Boids Simulation - C++

Emma Rubbi, Elisa Rodegher, Gaetano Valentino

21 dicembre 2024

1 Abstract

Esposizione del progetto "Boids Simulation" che si pone come scopo la simulazione visiva del comportamento collettivo di uno stormo di uccelli utilizzando il linguaggio C++ e l'ausilio della libreria grafica "SFML".

2 Fonti e Riferimenti

Il progetto si basa su un software sviluppato nel 1986 da Craig Reynolds [1] , che propone un modello elementare del comportamento dei singoli "boids" quando inseriti in un contesto di collettività.
L'algoritmo si basa su tre leggi, elencate in seguito e descritte graficamente nelle figure 1, 2, 3.

1. Separazione, che prevede l'aggiunta di una componente di velocità opposta al moto iniziale del boid, qualora esso si avvicini troppo ad altri boids.
2. Allineamento, che riguarda la tendenza dei boids a muoversi lungo un'unica direzione.
3. Coesione, che impone ai boids la tendenza al movimento collettivo verso il centro di massa dello stormo.

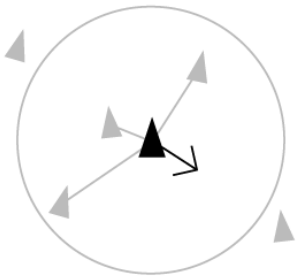

Figura 1: Separazione

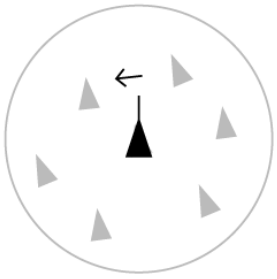

Figura 2: Allineamento

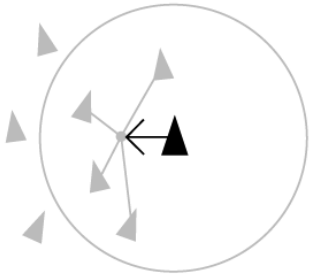

Figura 3: Coesione.

3 Struttura del progetto

Il progetto è stato suddiviso in più transition units, ognuna suddivisa in file header e codice sorgente:

- Operations: contiene la definizione di varie operazioni tra array bidimensionali (somma, prodotto per scalare, quadrato della norma).
- Boids: contiene la classe "boid", insieme alle funzioni che applicano le leggi di volo ai boid.
- Graphics: contiene le classi GraphicBoids e GrapichWind, le cui istanze sono rispettivamente le rappresentazioni grafiche dei singoli boids e la rappresentazione grafica della direzione del vento.
- Statistics: contiene le funzioni utilizzate per estrarre valori statistici dalla posizione e dalla velocità di ogni uccello nello stormo.

Tutte le funzioni e le classi sovracitate sono organizzate in un unico namespace chiamato "bds". Oltre a queste transition units, il progetto include:

- Codice sorgente della funzione main: se compilato insieme alle altre sorgenti restituisce il programma eseguibile.

- File “doctest.h” e codice di test: se compilato al posto del main restituisce i test di funzionamento delle varie parti del programma, eseguiti attraverso il programma “doctest”.
- File “CMakeLists.txt”: utilizzato per la compilazione del programma.
- Libreria grafica SFML: utilizzata per la parte grafica del programma.
- File ”.clang-format”: utilizzato per formattare il codice.

Si sono poi utilizzate delle ”Include_guards” in ognuno degli header files così da prevenire inclusioni multiple nella stessa transition unit.

4 Implementazione e funzioni

4.1 Gestione dello stormo

Abbiamo creato una apposita classe ”Boid”, la cui istanza è il singolo boid. I dati membri sono la posizione e la velocità, descritti come array bidimensionali.
La classe ”Boid” contiene un’interfaccia di funzioni membro, in cui risultano: costruttore, funzioni di modifica della velocità e della posizione, funzioni che estraggono velocità, posizione e angolazione.

Lo stormo è stato creato utilizzando un ”std::vector” di boid in modo tale da garantire una gestione collettiva dinamica e flessibile.

