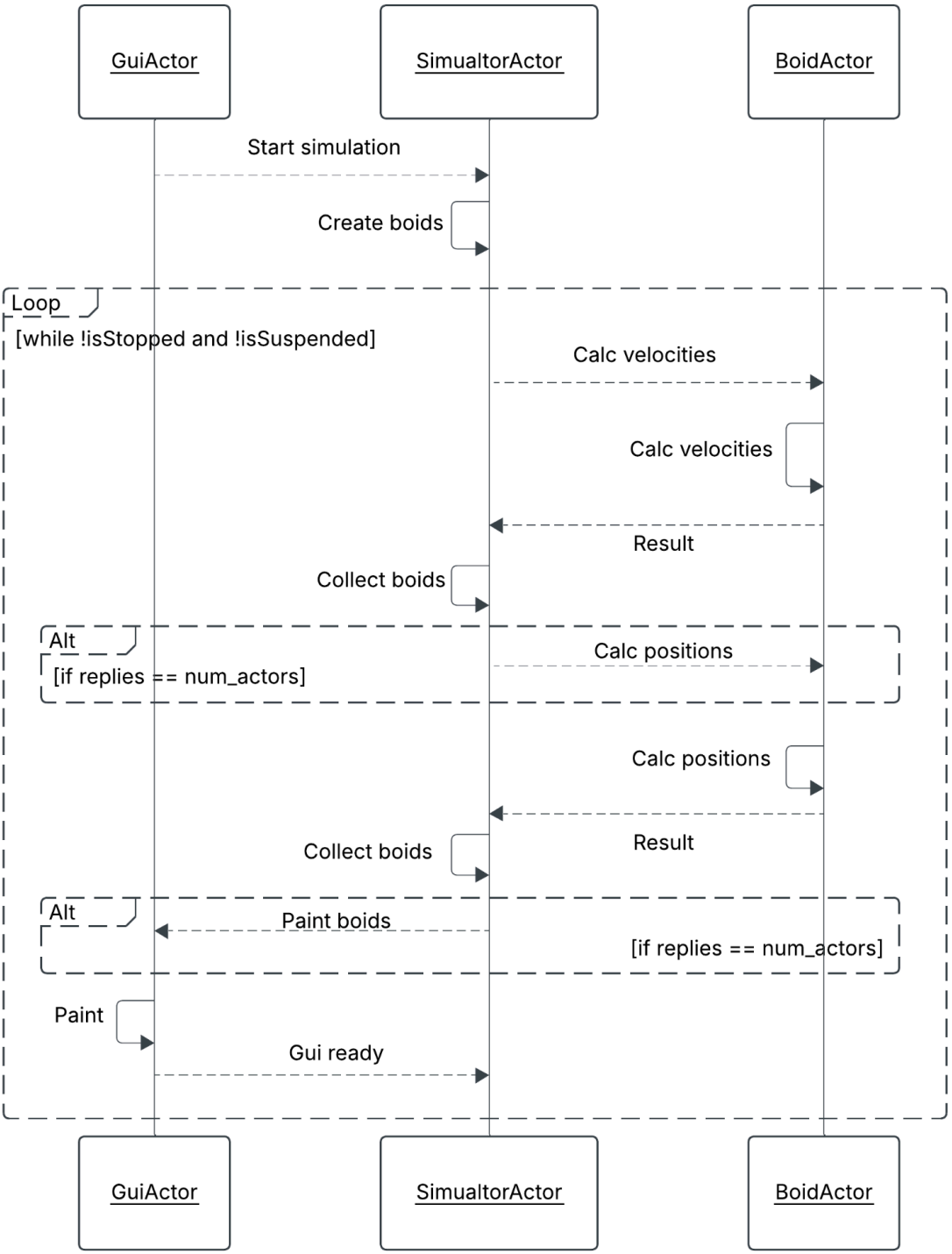

Figura 3: interazione tra gli attori