Simulazione del comportamento di stormi

Mazzarelli Elena, Cavazza Elisa

1 Scelte progettuali e implementative

Il progetto tenta di simulare il comportamento di stormi di uccelli in uno spazio bidimensionale. A tal proposito si è seguito il modello fornito da Craig Raynolds nel 1986, in cui agenti detti boids interagiscono tra di loro seguendo le seguenti regole:

- separazione: un boid si allontana dai boids vicini;
- allineamento: un boid tende ad allinearsi alle traiettorie dei boids vicini;
- coesione: un boid tende a muoversi verso il baricentro dei boids vicini.

Il programma presenta sette translation units: boid.hpp, boid.cpp, flock.hpp, flock.cpp, boid.test.cpp, main.cpp e main-sfml.cpp. Nei file boid e flock si sono implementate due classi, Boid e Flock, volte a riprodurre rispettivamente il comportamento del singolo agente e dello stormo. Per quanto riguarda la classe Boid, questa presenta come membri privati la posizione, la velocità, i parametri del modello e la velocità massima. Per rappresentare posizione e velocità, i quali sono vettori bidimensionali, si è impiegato il template della libreria sfml "sf::Vector2 ⟨ T ⟩", prendendo come tipo double. Si sono inoltre implementate tre funzioni volte a determinare rispettivamente la differenza fra due vettori (bd::distance) e il modulo e l'angolo di un singolo vettore (bd::magnitude, bd::angle), al fine di semplificare l'implementazione delle regole di volo e della parte grafica. I parametri del modello, invece, sono racchiusi nello struct Parameters e sono: "d", "ds", "s", "a", "c". I parametri "d" e "ds" rappresentano rispettivamente le distanze per le quali si attivano le regole di volo e la regola di separazione. In particolare, "ds" deve essere minore di "d". I parametri "s", "a", "c", sono i fattori di scala rispettivamente delle regole di separazione, allineamento e coesione, e devono avere valori compresi tra zero e uno. Ciò viene imposto tramite l'uso di eccezioni che interrompono l'esecuzione del programma qualora questi valori non siano rispettati.

Le regole di volo dei boids sono dettate dalle seguenti leggi matematiche, in cui $\vec{v}_1$, $\vec{v}_2$, $\vec{v}_3$ rappresentano le variazioni di velocità date rispettivamente dalle regole di separazione, allineamento e coesione, e $\vec{x}_c$ è il centro di massa dello stormo:

$$\vec{v}_1 = -s \sum_{j \neq i} (\vec{x}_{b_j} - \vec{x}_{b_i}) \quad \text{se} \quad \left|\vec{x}_{b_i} - \vec{x}_{b_j}\right| < d_s$$

$$\vec{v}_2 = a \left(\frac{1}{n-1} \sum_{j \neq i} (\vec{v}_{b_j} - \vec{v}_{b_i}) \right)$$

$$\vec{x}_c = \frac{1}{n-1} \sum_{j \neq i} \vec{x}_{b_j}$$

$$\vec{v}_3 = c(\vec{x}_c - \vec{x}_{b_i})$$

Queste leggi sono state implementate tramite tre metodi nella classe Boid ("separation", "alignement", "cohesion"). In particolare, sono stati impiegati dei range loops per iterare i boid in un vettore. All'interno di questi range loops degli if statements controllano che le condizioni per l'attivazione delle regole siano verificate per ciascun boid, e in caso affermativo la rispettiva posizione o velocità viene sommata. Le sommatorie escludono il vettore su cui è attivato il metodo, di conseguenza nel metodo "cohesion" dalla somma delle posizioni viene sottratta la posizione del boid di cui vogliamo determinare la variazione di velocità. Nell' if statement a ogni iterazione N viene incrementato di uno, partendo da zero, per determinare il numero di boids coinvolti nell'applicazione della regola. Se N è maggiore di uno vengono calcolati le velocità risultanti dalle regole.

