Relazione sul Progetto della Simulazione di Boids

Daniele Guiso, Alessandro Coppola, Filippo Contessa

08 Settembre 2022

Indice

1 Introduzione

Boids è un software realizzato per la prima volta nel 1986 da Craig Reynolds allo scopo di simulare il comportamento degli stormi di uccelli in volo.

Nel campo di applicazione scientifico,essi forniscono un modello matematico per simulare sistemi contenenti un numero elevato di elementi, anche detti **sistemi complessi**.

Nella sua versione più semplificata, il singolo boid possiede la sua personale posizione e velocità, parametri che possono essere modellati per simulare le interazioni tra i vari boid mediante 3 semplici regole:

1. **Separazione**, per cui il singolo boid cercherà di mantenere una distanza minima da tutti gli altri;
2. **Allineamento**, per cui il singolo boid tenterà di allinearsi alla traiettoria dello stormo;
3. **Coesione**, per cui il singolo elemento tenderà ad avvicinarsi al baricentro dello stormo;

Si noti che da qua in avanti, utilizzeremo il termine *flock* per indicare il vettore contenente l'insieme dei boids.

2 Principali scelte Progettuali

Il programma è suddiviso nei seguenti file:

main.cpp e main_sfml.cpp

Costituiscono i file iniziali che, presi in input i parametri per le variabili non costanti (numero di boid e fattori di separazione, allineamento e coesione), chiama le prime funzioni che a cascata faranno girare tutto il codice e, stampa a schermo i risultati nel caso di *main.cpp*, avvia la libreria grafica di sfml nel caso di *main_sfml.cpp*. Possiedono *int main*, per l'inizializzazione delle struct e dei valori in input, *void simulation*, per la reiterazione della funzione di evoluzione del flock e per la stampa a schermo dei risultati, e *struct val_simulation* per i valori utili alla sola simulazione.

flock.cpp e flock.hpp

Source ed header file per l'implementazione di tutte le caratteristiche del flock. Il source file possiede *struct values* contenente tutti i valori utili al flock, *struct boid* che descrive le caratteristiche spaziali e di velocità dei singoli boid, e *class Flock* con le definizioni delle member function per la creazione e popolazione (*void add_boids*) di flock e quelle utili alla sua evoluzione (*void update_flock*) e alla descrizione dello stato (*float velocity_mean*, *float d_separation_mean*, *coordinates center_mass*, *coordinates velocit_cm*), oltre ad alcune utili operazioni per la *struct*

coordinates e per il calcolo della deviazione standard. Nell'header file invece sono state dichiarate tutte le funzioni appena descritte.

evolve.hpp

Header file che controlla l'aggiornamento della posizione e velocità del vettore flock (*void update_position* e *void update_velocity*). È presente anche la funzione che sceglie i boid vicini (*auto checking_neighbors*) e l'angolo di visione dei singoli elementi (*bool boid_vision*).

velocity_rules.hpp

Header file che richiama tutte le funzioni legate alla modifica delle velocità dei boid. Le principali funzioni sono quelle per le velocità di separazione (*coordinates separation_velocity*), allineamento (*coordinates alignment_velocity*) e coesione (*coordinates coesion_velocity*), ma è presente anche la funzione per il comportamento ai bordi (*coordinates velocity_edge*) seguendo la scelta di limitare l'aerea di movimento dei boid ad un quadrato dai bordi chiusi, e le funzioni per limitare le velocità (*coordinates velocity_limit*).

test.cpp

È stato implementato un **unit test**, usando **Doctest**, per testare il comportamento delle principali funzioni. Questo aspetto verrà approfondito in seguito.

