Data: 14 Febbraio 2023

Simulazione del comportamento di stormi

Autori:
Vassallo Marco, Vignola Gabriele, Bici Giulio.

Contents

1 Introduzione

In questo progetto è stato sviluppato un algoritmo in grado di simulare il volo di uno stormo di uccelli (boids) basandosi su tre regole principali: coesione, separazione ed allineamento. In questo modo, attraverso l'utilizzo di una libreria grafica, è possibile osservare il fenomeno di formazione dello stormo.

2 Scelte procedurali e implementazione

Abbiamo deciso di creare sei translation unit, di cui due dedicate all'implementazione dei boid e del predatore, due dedicate alla creazione della geometria e alle regole di volo, una dedicata alla grafica e l'ultima contenente il main. Per lavorare in gruppo in modo più efficente ci siamo avvalsi di GitHub.

2.1 "Coordinates.cpp"

In questa translation unit sono state implementate le principali formule della geometria euclidea applicata ai vettori. Inizialmente si implementano i metodi getter & setter per manipolare ed ottenere le informazioni dai vettori;attraverso gli overloading degli operatori aritmetici è possibile svolgere le operazioni tra i vettori, invece con funzioni dedicate il prodotto scalare standard, calcolare l'angolo tra due vettori, ottenere la norma di uno e stampare a schermo le sue componenti

2.2 "Boid.cpp"

La classe Boid è la classe madre che permette di strutturare i Boid, dotati di attributi privati come le coordinate spaziali e le componenti della velocità.
Come metodi della classe si implemeneta un costruttore parametrico, i metodi getter & setter per poter lavorare con i Boids. La funzione *Wall_Hit* gestisce il comportamento ai bordi dei boids, sommando una velocità in direzione opposta in modo da deviare la traiettora dei Boid a cui è applicata.

2.2.1 "Predator.cpp"

La classe *Predator* è una classe derivata pubblicamente di *Boid*, pertanto eredita i membri pubblici di quest'ultima. Ovviamente possiede alcune caratteristiche aggiuntive come ad esempio il movimento "forzato" imponibile da tastiera tramite "$\uparrow \downarrow \rightarrow \leftarrow$" oppure, con l'utilizzo una variabile booleana, la possibilità di essere fermato e fatto ripartire rispettivamente con "C" ed "A".

2.3 "Flock.cpp"

Si implementano qui le tre principali regole di volo: separazione, coesione e allineamento. Attraverso il costruttore di classe si inseriscono i boids nel vettore *Flock_* con velocità e posizione casuali estratte da una distribuione normale con media 110 e deviazione standard 60, in modo analogo il predatore viene generato casualmente, così come gli ostacoli. le tre funzioni *V_Separation_i, V_Allignement_i, V_Cohesion_i* servono a gestire le regole di volo, insieme a *Predator_Repulsion* e *Obstacles _Repulsion*, che gestiscono le repulsioni

dovute agli ostacoli e al predatore.
Il calcolo del centro di massa, della velocità media e della distanza media viene gestito da funzioni apposite con l'ausilio di uno speciale Boid chiamato "*Medium*" e stampate a schermo

2.4 "Graphic.cpp"

Attraverso la libreria grafica SFML si crea la finestra di simulazione. Con la funzione angle si calcola l'angolo della velocità dei boid rispetto all'orizzontale, in modo da far ruotare lo sprite dei boids in quella direzione attraverso setRotation. Le altre funzioni gestiscono gli I/O e l'evoluzione della simulazione,descritte in modo più preciso nella sezione apposita.

2.5 "Main.cpp"

Nel main sono contenute le principali funzioni di input e output, quelle dedicate alla generazione di messaggi d'errore in caso di input invalidi e l'inizializzazione di un'istanza della classe *Game* in grado di richiamare la funzione *Run* per far partire la simulazione.

3 Istruzioni su come compilare, testare ed eseguire il progetto

3.1 Compilazione ed esecuzione

Per compilare il programma è necessario innanzitutto installare la libreria grafica (in Ubuntu) con il comando *sudo apt-get install libsfml-dev*.Dopodichè clonare la cartella con il comando *git clone* seguito dal link github contenente il progetto (in questo caso il link è *https://github.com/Mrcocvs02/Boid.git*). In questo modo si clona la cartella nella directory in cui ci si trova. Si entra all'interno della cartella appena scaricata e si può quindi compilare con *g++ Main.cpp Coordinates.cpp Flock.cpp Boid.cpp Predator.cpp Graphic.cpp -lsfml-graphics -lsfml-window -lsfml-system* ottenendo, all'interno della stessa cartella, un file eseguibile chiamato "a.out", ossia il nome che di default il compilatore gli attribuisce. Per cambiare nome al file, basterà aggiungere al precedente comando "-o" seguito dal nuovo nome, ad esempio per chiamare l'eseguibile "game" sarà sufficiente inserire il comando *Coordinates.cpp Flock.cpp Graphic.cpp Predator.cpp Boid.cpp Main.cpp -lsfml-graphics -lsfml-window -lsfml-system -o game*. Ora, digitando *./game*, partirà il programma.