4.2 Velocità

L’applicazione delle regole sulla velocità (e delle velocità extra, vedere ”Aggiunte”) è stata fatta attraverso delle funzioni libere ”di calcolo”, che leggono il vettore stormo e restituiscono il contributo di velocità legato ad ogni regola. Le funzioni libere in seguito rimandano i loro valori ad una funzione che si occupa attivamente di modificare la velocità del boid.

BoidsAreNear: La gestione della variazione della velocità dei singoli boids si è basata dapprima su un “controllo dei boids vicini” gestito dalla funzione BoidsAreNear, che restituisce un valore booleano prendendo in input due boid e una distanza di vicinanza.
BoidsAreNear viene richiamata nell’applicazione delle leggi di Separazione e di Coesione.

Regole di volo: Le tre regole relative al volo sono applicate attraverso l’uso di funzioni libere. Esse utilizzano algoritmi della Standard Library per ”scorrere” il vettore stormo e calcolare i vari contributi.
A seguire sono riportate le leggi che costituiscono la base delle tre funzioni che governavano la velocità.

Separazione:
$\vec{v}_{sep} = -s \sum_{i \neq j} (\vec{x}_j - \vec{x}_i)$

Allineamento:
$\vec{v}_{alig} = a[\frac{1}{n-1}(\sum_{i \neq j} \vec{v}_j) - \vec{v}_i]$

Coesione:
$\vec{v}_{coes} = c[\frac{1}{n-1}(\sum_{i \neq j} \vec{x}_j) - \vec{x}_i]$

Dove

- x_i e v_i sono risp. l’array posizione e l’array velocità dell’i-esimo boid.
- ”s = sep_fact”, ”a = alig_fact”, ”c = coes_fact” sono tre parametri impostabili dall’utente.

4.3 Posizione ed Effetto Pacman

La posizione di ogni boid viene modificata unicamente in funzione della velocità del boid stesso e dell’intervallo discreto di tempo trascorso tra un’iterazione e l’altra (che viene sincronizzato con la frequenza di refresh della finestra grafica). La sola eccezione a questa regola è la funzione ”Effetto Pacman” che si occupa di gestire il comportamento dei boid quando essi superano i confini dello spazio di movimento. Come suggerito dal nome, il boid che supera i confini viene ”teletrasportato” dalla parte opposta, come se la zona di movimento fosse toroidale.

5 Aggiunte

Oltre all’implementazione base del programma, sono state fatte alcune aggiunte extra.

5.1 Vento e random boost

Sono stati aggiunti due contributi alla velocità del singolo boid.

- "Random boost" è una velocità di modulo sempre uguale (inizializzabile manualmente insieme ai vari coefficienti, oppure lasciata ad un valore di default) la cui orientazione viene però scelta in maniera casuale per ogni boid, a ogni iterazione.
- Il programma prevede un "vento" gestito da una classe apposita. Esso varia casualmente durante l'esecuzione del programma, e alcune sue caratteristiche sono manualmente impostabili nell'interfaccia di input (oppure possono essere lasciate ai valori default). Il vento fornisce a tutti i boid una leggera deviazione proporzionale alla sua intensità e direzione.

5.2 Finestra grafica

Abbiamo creato una finestra grafica che rappresenta lo stormo in movimento, strettamente connessa all'esecuzione del programma. Questa parte del programma è stata realizzata con l'uso della libreria grafica "SFML", presente nella cartella. La finestra mostra i boid in movimento e la direzione del vento (rappresentata tramite una freccia in un riquadro in alto a sinistra). Attraverso l'uso di SFML vengono anche gestiti tutti gli intervalli temporali (durata della simulazione, frequenza di stampa dei dati statistici, frequenza di cambio del vento).

6 Testing e Compilazione

Il testing del progetto è stato effettuato attraverso l'uso di "doctest.h", incluso nella cartella. Per compilare il programma in modalità testing è necessario eseguire il comando "g++ test_file.cpp boids.cpp operators.cpp statistics.cpp -o test -lsfml-graphics -lsfml-system -lsfml-window", che restituisce un file "test" eseguibile. Il file "test_file.cpp" contiene diversi test che controllano il funzionamento delle più importanti parti del programma in varie casistiche.
Per la compilazione del progetto si è optato per CMake, uno strumento che semplifica la gestione e la compilazione di progetti, creando i file di configurazione necessari per ambienti di sviluppo diversi.
A questo proposito è stato creato un file "CMakeLists.txt" che è servito a configurare le opzioni di compilazione, in cui sono state aggiunte delle flags per ottenere messaggi più dettagliati nella fase di debugging. Per compilare il progetto si suggerisce di seguire i seguenti steps:

