Relazione Progetto Boids

Francesco Bartoli

Contents

1 Introduzione

1.1 Scopo

Il programma ha come obiettivo quello di simulare in uno spazio bidimensionale il comportamento di stormi di uccelli in volo, che verranno indicati con il nome di *boids*.

1.2 Installazione

Le istruzioni su come compilare, testare, eseguire sono presentate nel README del progetto, riportato qui sotto:

Build instructions are for **Ubuntu 22.04**.

1.2.1 Prerequisites

SFML (2.5): Library for graphic representation.
TGUI (1.0): Library for graphic interface.

1.2.2 SFML and TGUI Installation

Install SFML:

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

Install TGUI:

```
sudo add-apt-repository ppa:texus/tgui
sudo apt update
sudo apt install libtgui-1.0-dev
```

1.2.3 Clone the Repository

```
git clone https://github.com/Evyal/boids.git
```

1.2.4 Build the Project

- **Create the build directory**

  ```
  mkdir build
  cd build
  ```

- **Configure CMake in Release mode**

  ```
  cmake .. -DCMAKE_BUILD_TYPE=Release
  ```

- **Build the project**

  ```
  cmake --build .
  ```

1.2.5 Running the program

```
./boids
```

---

2 Struttura del programma

Segue una breve descrizione delle principali scelte progettuali e implementative del programma.

I file del progetto sono organizzati in sottocartelle, nello specifico i `.cpp` si trovano nella `/source`, mentre i rispettivi header sono situati nella `/include`. La directory `/testing` contiene i file di testing, e infine in `/assets` sono presenti alcuni file necessari per il corretto funzionamento del programma.

2.1 Regole di volo

I *boids* si seguono delle regole di volo, che ne determinano il comportamento. Ad ogni istante, il programma modifica le velocità e le posizioni dei *boids* attraverso le seguenti formule:

$$\vec{v}_{bi} = \vec{v}_{bi} + \vec{v}_S + \vec{v}_A + \vec{v}_C + \vec{v}_R$$

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

Dove $\vec{v}_S$, $\vec{v}_A$, $\vec{v}_C$, e $\vec{v}_R$ sono rispettivamente:

2.1.1 Separazione

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

2.1.2 Allineamento

$$\vec{v}_A = a \left(\frac{1}{n-1} \sum_{j \neq i} \vec{v}_{b_j} - \vec{v}_{b_i} \right) \quad \text{se} \quad \left|\vec{x}_{b_i} - \vec{x}_{b_j}\right| < i$$

2.1.3 Coesione

$$\vec{x}_c = \frac{1}{n-1} \sum_{j \neq i} \vec{x}_{b_j} \quad \text{se} \quad \left|\vec{x}_{b_i} - \vec{x}_{b_j}\right| < i$$

$$\vec{v}_C = c(\vec{x}_c - \vec{x}_{b_j})$$

Dove $n-1$ assume valori dipendenti dal numero di boid nel range di interazione, e non è un valore fisso dipendente dal numero di boids nello stormo.

E s, ds, a, c, i sono parametri della simulazione, e nel progetto sono indicati con i nomi di *separationStrength*, *separationRange*, *alignmentStrength*, *cohesionStrength* e *interactionRange*.

2.1.4 Repulsione

Questa regola in termini di formula è analoga a quella della separazione e determina l'allontanamento tra *boids* di stormi differenti introducendo due nuovi parametri r, dr, che nel progetto sono indicati con i nomi di *repelStrength* e *repelRange*.

2.1.5 Interazione *On Click*

$$\vec{v} = \pm p \sum_{j} (\vec{x}_{b_j} - \vec{x}) \quad \text{se} \quad \left|\vec{x}_{b_j} - \vec{x}\right| < i$$

Anche questa regola ha una formula analoga a quella della separazione e permette all'utente di interagire con i boids. Il $\pm$ è dovuto al fatto che questa interazione può essere sia attrattiva che repulsiva mentre $\vec{x}$ è il punto in cui l'utente ha cliccato. Il parametro che determina l'intensità di questa interazione prende il nome di *clickStrength* all'interno del progetto.

