Simulazione di uno stormo

Piero Selvi, Francesco Martiri, Joachim Sonet, Federico Vecchi

6 Febbraio 2026

Struttura del codice

Il progetto è strutturato in più file sorgente, con ciascun componente principale incapsulato in una classe separata:

- **Boid** (`boid.hpp/.cpp`): gestisce il comportamento di un singolo agente.
- **Flock** (`flock.hpp/.cpp`): gestisce la logica globale dello stormo e lo step di simulazione.

Sono presenti inoltre altri file sorgente che implementano alcuni cambiamenti dall'idea originale:

- **Parametri dinamici** (`slider.hpp/.cpp`): consente di modificare i parametri dello stormo in tempo reale tramite sliders.
- **Valori simulazione** (`simvalues.hpp/.cpp`): una struct che gestisce tutti i parametri e controlla che questi siano in un range accettato.
- **Predatore** (`predator.hpp/.cpp`): gestisce il comportamento del predatore, ovvero un boid rosso più grande e che eredita i metodi di boid e implementa una funzione di inseguimento.
- **Configurazione stormo** (`flockconfiguration.hpp/.cpp`): contiene una configurazione standard per l'inizializzazione del flock e controlla che i valori dati siano sensati per la generazione dei boid del flock.
- **Grafica simulazione** (`simgraphics.hpp/.cpp`): gestisce la parte grafica del flock e del predatore.
- **Vettori 2D** (`vector2D.hpp/.cpp`): implementa dei vettori bidimensionali e le loro operazioni adatte ai nostri scopi.
- **Info cumulative** (`cumulativeinfos.hpp`): una struct che permette di tenere le informazioni cumulative per le operazioni sui boids.

Infine è presente un file `main.cpp` che inizializza il sistema e avvia il ciclo di simulazione.

Scelte progettuali e implementative

Sono state fatte le seguenti scelte progettuali:

- utilizzo di *spatial partitioning* su griglia uniforme per velocizzare il calcolo dei vicini dei boids,
- ogni cella della griglia mantiene una lista concatenata dei boids tramite vettore `header` e vettore `next`,
- lo spazio simulato è toroidale: bordi opposti coincidono per evitare effetti di contenimento,
- per modificare i parametri del flock sono presenti come buona pratica delle funzioni `modify` per evitare valori negativi o valori troppo alti,
- Rispetto al modello originale ora il comportamento dei boid segue un modello di steering basato sulla differenza tra la velocità attuale e la velocità desiderata e una accelerazione limitata.

Librerie esterne

- **SFML**: libreria grafica per output 2D in tempo reale,
- **TBB**: libreria che permette di sfruttare il multithreading.

Installazione delle librerie su sistemi Debian/Ubuntu:

```
apt install libsfml-dev
apt install libtbb-dev
```

Esecuzione del programma

Per compilare ed eseguire il programma devono essere lanciati da terminale i seguenti comandi in ordine:

1. `$ cmake -S . -B build -G"Ninja Multi-Config"`

2. `$ cmake --build build --target run`

Il terminale chiede, per avviare la simulazione, di inserire un numero di boid (qui accetta solo interi); consigliato tra i 500 e i 3000 boids. A questo punto è possibile modificare i parametri in tempo reale tramite sliders presenti sullo schermo.

Parametri di input e output

La simulazione è parametrizzata tramite la struct `SimValues`:

- Parametri modificabili in tempo reale:
  - `s` (float): intensità della separazione,
  - `a` (float): intensità dell'allineamento,
  - `c` (float): intensità della coesione,
  - `e` (float): intensità della fuga dal predatore,
  - `ds` (float): raggio di interazione per la separazione,
  - `ch` (float): intesità dell'inseguimento del predatore.
- Parametri fissi della simulazione:
  - `accmax` (float): accelerazione massima dei boids,
  - `vmax` (float): velocità massima dei boids,
  - `d` (float): raggio di interazione,
  - `dt` e `timescale`: passo temporale e scala del tempo,
  - `maxX, maxY`: dimensioni dello spazio simulato.

Output: rappresentazione grafica 2D dei boids come triangoli orientati, aggiornati in tempo reale. Non vengono prodotti file di output.

Data la già presente simulazione, grafica non sono presenti i parametri a schermo dello stormo (l'andamento nel tempo della distanza media tra i boids, della loro velocità media e le rispettive deviazioni standard). Questi renderebbero pesante il calcolo eliminando un punto di forza del nostro programma (la possibilità di gestire una grande quantità di boid).

Inoltre esce a terminale il seguente messaggio: `Setting vertical sync not supported`.

Interpretazione dei risultati

La simulazione mostra come lo stormo reagisce all'approccio del predatore, evidenziando pattern di fuga collettiva, variazioni di densità e dispersione degli individui. Gli sliders consentono di sperimentare l'effetto dei parametri sul comportamento collettivo.

Strategia di test

Per eseguire i test è sufficente lanciare il seguente comando sul terminale:

```
cmake --build build --target check
```

Il file `flock.t.cpp` utilizza Doctest per test automatici:

- verifica della funzione `getcell` per l'assegnazione dei boids alle celle corrette,
- controllo dei limiti dello spazio toroide,

- controllo delle varie regole ai limiti dello spazio toroide,
- gestione delle eccezioni per celle negative o dimensione cella nulla,
- controllo del chase del predatore su un target,
- alcuni controlli sugli slider.

Uso di sistemi di Intelligenza Artificiale generativa

Uso dell'IA per l'implementazione corretta del file cMake, ispirazione per l'implementazione corretta degli slider e strategia di test, modello per l'implementazione della grafica usando la libreria sfml.

Informazioni aggiuntive

- Compatibile con C++17+,
- Parametri modificabili in tempo reale tramite sliders,
- i valori di d (dimensioni cella della griglia), maxX e maxY(dimensioni dello spazio) sono fissati a runtime e preimpostati da noi.

Tutto il codice è presente su una cartella pubblica

`https://github.com/Martiri/progetto_rn`