Boids - Simulazione del volo di uno stormo di uccelli

DIFA UniBo - Programmazione per la Fisica

Michele Busti, Virginio Miroglio, Simone Pasquini, Giacomo Pivetti

Anno Accademico 2021/2022

Indice

1 Scelte progettuali e implementative

Il progetto è suddiviso in diversi header file e file di implementazione in modo da gestire e suddividere al meglio i blocchi di base necessari per il corretto funzionamento della simulazione. In particolare, il codice è diviso in:

- **Objects:** in "objects" vengono definite classi e struct utili all'implementazione, come ad esempio i vettori 2-dimensionali per rappresentare posizioni e velocità dei boid, e con loro i vari metodi e le funzioni libere utili a gestirne alcuni calcoli matematici.

- **Functions:** in "functions" vengono definite tutte le funzioni inerenti l'evoluzione dello stormo (come le regole di volo), quelle necessarie a una corretta evoluzione dovute alle scelte nell'implementazione di evolve (come la funzione `influence`) e quelle utili al calcolo degli output richiesti (distanza e modulo della velocità media, con relative deviazioni standard).

- **Evolve:** in "evolve" sono contenute le funzioni che rendono possibile la simulazione dell'evoluzione dello stormo nel tempo. In particolare, si è scelto di definire due funzioni, una per l'evoluzione del singolo boid e una per l'evoluzione dell'intero stormo (rispettivamente `evolve_boid` ed `evolve_flock`). In questo modo, oltre a una maggiore pulizia e linearità dal punto di vista logico, è stato estremamente più semplice gestire l'evoluzione dei singoli boid nel caso essi si trovassero in situazioni particolari (come avere velocità nulla o essere in una posizione troppo vicina al predatore) che tendono a verificarsi raramente, ma di cui si deve tenere conto. Riguardo ai bordi della simulazione, è stato scelto di riposizionare ogni boid che ne supera uno sul bordo opposto (lo spazio è *bidimensionale toroidale*).

- **Sfml objects:** in "sfml objects" sono stati definiti due oggetti estremamente utili per l'implementazione di funzionalità aggiuntive alla simulazione, avvenuta attraverso SFML; in particolare, sono state importanti le creazioni di una classe `button`, che rende significativamente più agevole creare dei veri e propri pulsanti in SFML, e di una classe `data`, che permette la visualizzazione dei dati richiesti in output in continuo aggiornamento.

- **Sfml:** in "sfml" è presente un'unica funzione (uno dei motivi per cui si è scelto di non dividere questa parte in header file e file di implementazione), `run_simulation`. Tale funzione prende in input tutti i valori

necessari a creare la simulazione di uno stormo di boid, crea tutti gli oggetti di SFML necessari a una corretta visualizzazione grafica (pulsanti, dati di output, rappresentazione grafica di boid e predatore) e renderizza la finestra grafica con tutti gli oggetti appena citati.

- **Main:** il "main" è il blocco più semplice: contiene solamente il codice necessario ad accettare in input il numero di boid iniziali e i parametri per le regole di volo, per poi visualizzare la simulazione attraverso la funzione `run_simulation`. I boid vengono generati secondo una distribuzione gaussiana che restituisce velocità e posizioni iniziali appropriate, secondo le regole imposte ai bordi della finestra grafica e la velocità massima dei boid.

2 Istruzioni di compilazione, testing ed esecuzione

Al fine di produrre l'eseguibile ed i test, per la compilazione dei file sorgente è stato scelto l'utilizzo di CMake. Inserendo in input nel terminale (dalla directory contenente tutti i file necessari per il funzionamento del programma) il comando:

```
$ cmake -S . -B build -DCMAKE_BUILD_TYPE=Debug
```

vengono configurate le istruzioni di compilazione (il tipo `DEBUG` aggiunge anche `-fsanitizer=address` fra le istruzioni di compilazione). Viene creata una cartella build con le varie istruzioni necessarie a CMake per compilare correttamente i file contenenti il codice sorgente. Attraverso il comando:

```
$ cmake --build build
```

(sempre dalla stessa directory di partenza) vengono creati direttamente nella directory in cui ci si trova l'eseguibile `simulation` e i test `simulation.t`.