2.2 Files

Tutti i file del progetto sono stati inseriti nel namespace `ev`. Tutti i file menzionati di seguito come `.cpp` hanno un corrispettivo header.

2.2.1 constants.hpp

Namespace che contiene valori come limiti di velocità o posizione per i *boids*, parametri di interazione di default, o ulteriori valori per l'inizializzazione degli elementi dell'interfaccia grafica.

2.2.2 structs.hpp

Struct per impacchettare dei valori usati per inizializzare bottoni o altri elementi di interfaccia grafica

2.2.3 boid.cpp

File di implementazione della classe Boid e funzioni ausiliare per gestirne il comportamento

2.2.4 flock.cpp

File di implementazione della classe Flock che determina la struttura collettiva dei *boids* all'interno di uno stormo.

2.2.5 random.cpp

File che si occupa della generazione di numeri casuali

2.2.6 statistics.cpp

File che si occupa del calcolo delle statistiche restituite a schermo, riguardanti valori medi e deviazioni standard delle posizioni e velocità dei *boids*.

2.2.7 graphics.cpp

Breve file che contiene una funzione per la corretta rappresentazione grafica dei *boids*, ed un'altra che costruisce un rettangolo (`sf::Rectangle`) prendendo come parametro una delle struct definita nel file sopracitato.

2.2.8 switchbutton.cpp

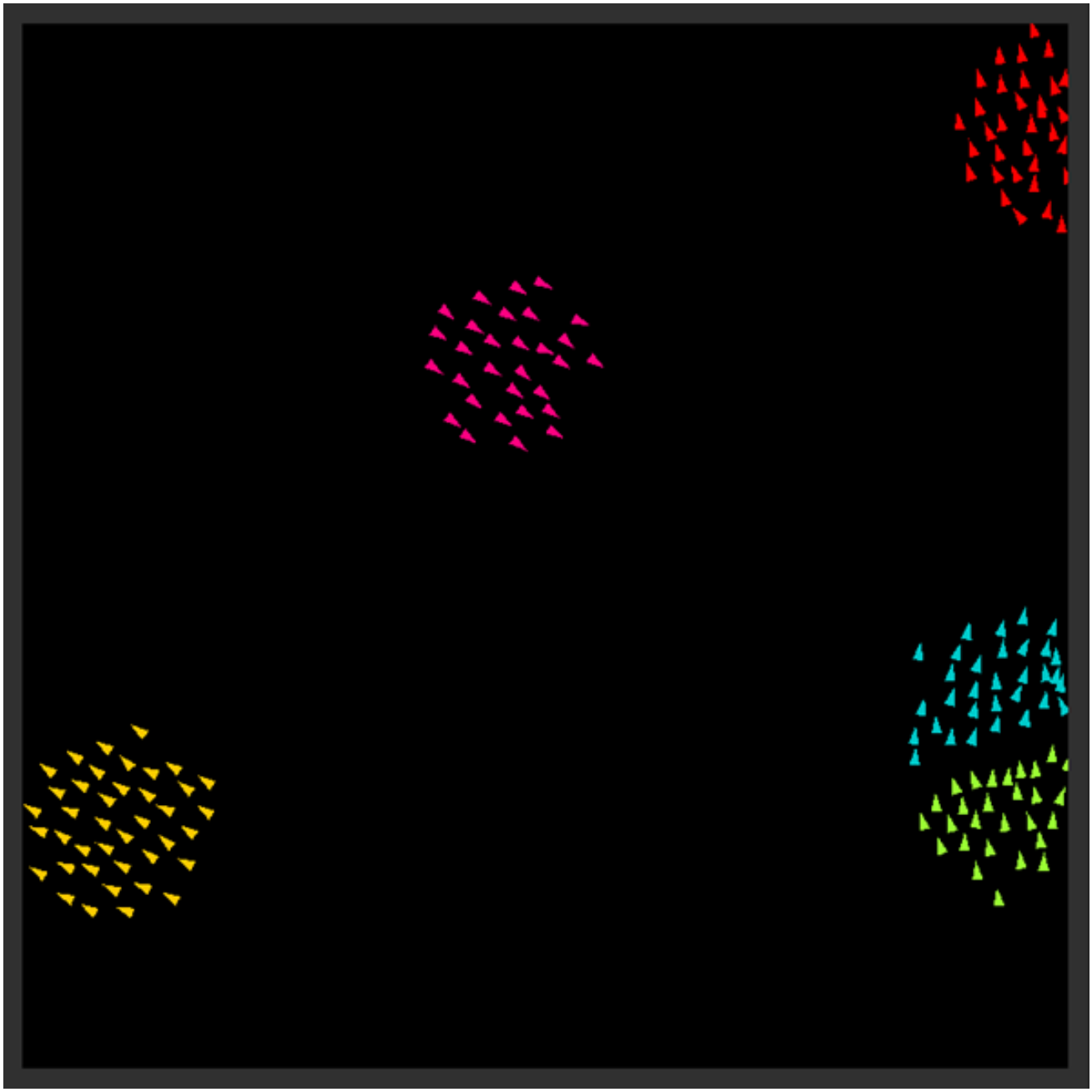
La classe Switchbutton introduce la funzionalità di un bottone che può trovarsi in due stati, non fornita da TGUI.