Si specifica che per visualizzare correttamente la grafica è neccesario usare un X server; tra i vari software gratuiti è stato scelto *MobaXterm*.

3.2 Test

I test sono contenuti nel sorgente *Test.cpp* e sono stati organizzati per verificare la grandissima parte delle funzioni definite nei file "Coordinates.cpp","Boids.cpp" e "Flock.cpp". Per i file "Coordinates.cpp" e "Boid.cpp" questi ultimi sono stati effettuati in modo minimale per testare il semplice funzionamento del codice scritto. In "Flock.cpp" sono stati implementati dei test volti a studiare il diretto funzionamento nei casi comuni e nei casi limite delle funzioni presenti. In particolare la maggior parte dei test sono volti a studiare il corretto funzionamento delle regole standard dei boids e del predatore, ma anche delle regole riguardanti l'implementazione del muro e degli ostacoli. Per la compilazione è necessario eseguire il comando *g++ Test.cpp Boid.cpp Predator.cpp Coordinates.cpp Flock.cpp -o Test* nel terminale per produrre un eseguibile chiamato *Test*. Per eseguire i test è poi necessario eseguire il comando *./Test*.

4 Descrizione degli Input/Output

4.1 Input

Appena eseguito il programma con il comando *./game* verrà chiesto quanti boids si vuole inserire (quest'ultimi avranno posizioni e velocità generate casualmente e in modo autonomo) e in seguito,volendo usare i parametri settati di default, basterà premere "d", altrimenti premendo "s" si potranno scegliere a piacere. Successivamente apparirà nel terminale una breve spiegazione sulle funzionalità aggiuntive inserite nel programma, con i relativi tasti per richiamarle. Queste ultime, nel dettaglio, sono:

- tasto "P" mette in pausa la simulazione;
- tasto "Space" per far ripartire la simulazione;
- tasto "Esc" interrompe la simulazione;
- tasto "C" ferma il predatore;
- tasto "A" fa ripartire il predatore;
- tasto "↑" sposta verso l'alto il predatore;
- tasto "↓" sposta verso il basso il predatore;
- tasto "→" sposta verso destra il predatore;
- tasto "←" sposta verso sinistra il predatore;

4.2 Output

Dopo aver inserito tutti gli input necessari, si avvierà la parte grafica che costituisce l'output principale del programma. In una finestra che si aprirà autonomamente si avranno dei piccoli triangoli bianchi che rappresentano i boids (con le relative posizioni e velocità attribuite casualmente), i quali cercheranno di formare uno stormo e al contempo di allontanarsi dal predatore rappresentato un cerchietto rosso che, se non vengono premute le freccette direzionali, seguirà autonomamente il centro dello stormo. Gli ostacoli vengono rappresentati come dei cerchi marroni statici, della stessa dimensione del predatore.
In basso a sinistra si può notare un istogramma che mostra l'andamento della velocità media in tempo reale, con annessa dev.standard. Accanto, per indicare la posizione del centro di massa dei boids, troveremo un pallino rosso inscritto in una circonferenza il cui raggio indica la dev.standard.

Nel terminale invece si avranno le seguenti informazioni aggiornate in tempo reale (in ordine dall'alto verso il basso):

- "t" che indica il numero di tic (la frequenza è 30 tic al secondo);
- la distanza media;
- la dev. standard della distanza media;
- la velocità media;
- la dev. standard della velocità media.

5 Conclusioni

Usando le tre semplici regole di volo, si può notare la formazione di uno stormo coeso, non solo confermato graficamente, ma anche guardando i valori delle distanze e velocità medie, che risultano essere *quasi* costanti. L'introduzioni di regole aggiuntive, come il muro, gli ostacoli o un oggetto come il predatore, rende più realistica e varia la simulazione, rispetto alle semplici regole, senza incidere sul formarsi di uno stormo coeso ma solo sul tempo di formazione dello stesso.

References

http://www.red3d.com/cwr/boids/

Descrizione di "Boids" da parte dal suo creatore Craig Reynolds.