Implementazione C++ del modello Boids

Riccardo Marchese, Catalin Sajin

8 gennaio 2023

Sommario

Il progetto, sviluppato collaborando tramite github, consiste nell'implementazione di una simulazione del comportamento degli stormi di uccelli in volo in uno spazio bidimensionale. Reso noto da un software di intelligenza artificiale realizzato nel 1986, la simulazione si basa sull'interazione tra agenti detti boids in uno spazio. Nel modello più elementare gli agenti determinano la loro traiettoria tramite tre regole. **Separazione**: il boid si allontana dai boids vicini; **Allineamento**: il boid tende ad allinearsi alle traiettorie dei boids vicini; **Coesione**: il boid tende a muoversi verso il baricentro dei boids vicini.

1 Introduzione

Si è deciso di implementare questo modello creando varie `struct` che racchiudessero in loro l'essenza del boid (posizione, velocità e numero progressivo) da cui si è creata la classe `UState` che tiene conto delle sue due componenti della posizione, della velocità e dell' `UPN`, il quale aiuta ad identificare il boid all'interno del vettore. Successivamente alla definizione del boid come `UState` si sono divisi due momenti fondamentali: il controllo dei vicini e le regole di volo.

Riguardo **il controllo dei vicini** si è deciso di implementare, anzi tutto, una `struct` di coppia di boids (con solamente due `UState` all'interno), e successivamente una funzione `Check` che prende come argomento la distanza, a partire dalla quale un boid si dice vicino ad un altro, ed il vettore di `UState`, contenente tutti i boids, ed esegue le operazioni di verifica dei vicini, per cui infine tutti i vettori che avranno passato la verifica dei vicini saranno messi in un vettore di `Coppia`, in modo da tenere sempre il conto di quali sono i vicini; in particolare l'operazione si basa su due cicli `for loop` che fanno una verifica "uno - molti" (ossia si procede inizialmente con la verifica del primo con tutti gli altri boids, successivamente del secondo con tutti gli altri escludendo il primo, e così via).

Invece, le **regole di volo** sono state sviluppate a partire dalla considerazione di chi sono i boids vicini, di cui ognuna contiene un `operator()` che permette di calcolare le velocità caratteristiche (funzione coadiuvante è stata la funzione membro `convertPtoV` che ci ha permesso di prendere i valori delle componenti della posizione e convertirli semplicemente in tipo `Velocity`). Ognuna della classi restituisce un vettore di velocità con le velocità caratteristiche delle regole se due o più boids sono vicini, altrimenti darà valore zero. In particolare per le regole Allineamento e Coesione è stato sviluppato un `std::vector<short int> Counters` che permette di tenere conto del numero di operazioni eseguite e, quindi, del calcolo del centro di massa dei boids (cosa che altrimenti non si sarebbe potuta verificare dividendo soltanto per la "size" del vettore di boids).

La classe `Boids` ha anche, in particolare, come membro la distanza (o raggio di visione) dei boids, che ha facilitato lo sviluppo della funzione membro `void Boids::evolve`, al cui interno si sommano le due componenti della velocità e si ricalcola la nuova posizione alla luce delle nuove velocità (qualora il boid fosse abbastanza vicino con altri), implementando anche uno spazio toroidale (nel codice sono stati lasciati commentati anche altri due tipi di spazi, uno chiuso e un altro semichiuso).

Nel `main.cpp` invece è stato inserito quanto necessario per il comparto grafico: il codice infatti dispone di due finestre: una in cui viene mostrata la simulazione vera e propria dei boids e l'altra che mostra gli output della distanza media e deviazione standard.

2 Istruzioni sulla compilazione

Per **compilare il programma** si possono usare due metodi altrettanto equivalenti:

- **Cmake** : Installare anzitutto `Cmake ed Sfml` tramite i comandi

```
# Mac
brew install cmake sfml
# Ubuntu
sudo apt install cmake libsfml-dev
```

Successivamente cambiamo cartella di lavoro eseguendo sul terminale `cd Boids` e creiamo la cartella di build

```
mkdir build
cd build
# Configurazione
cmake -DCMAKE_BUILD_TYPE=Release ../
# Build e creazione dell'eseguibile
cmake --build . -j4 --target Boids
#Testing
cmake --build build --target boids.t
```