Eseguendo il file `simulation` verrà richiesto l'inserimento del numero iniziale di boid e dei parametri s, a, c (separazione, allineamento e coesione) relativi alle tre regole di volo, in questo ordine. In seguito, verrà visualizzata la finestra grafica contenente i boid (esagoni arancioni) e un predatore fermo (quadrato nero). Inoltre, in alto a destra verranno visualizzati gli output in continuo aggiornamento, mentre in alto a sinistra saranno presenti tre pulsanti attraverso i quali è possibile aggiungere o rimuovere boid dalla simulazione (il numero minimo di boid presenti deve essere sempre 2, per evitare errori o crash della simulazione) e un pulsante che permette di bloccare momentaneamente l'evoluzione della simulazione nel caso si volessero

visualizzare i dati di output in un determinato istante.

Premendo la barra spaziatrice è possibile mettere in movimento e fermare il predatore, è possibile inoltre cambiarne la direzione e il verso del moto lungo gli assi del piano cartesiano attraverso le freccette direzionali della tastiera. Premendo `LCtrl` / `LShift` il predatore aumenterà / diminuirà la sua velocità. Aumentando significativamente la velocità del predatore è possibile superare l'effetto di separazione con i boid e, riuscendo ad avvicinarsi oltre una certa soglia, il predatore eliminerà dallo stormo i boid troppo vicini a lui (è possibile tenere conto di questo effetto controllando il contatore del numero di boid presente in alto a sinistra). Per le stesse motivazioni date per il pulsante `remove boid`, una volta arrivati ad uno stormo di due soli boid il predatore non sarà più in grado di eliminarne altri. La funzionalità attraverso la quale il predatore è in grado di eliminare i boid dallo stormo viene attivata dopo due secondi dall'inizio della simulazione (le motivazioni sono spiegate in "Note aggiuntive") . Quando la funzionalità sarà attiva il predatore si colorerà di blu.

È inoltre possibile incrementare / diminuire a step di 0.05 i parametri s, a, c. Premendo il relativo pulsante della tastiera (le lettere s, a, c) viene selezionato uno di questi 3 parametri (è indicato con una freccetta il parametro attualmente selezionato), premendo successivamente i tasti `enter/backspace` il parametro selezionato viene rispettivamente aumentato o diminuito.

È comunque possibile in qualunque momento durante la simulazione premere il tasto H della tastiera per far si che nel terminale vengano restituite tutte le istruzioni qui riportate.

2.1 Note aggiuntive

1. È possibile che il numero iniziale dei boid differisca di 1 o 2 da quello inserito, questo è dovuto al fatto che, essendo generati casualmente, alcuni boid potrebbero avere una velocità troppo elevata e in direzione del predatore (inizialmente fermo). Di conseguenza la velocità di separazione dal predatore non è in grado di contrastare l'avanzamento del boid in direzione del predatore per cui quello stesso boid viene eliminato. Per ovviare a questo problema si è scelto di attivare la funzionalità che permette di eliminare i boid attraverso l'uso del predatore solo dopo un certo lasso di tempo dall'inizio della simulazione durante il quale le velocità dei boid si stabilizzano.

2. Compilando con l'opzione `-fsanitize=address`, una volta chiusa la finestra grafica (e terminato quindi il programma) vengono restituiti dal terminale dei messaggi di errore riguardanti dei memory leak. A seguito di una ricerca approfondita si è giunti alla conclusione che gli errori restituiti sono in realtà molto comuni e non dovuti a problematiche del codice sorgente, ma molto probabilmente a incompatibilità fra driver grafici e SFML. Difatti, in fase di esecuzione non vengono comunque restituiti messaggi di errore riguardanti memory leak.