2.2.9 gui.cpp

Classe che introduce gli elementi necessari a costruire l'interfaccia grafica del programma, e si occupa di coordinare tutti i file che implementano la logica all'interno di essa.

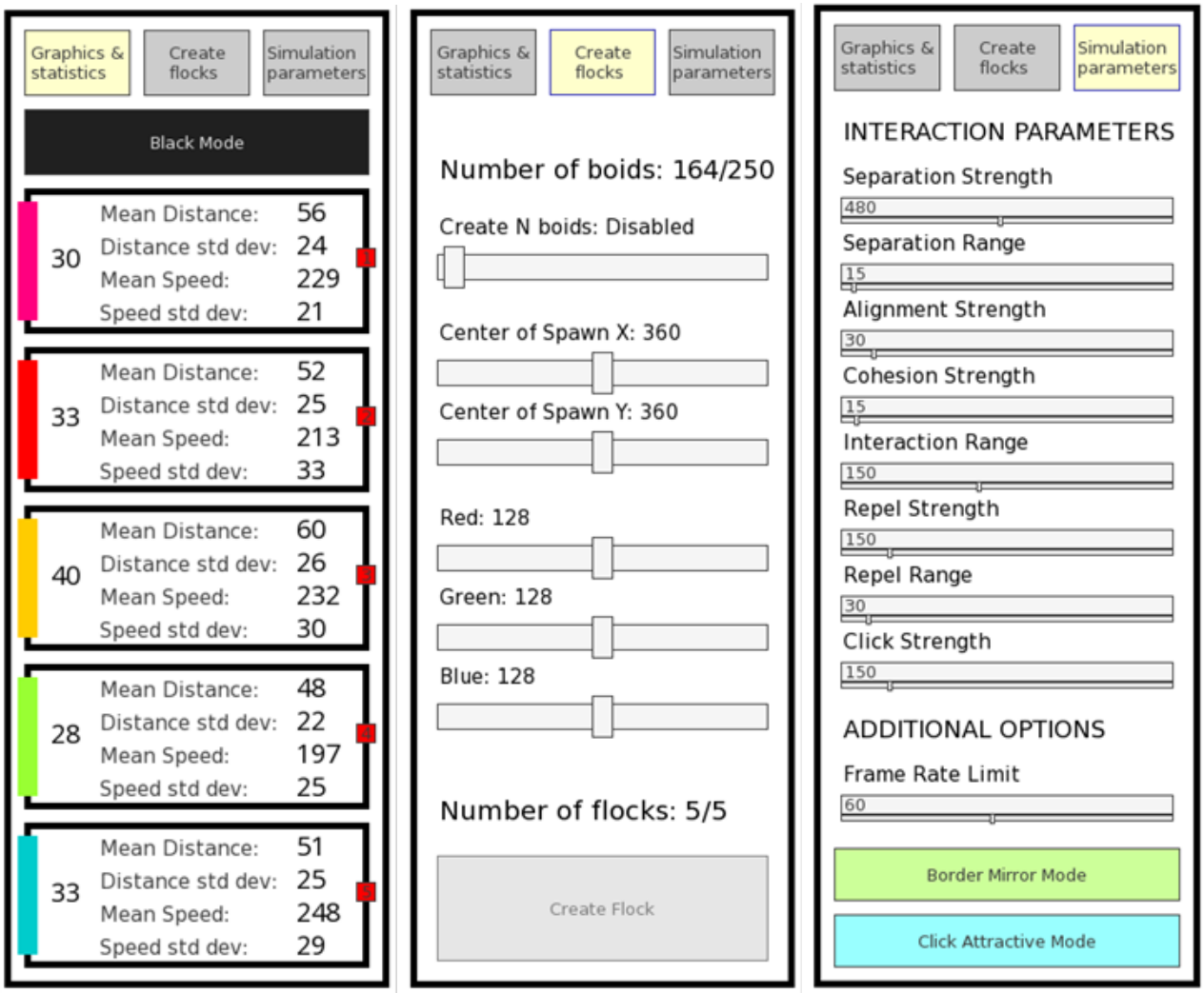
3 Interfaccia della simulazione

Anche questa parte deriva in parte dal README del progetto.



Features

- Real-time visualization of flock movement
- Real-time statistics display for each flock
- Interactive controls for adding or removing flocks
- Adjustable parameters for interactions between boids



3.1 Option 1: Graphics and statistics

- **Background color button**: Changes the colour of the background. (black and white)
- **Red numbered buttons**: Delete the corresponding flock.

3.2 Option 2: Create Flocks

- **Number of boids slider**: Selects the number of boids for a new flock.
- **Center of spawn sliders**: Select the spawn location of a new flock.
- **RGB sliders**: Select the color of a new flock. (Creating a white or black flock is disallowed because it would be invisible)
- **Create flock button**: Creates a new flock if there is enough space. (Max 250 boids; Max 5 flocks)

3.3 Option 3: Simulation Parameters

- **Interaction parameters sliders**: Change the values of the parameters of the rules that determine the movement of *boids*.
- **Border mode button**: Changes the behaviour of *boids* at the borders. (mirror or toroidal)
- **Click mode button**: Changes the interaction on click. (attractive or repulsive)

3.4 Key Controls

- **Left Click**: Interact with boids, attracting or repelling them to cursor.