- **Senza Cmake**:

```
mkdir build
cd build

#Mac (arm64)
g++ -Wall -Wextra -fsanitize=address ../src/main.cpp ../src/UState.cpp ../src/boids.cpp
-o Boids -I/opt/homebrew/Cellar/sfml/2.5.1_1/include
-L/opt/homebrew/Cellar/sfml/2.5.1_1/lib
-lsfml-graphics -lsfml-window -lsfml-system -std=c++17

#Ubuntu
g++ -Wall -Wextra -O3 -fsanitize=address ../src/main.cpp ../src/UState.cpp
../src/boids.cpp -o Boids -lsfml-graphics -lsfml-system -lsfml-window -std=c++17

#Testing
g++ -Wall -Wextra -fsanitize=address ../src/boids.test.cpp
../src/UState.cpp ../src/boids.cpp -o boids.t -I/opt/homebrew/Cellar/sfml/2.5.1_1/include
-L/opt/homebrew/Cellar/sfml/2.5.1_1/lib
-lsfml-graphics -lsfml-window -lsfml-system -std=c++17

or

g++ -Wall -Wextra -O3 -fsanitize=address ../src/boids.test.cpp
../src/UState.cpp ../src/boids.cpp -o boids.t  -std=c++17
```

Avvio del programma: `./Boids`, oppure `./boids.t` per eseguire i test.

2.1 Input e Output

La schermata di input prevede che vengano inseriti manualmente il numero di boids, i coefficienti di Separazione, Allineamento, Coesione ed infine il raggio di "visione" dei boids. I parametri non accettati per la Separazione e la Coesione sono quelli minori di zero, per l'Allineamento vengono accettati solo parametri tra 0 e 1; mentre, per la distanza tra i boids vengono accettati parametri non negativi e maggiori della distanza di separazione (se così non fosse verrebbe sollevata un eccezione che fa abortire il programma). Un esempio di input è il seguente

```
Inserire il numero desiderato di boids : 200
Inserire il parametro di separazione : .6
Inserire il parametro di allineamento : .4
Inserire il parametro di coesione : .3
Inserire la distanza di interazione fra boids (in pixel; ricordare che il
parametro distanza della separazione è 30 pixel) : 70
```

Bisogna notare che l'ultimo parametro è fortemente dipendente dalla risoluzione dello schermo, per cui mediamente il parametro più adatto è quanto mostrato precedentemente.

Per quanto riguarda l'output viene mostrata sulla seconda finestra la distanza media fra i boids e la deviazione standard che diminuiscono nel tempo (vedi sezione successiva).

3 Intepretazione dei risultati

Alla luce di quanto sviluppato il modello si comporta in modo aspettato, ossia, inserite delle variabili adatte (vedi sezione successiva), gli output di media e deviazione standard tendono a diminuire nel tempo in modo coerente con quanto aspettato; si può facilmente comprendere che all'aumentare del tempo la distanza media tra i boids tende a diminuire e dunque essi tendono a formare un unico stormo per cui le distanze tra loro diminuiscono considerevolmente (solitamente con una buona simulazione la distanza fra boid alla fine arriva ad essere circa minore o uguale a 200, mentre la deviazione standard minore o uguale a 100).

Nota: la distanza media e la deviazione standard sono spesso soggetti a fluttuazioni (cambi repentini da valori "bassi" ad "alti") dovuto al tipo di spazio implementato, il quale consente di passare da un angolo ad un altro modificando le coordinate in x e in y dei boids.

4 Strategie sull'utilizzo del modello

Le strategie principali di verifica, che quanto ottenuto sia esente da errori, sono state lo sviluppo di test tramite `DOCTEST` che ha permesso di testare il funzionamento della funzione di ricerca dei vicini, delle varie classi sulle regole di volo e principalmente la simulazione di volo tra boids e il modo in cui si evolve la loro interazione.

Dunque, il test principale, dal momento che questo modello così implementato accetta in input 5 parametri, risiede nell'esecuzione del programma stesso che ci permette di verificare quali parametri in input sono i più adatti. Si è verificato che immettere un numero di boids troppo esiguo, così come anche per il coefficiente di separazione, non permette di eseguire una simulazione abbastanza "ragionevole". Invece, per quanto riguarda il parametro di coesione, qualora l'utente mettesse in input un coefficiente troppo alto la visione dei boids verrebbe "falsata" (si osserva una sovrapposizione di un immagine sull'altra, dato che la forza di coesione è considerevolmente alta) e di conseguenza consigliamo di seguito dei parametri modello che aiutano una buona simulazione.

- **Numero di boids**: da 50 a 400 (oltre questo numero il computer tende a far aumentare il lag di display);
- **Coefficiente di separazione**: da 0.3 a 20 (solitamente mantendo la separazione abbastanza alta si permette alla simulazione di durare più a lungo data la difficoltà a unirsi tra loro dei boids);
- **Coefficiente di allineamento**: da 0.4 a 0.6;
- **Coefficiente di coesione**: da 0.1 a 0.5;
- **Distanza dei boids**: da 50 a 100 (solitamente dipende dalla risoluzione dello schermo che l'utente possiede, per cui si consiglia di mettere valori bassi per schermi Full HD e alti per chi possiede schermi 4K);