3 Formato di input e output

Al fine di ottenere una corretta compilazione è necessario fornire come dati in input il numero inziale di boid, di tipo int, e i tre parametri s, a, c di tipo double (utilizzati per calcolare rispettivamente le velocità di separazione, allontanamento e coesione). Per un risultato ottimale, è preferibile inserire un numero iniziale di boid compreso fra 50 e 250 ed i seguenti valori per i tre parametri: s = 0.7, a = 0.02, c = 0.01. Questi valori sono stati individuati a seguito di numerosi test con il fine di creare una simulazione più corretta possibile. L'output della simulazione comprende la distanza media fra i boid con relativa deviazione standard e velocità media dei boid con relativa deviazione standard. La visualizzazione dell'output avviene attraverso la rappresentazione di un oggetto testo di SFML nella finestra grafica che viene aggiornato automaticamente ogni due secondi con nuovi dati. Il pulsante pause evolution permette inoltre di mantenere più a lungo visualizzati i dati relativi a un certo istante di tempo della simulazione bloccando momentaneamente l'evoluzione dello stormo.

4 Interpretazione dei risultati ottenuti

Per un'interpretazione ottimale dei risultati si può studiare il comportamento che i boid hanno nella simulazione visualizzata grazie all'uso di SFML. Quello che inizialmente è un'insieme confusionario, in un breve numero di evoluzioni si trasforma in uno o più stormi coesi che, incontrandosi, si uniscono andando a formare un unico stormo. Ciò è dovuto al fatto che l'area della simulazione è molto maggiore di quella di influenza fra i singoli boid e perciò i vari piccoli stormi si uniscono solo quando vengono a contatto fra loro. Dalla creazione di un unico stormo seguono una diminuzione della distanza media tra i boid e della deviazione standard della velocità, in quanto le singole velocità tendono

ad uniformarsi. Ovviamente questo processo dipenderà dai dati forniti in input come riportato sopra.

5 Strategie di test

Una volta compilato il codice sorgente con CMake viene restituito in output l'eseguibile `simulation.t` che ha il compito di testare le funzionalità implementate utilizzando il framework DOCTEST. Questa è una parte essenziale del progetto in quanto consente di verificare il corretto funzionamento del programma e di testare il suo comportamento nei casi limite. Per esempio, la presenza di soli due boid rappresenta una situazione particolare in quanto il predatore non è più in grado di mangiare. Anche la presenza di un singolo boid è ambigua, in quanto le funzioni che calcolano i vari dati richiesti sollevano delle eccezioni. I test implementati mirano quindi a verificare il corretto comportamento del programma in queste situazioni. È dunque importante testare il programma in condizioni limite per cercare di trovare e correggere anche errori poco probabili. A questo fine le funzioni sono state testate anche nel caso i boids occupino la stessa posizione o siano ad una distanza uguale a quella di separazione o quella massima alla quale possano interagire con gli altri. Per testare il comportamento dell'evoluzione dei boid ai bordi della simulazione (la funzione `pacman`) si è poi scelto di posizionarne alcuni in questi punti limite per studiarne il comportamento. I test effettuati forniscono tutti un esito positivo dando così la certezza del corretto comportamento delle funzioni utilizzate, e, di conseguenza, di tutto il programma.

6 Conclusioni

Il programma non presenta errori durante la compilazione, lo stormo si muove in modo coeso e tutte le regole di volo vengono rispettate. Inoltre l'implementezione del predatore permette di spostare e spezzare lo sotrmo utilizzando la tastiera. La possibilità di modificare il numero di boid e di fermare l'evoluzione dello stormo aiuta a comprendere il comportamento del singolo boid rispetto a tutti gli altri. Manipolando i parametri delle regole di volo durante la simulazione è possibile osservare come aumentando il valore di s lo stormo tenda a disperdersi sempre più, mentre aumentando il valore di a, esso tenda ad assumere una conformazione sempre più uniforme e quasi geometrica. Questi risultati sono interpretabili come ulteriore conferma della correttezza della simulazione, in quanto una separazione maggiore è

sinonimo di dispersione, mentre con un allineamento maggiore è ragionevole pensare che ogni boid dello stormo tenda ad assumere una velocità allineata agli altri. Attraverso l'aggiornamento di distanza e velocità media è inoltre possibile capire meglio ciò che avviene durante l'evoluzione da un punto di vista più quantitativo.

7 Modifiche intercorse alla consegna precedente