Per determinare la nuova velocità e posizione del boid sono state implementati i metodi "updateVelocity" e "updatePosition", i quali restituiscono i valori dati dalle seguenti formule:

$\vec{v}_{b_i} = \vec{v}_{b_i} + \vec{v}_1 + \vec{v}_2 + \vec{v}_3$

$\vec{x}_{b_i} = \vec{x}_{b_i} + \vec{v}_{b_i}\Delta t$

Il metodo "updateVelocity" tiene conto anche della velocità massima ("maxspeed"), per cui se tale velocità viene superata le componenti del vettore "velocity" vengono riscalate.

Inoltre, è stato implementato un metodo "borders" per descrivere il comportamento dei boid ai bordi dello schermo. Si è scelto un modello toroidale, per cui se il boid supera un bordo viene teletrasportato al lato opposto dello schermo. Degli if statements verificano se il boid supera il bordo dello schermo le cui coordinate sono 1280 e 720. Se tali condizioni sono verificate, la posizione del boid viene modificata per trasportarlo dall'altro lato.

I metodi "updateVelocity", "updatePosition" e "borders" vengono chiamati insieme nel metodo "update", che serve ad aggiornare posizione e velocità ogni volta che viene chiamato.

La classe Flock presenta come membro privato un vettore di boid. Per aggiungere boid a questo vettore è stato implementato il metodo "addBoid" che impiega "push_back", mentre per averne la dimensione c'è il metodo "size". Il metodo "updateFlock" chiama tramite un range loop il metodo "update" per ciascun boid appartenente allo stormo aggiornandone posizione e velocità. E' possibile avere distanza e velocità media dello stormo tramite i metodi "average_distance" e "average_speed", i quali calcolano media e deviazione standard, statistiche racchiuse nello struct "Statistics".

Per realizzare gli istogrammi si è implementata la funzione "histogram" che prende come parametri due vettori, uno contente le medie e l'altro le deviazioni standard, e un double per la norma. La funzione per ogni elemento dei vettori stampa a inizio riga media e deviazione standard e poi, se la deviazione standard è minore del valore medio, stampa tante righe "-" fino all'intervallo di confidenza della misura in considerazione. Tale intervallo ha come centro la media (indicata con un asterisco "*") e ampiezza due deviazioni standard. Gli estremi dell'intervallo sono dati dal simbolo "σ". Se la deviazione standard è maggiore della media verranno stampate tante righe "-" fino al valore medio.

All'interno del main.cpp vengono gestiti parte degli input e output del programma. In particolare se l'input è il char "g" vengono chiesti parametri e numero di boid. Tramite queste informazioni viene generato uno stormo con N boid posizionati uniformemente sullo schermo e con velocità comprese fra -1 e 1. Per fare ciò si è usata la classe std::default_random_engine della libreria "random" e si sono dichiarate quattro variabili prese da distribuzioni uniformi. Viene inoltre inizializzata la velocità massima a 500 e impostati i parametri tramite il metodo "setParameters" della classe Flock. In seguito tramite un while loop posizioni e velocità degli stormi vengono aggiornati dieci volte. Il while loop agisce su un tempo di 10s partendo da 0s e a ogni iterazione aggiunge un delta_t di 1s. Tale delta_t è usato come parametro di "updateFlock". A ogni iterazione vengono calcolati media e deviazione standard delle distanze e delle velocità e vengono inseriti nei vettori "av_distances", " s_distances", "av_speeds" e "s_speeds". Se l'input è il char "s" il contenuto di tali vettori viene stampato a schermo, mentre con il comando "h" i vettori vengono usati come parametri della funzione "histogram". Le norme degli istogrammi vengono prese in input.

In modo analogo, all'interno del file main-sfml.cpp viene chiesto all'utente di inserire in input da tastiera uno di tre comandi: il comando "g" chiederà di inserire il numero di boids N ed i parametri del modello (s, a, c, d, ds). Questi dati vengono inseriti in un oggetto della struct Parameters. Tramite l'uso della classe std::uniform_real_distribution e di un seed random dato da un random engine si genera una distribuzione uniforme di posizioni su tutta l'area della finestra e di velocità comprese fra -1 e 1. Questo procedimento si ripete per tutti gli N boids tramite un for loop, nel quale si fa uso dei metodi setVelocity, setMaxspeed e setPar per aggiornare ad ogni ciclo del loop l'oggetto di tipo Boid chiamato boid. Questo procedimento serve a dare a tutti gli oggetti dello stormo una posizione e una velocità iniziali per t = 0, prima che le regole di volo inizino ad avere effetto. Infine si aggiunge ogni boid ad un oggetto di tipo Flock chiamato flock1 tramite addBoid().