3 Come compilare ed eseguire

Seguono le istruzioni valide per Linux:

3.1 Stampa su terminale

Da terminale recandosi nella cartella contenente tutti i file del progetto, è possibile compilare i due source file *main.cpp e flock.cpp* con **g++**, il compilatore C++ di GNU, con il seguente comando:

$ g++ main.cpp flock.cpp

Se privo di problemi, restituirà l'eseguibile *a.out* che potremo eseguire con il successivo comando:

$ ./a.out

3.2 Libreria grafica

Similmente a prima, da terminale recandosi nella cartella contenente tutti i file del progetto, è possibile compilare i file *main_sfml.cpp e flock.cpp* con **g++**

usando questa volta il comando **-c** per non far partire i linker e avere compiles e assembles separati. Il comando è il seguente:

g++ -c main_sfml.cpp flock.cpp

Colleghiamo ora il compile file con la libreria di SFML per avere il file eseguibile finale con il comando di seguito:

$ g++ main_sfml.o flock.o -o sfml-app -lsfml-graphics -lsfml-window -lsfml-system

Possiamo ora eseguire il programma compilato:

$ ./sfml-app

4 Descrizione del formato di input e di output

Input

Una volta compilato ed eseguito il programma, questo richiede in **input**:

- Il numero di boid;
- I tre parametri utilizzati nelle tre regole di volo (separation, alignment e coesion).

Sono messi tra parentesi i **valori suggeriti** per i rispettivi parametri:

- Number of boids (n $>$ 0): (100);
- Separation factor (s $>$ 0): (2);
- Alignment factor (0 $<$ a $<$ 1): (0.4);
- Coesion factor (c $>$ 0): (0.2).

Output

Invece, in **output** il programma mostra per ogni step:

- distanza media tra boids (dsm), con annessa deviazione standard della media.
- la velocità media dei boids (vm) corredata da rispettiva deviazione standard della media.
- la posizione del centro di massa (cm) di tutti i boids presenti.
- Vengono inoltre mostrati i valori pre-impostati (preset values) della simulazione.

Per un esempio dei valori tabulati, si veda Tab.1.

Step	vm	std	dsm	std	cm x	cm y	vcm x	vcm y
1	33.34	1.32	316.07	2.13	315.48	290.81	0.41	-3.25
100	55.25	1.20	425.99	3.06	295.20	369.85	2.28	-1.31
200	51.41	1.83	411.35	3.13	328.38	406.86	0.03	12.87
300	47.41	1.44	445.30	3.27	354.21	318.18	4.86	16.38
400	50.99	1.53	435.45	3.31	390.39	316.27	4.95	11.89
500	46.48	1.68	434.62	3.32	364.45	385.17	5.99	4.31

Tabella 1: Esempio di simulazione

Mediante SFML

Il programma compilato ed eseguito con SFML, a differenza di quello precedente, genera in output un interfaccia grafica che riproduce l'andamento dei boid nel tempo invece che la stampa a schermo della tabella con tutti i valori.

5 Interpretazione dei risultati ottenuti

Dai dati ottenuti in output è possibile fare le seguenti considerazioni.

Se opportunamente simulati i boids dovrebbero tendere ad avvicinarsi tra loro, modellando ed uniformando le singole velocità. Si noti inoltre come se ciò fosse confermato si dovrebbe riscontrare una riduzione della deviazione standard della velocità media e della distanza media.

Mediante SFML, nella finestra grafica si dovrebbe osservare un comportamento simile a quello degli stormi, con gruppi condensati di boids che non occupano tutto lo spazio a disposizione seguendo direzioni casuali. Nel muoversi inoltre, i boids dovrebbero mantenere una distanza minima tra di loro.

6 Strategia di test

Utilizzando Doctest sono stati implementati dei Test atti a verificare la correttezza delle singole funzioni rispetto le aspettative iniziali.

Si sottolinea come i vari TESTCASE siano stati divisi in ulteriori SUBCASE che rappresentano casi particolari. Per esempio nel TESTCASE che prendeva in considerazione la velocità ai bordi è stato testato il comportamento di un boid in tutti e quattro i lati per verificare che la regola funzionasse in modo uniforme.

7 Appendice

Si evidenzia l'utilizzo durante l'implementazione del programma dei seguenti strumenti:

- **Clang Format**, un formattatore automatico di codice.

- **GitHub**, utile per la sincronizzazione dei file locali mediante una repository online. Si allega il link della repository online:

  https://github.com/DanieleGsu/project.

- **Sfml**, per la parte grafica del progetto.