- **Space Bar**: Pause/Resume simulation.

4 Testing

Tutti i file incaricati dell'implementazione di parte della logica del programma hanno un corrispettivo file di testing. Più precisamente, sono presenti i seguenti: `testboid.cpp`, `testflock.cpp`, `testrandom.cpp` e `teststatistics.cpp`.

Attraverso i test si è cercato di controllare che le classi, i metodi delle classi e le funzioni introdotte fossero esenti da errori e mostrassero il comportamento atteso. Sono stati eseguiti test in casi semplici per poter stabilire il funzionamento corretto del codice, e anche in alcuni casi particolari quando ritenuto necessario.

Il framework che si è utilizzato per creare le testing unit è doctest.h, il cui file è incluso nella cartella nel progetto (`/assets/doctest.h`). Questa libreria è in grado di generare autonomamente un main e permette l'esecuzione dei test semplicemente includendo il file sopracitato.

4.1 Eseguire i test

Per potere eseguire i test è necessario trovarsi nella cartella dove vengono prodotti gli eseguibili dei file precedentemente menzionati, seguendo i passaggi elencati sotto. Immaginando di trovarsi nella cartella principale dov'è contenuto il progetto:

```
cd build
cd testing
```

E digitare il comando corrispondente al test che si vuole eseguire:

```
./testboid
./testflock
./testrandom
./teststatistics
```

O eventualmente eseguendoli tutti in una volta utilizzando il seguente comando

```
ctest
```

5 Descrizione dei risultati

6 Links

Link alla repository di github utilizzata durante la realizzazione del progetto.

`https://github.com/Evyal/boids`