Il comando "b" apre una finestra della libreria grafica SFML e successivamente un evento: questo permette di chiudere la finestra in qualsiasi istante premendo il tasto "esc" da tastiera o cliccando

la X in alto a destra nella finestra.
Prima di disegnare i boids si esegue una chiamata a sf::RenderTarget::clear() per liberare lo schermo da eventuali oggetti già presenti nella finestra. Successivamente si chiama il metodo "updateFlock" per aggiornare posizioni e velocità dell'oggetto flock1 e per fare il controllo del comportamento ai bordi dello schermo tramite borders(). Poi si disegnano le figure (triangoli isosceli) per mezzo della classe sf::ConvexShape e chiamando il metodo sf::RenderTarget::draw(). Viene anche incluso l'utilizzo della funzione "angle" per fornire una rotazione ai boids, con la dovuta conversione da radianti a gradi. All'esterno del loop vi è una chiamata a sf::Window::display() per mostrare sullo schermo ciò che è stato prodotto dal rendering. Per finire, il comando "q" termina l'esecuzione del programma.

2 Test

Si è impiegato doctest per testare le classi Boid e Flock e le funzioni libere relative ai vettori bidimensionali.A tal proposito il file "boid.test.cpp" presenta tre test case e un totale di 85 subcases. Per quanto riguarda i test relativi alla classe Boid, questi coprono la corretta gestione delle posizioni e delle velocità dei boid, oltre alla configurazione e all'aggiornamento dei parametri dei boid stessi. In particolare, i test verificano le regole di separazione, allineamento e coesione, assicurando che restituiscano i valori dettati dal modello matematico rispecchiando le condizioni dettate dai parametri. Per quanto riguarda le funzioni di aggiornamento, si è verificato il loro comportamento in situazioni standard e nei casi limite, ovvero quando un boid supera il bordo dello schermo o la velocità massima. Si è inoltre verificato che le eccezioni lanciate nei metodi per settare i parametri funzionino correttamente. Per quanto riguarda la classe Flock, i test sono finalizzati a verificare la corretta gestione dei boid all'interno di uno stormo. Sono stati testati l'aggiunta e il recupero dei boid dalla classe Flock, nonché il calcolo della velocità e della distanza medie all'interno del gruppo. Inoltre, vengono esaminati gli effetti dell'aggiornamento della velocità e della posizione dell'intero stormo, garantendo che il comportamento collettivo dei boid rispetti le regole stabilite. In fase di debugging, all'interno di ciascuno dei tre metodi per le regole di volo della classe Boid (separation, alignment, cohesion) è stato introdotto un assert statement su distance1 per verificare che "distance" restituisse correttamente un valore nullo o positivo. La stessa cosa è stata fatta su mag_v dentro al metodo "updateVelocity" per verificare che "magnitude" funzionasse correttamente. Nella classe Flock, si è ripetuto lo stesso procedimento per distance1 e average_distance dentro al metodo "average_distance()" e per average_speed all'interno di "average_speed()".

3 Istruzioni

Sono stati impiegati come strumenti per la realizzazione del progetto Doctest, Clang-format, cMake e la libreria grafica sfml. Il comando per scaricare quest'ultima tramite terminale è:

```
sudo apt-get install libsfml-dev
```

Per la compilazione si è impiegato cMake. Per compilare i due main si usa lo stesso file cMake in cui però va modificato alla riga 22 l'argomento di add_executable: di default è inserito main.cpp, il quale va sostituito con main-sfml.cpp in caso si voglia eseguire la parte grafica. I comandi per compilare ed eseguire i due main sono gli stessi e sono :

```
cmake -S . -B build -DCMAKE_BUILD_TYPE=Debug ;
cmake --build build ;
./build/boid ;
```

Per verificare invece i test i comandi sono i seguenti:

```
cmake -S . -B build -DCMAKE_BUILD_TYPE=Debug ;
cmake --build build --target test ;
```

4 Input e output

Sono stati realizzati due possibili file contenenti il main: main.cpp e main-sfml.cpp.
Una volta eseguito il primo, verranno mostrati una serie di comandi validi: "g" per generare uno

stormo, "s" per generare le distanze e velocità medie nell'arco di 10 secondi e "h" per generare gli istogrammi relativi a tali statistiche. Prima di digitare "s" o "h" è necessario generare lo stormo, altrimenti verrà lanciata un'eccezione. Digitato "g" verranno richiesti in input i parametri del modello e il numero di boid.