1. Creare una sottocartella, comunemente chiamata "build", adibita a contenere i files generati dalla compilazione.
2. A partire dalla cartella /progetto/build, eseguire il comando "cmake ..", che crea i file di configurazione necessari per il sistema di compilazione.
3. Sempre nella cartella /progetto/build, eseguire il comando "make", che compila il codice sorgente e crea il file eseguibile "boids" in una sottocartella di build denominata "bin".
4. Dalla directory /progetto/build/bin, eseguire il file "boids" con il comando "./boids".

Appena il programma viene eseguito, compare un'interfaccia utente nella quale si richiede di inserire diversi parametri. Consigliamo di:

1. Impostare una distanza di vicinanza tra 15 e 45.
2. Impostare una distanza di separazione molto inferiore (intorno a 1-5).
3. Scegliere un fattore di separazione dell'ordine delle unità.
4. Scegliere un fattore di allineamento dell'ordine dei decimi/centesimi.
5. Scegliere un fattore di coesione dell'ordine dei millesimi.
6. Se inizializzata la velocità casuale, scegliere un modulo dell'ordine di qualche unità.

7 Risultati e conclusione

Il programma compila senza warnings e il comportamento collettivo dello stormo è conforme a quanto previsto dal modello.
Tuttavia, al momento della chiusura della finestra grafica, viene stampato a schermo il seguente errore, mostrato in figura 4. Non è stato possibile risalire alla causa del memory leak, ma come suggerito da alcuni compilatori, il problema sembra essere legato intrinsecamente alla libreria grafica "SFML".

```
=================================================================
==6087==ERROR: LeakSanitizer: detected memory leaks

Direct leak of 128 byte(s) in 1 object(s) allocated from:
    #0 0x7fb7ad2fcc38 in __interceptor_realloc ../../../../src/libsanitizer/asan/asan_malloc_linux.cpp:164
    #1 0x7fb7a6cf07a8  (<unknown module>)

Indirect leak of 56 byte(s) in 1 object(s) allocated from:
    #0 0x7fb7ad2fca57 in __interceptor_calloc ../../../../src/libsanitizer/asan/asan_malloc_linux.cpp:154
    #1 0x7fb7a785f7d5  (<unknown module>)

Indirect leak of 56 byte(s) in 1 object(s) allocated from:
    #0 0x7fb7ad2fca57 in __interceptor_calloc ../../../../src/libsanitizer/asan/asan_malloc_linux.cpp:154
    #1 0x7fb7a785f7be  (<unknown module>)

SUMMARY: AddressSanitizer: 240 byte(s) leaked in 3 allocation(s).
```

Figura 4: Rilevazione di un "Memory Leak" alla chiusura della finestra grafica

Si riporta in seguito, in figura 5, un frame della simulazione.

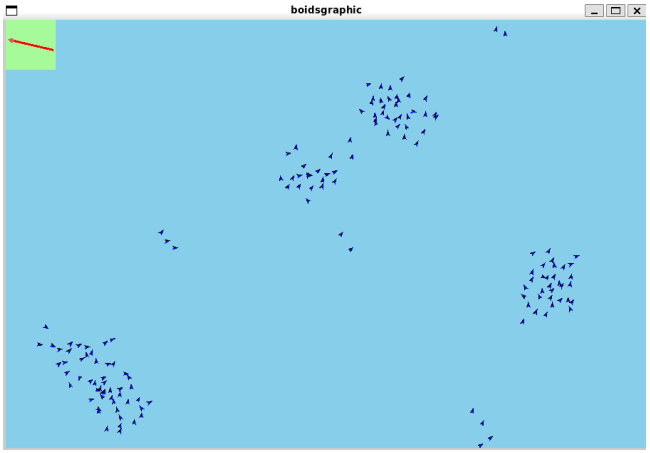

Figura 5: Frame della simulazione "Boids"

Riferimenti bibliografici

[1] Craig Reynolds, 1995, *Boids*, URL: `https://www.red3d.com/cwr/boids/`