ES:
d = 500;
ds = 50;
s = 0,02;
a = 0,02;
c = 0,02;
N = 1000.

Per N il programma accetta solo valori interi maggiori di 2 (numero minimo per l'attivazione delle regole di volo). I parametri s, a, c hanno valori inclusi tra 0 e 1; la distanza d deve essere un numero positivo e la distanza ds un numero positivo ma minore di d. In particolare, si consigliano valori di ds molto minori di d (all'incirca un ordine di grandezza) per risultati ottimali. Tutti questi range di valori vengono controllati catturando delle eccezioni qualora qualche input non dovesse rientrare nei range adatti.

A partire da questi dati il main genererà N boids disposti uniformemente sullo schermo e calcolerà le statistiche accessibili tramite "s". Per visualizzare gli istogrammi oltre a digitare "h" si dovranno inserire anche le norme dei rispettivi istogrammi. Le norme devono necessariamente avere dei valori positivi e non nulli, anche in questo caso l'input viene controllato lanciando un'eccezione. Valori consigliati come norme per visualizzare righe adatte a schermo intero sono 25 per l'istogramma delle distanze medie e 5 per quello delle velocità medie.

Eseguendo invece il file main-sfml.cpp, verranno mostrati i comandi possibili: "g" per generare uno stormo, "b" per visualizzare la simulazione, "q" per terminare l'esecuzione del programma . Prima di digitare "b" è necessario generare lo stormo, altrimenti verrà catturata un'eccezione. Digitato "g" verranno richiesti in input i parametri del modello e il numero di boid, ad esempio:

N = 1000.
s = 0,5;
a = 0,4;
c = 0,5;
d = 300;
ds = 50;

I valori validi per i parametri N, s, a, c, d, ds coincidono con quelli del file main.cpp. A partire da questi parametri, tramite il comando "b" il programma mostrerà i boids appena generati su una finestra della libreria grafica. Per generare un nuovo stormo è sufficiente chiudere la finestra e scegliere nuovamente il comando "g".

5 Risultati ottenuti

Le statistiche relative alle distanze e velocità medie mostrano come per valori standard dei parametri all'avanzare del tempo la distanza fra i boid diminuisca, mentre la velocità aumenta, risultati aspettati in quanto i boid tendono ad avvicinarsi e a formare degli stormi.
Negli istanti iniziali della simulazione si può verificare in alcuni casi che gli errori siano più grandi dei valori medi. Ciò è dovuto al fatto che i boid sono generati con posizioni e velocità iniziali casuali e di conseguenza essendo tali valori molto diversi tra di loro, la varianza risulta essere molto alta.

Per valori intermedi (circa come quelli indicati nel secondo esempio) la grafica permette di vedere la formazione e l'evoluzione di uno stormo: la sua velocità aumenta fino a raggiungere quella massima e venire riscalata, le distanze tra boids tendono ad avvicinarsi nei limiti imposti dai parametri s e ds.
Per valori estremi di uno o più dei parametri, lo stormo tende o a non formarsi (nel caso di N e d troppo piccoli, oppure s molto maggiore di a e c) o a formare piccoli flock che non si uniscono tra loro e oscillano attorno alle loro posizioni iniziali (nel caso di ds troppo vicino al valore di d).

Inoltre, in certi casi si si possono verificare divisioni dello stormo quando parte di questo supera il bordo dello schermo: questo è dovuto al fatto che le posizioni dei boids che superano il bordo dello schermo vengono calcolate in accordo con la geometria toroidale da "borders()", mentre le distanze tra boids sono in accordo con una distanza lineare $d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$ in quanto l'utilizzo di una distanza toroidale comporterebbe una tassellazione dello spazio non in accordo con l'implementazione delle regole di volo.