Simulazione di Boids con OpenMP

Silviu Robert Plesoiu - 7051276

September 5, 2024

1 Introduzione

La simulazione di Boids è un esempio classico di simulazione di comportamento di gruppi di agenti autonomi. Fu introdotta da Craig Reynolds nel 1987 e prevede tre regole principali:

- **Separazione**: Evita che i Boids si ammucchino, mantenendo una certa distanza dagli altri.
- **Allineamento**: Tende a far sì che ogni Boid si muova nella stessa direzione dei suoi vicini.
- **Coesione**: Fa sì che i Boids si muovano verso la posizione media dei loro vicini.

In questa relazione, mostriamo come abbiamo implementato una simulazione di Boids utilizzando la libreria SFML per la visualizzazione grafica e OpenMP per la parallelizzazione delle regole. Abbiamo anche testato il miglioramento in termini di tempo di esecuzione con diversi numeri di Boids.

1.1 Pseudocodice delle Tre Regole

Per dare una panoramica di come funzionano le tre regole principali nel nostro programma, ecco il loro pseudocodice:

Separazione:

```
Per ogni boid B:
    Per ogni vicino V di B:
        Se distanza(B, V) < soglia:
            Evita V
```

Allineamento:

```
Per ogni boid B:
    Calcola la direzione media dei vicini
    Allinea la direzione di B a quella media
```

Coesione:

```
Per ogni boid B:
    Calcola la posizione media dei vicini
    Muovi B verso la posizione media
```

2 Implementazione con SFML

Abbiamo utilizzato SFML (Simple and Fast Multimedia Library) per implementare la simulazione e visualizzare il movimento dei Boids. SFML è una libreria open-source che permette di gestire finestre, eventi, grafica e suono in modo molto efficiente.

Nella nostra simulazione, ogni Boid è rappresentato da un cerchio verde, e le tre regole vengono applicate in parallelo usando OpenMP. L'interfaccia grafica mostra, inoltre, informazioni come il numero di Boids, la regola applicata e il tempo impiegato per calcolare la regola.

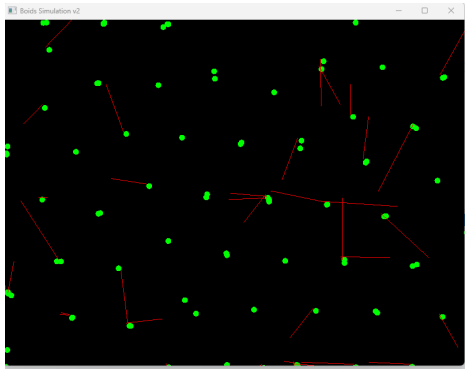

Figure 1: Simulazione con 100 Boids

3 Parallelizzazione con OpenMP

Per ottimizzare l'algoritmo, abbiamo parallelizzato la funzione `applyRules` utilizzando OpenMP. In particolare, la direttiva `#pragma omp parallel for` ci ha permesso di eseguire in parallelo il calcolo delle regole su ciascun Boid. Questo ha ridotto significativamente il tempo di calcolo con un grande numero di Boids.

Ecco un esempio del codice per il calcolo della velocità media parallelizzato:

```
float totalSpeed = 0.0f;
#pragma omp parallel for reduction(+:totalSpeed)
for (int i = 0; i < boids.size(); ++i) {
    totalSpeed += std::sqrt(boids[i].velocity.x * boids[i].velocity
    .x +
                            boids[i].velocity.y * boids[i].velocity
    .y);
}
```

Listing 1: Parallelizzazione del calcolo della velocità media

4 Risultati e Benchmark

Abbiamo eseguito la simulazione con diversi numeri di Boids, confrontando i tempi di esecuzione con OpenMP abilitato e disabilitato. I risultati mostrano che l'uso di OpenMP migliora sensibilmente i tempi di calcolo con un numero elevato di Boids.

Numero di Boids	Tempo (senza OpenMP)	Tempo (con OpenMP)
100	0.000465s	0.000919s
500	0.005138s	0.002853s
1000	0.018215s	0.007647s

Table 1: Tempi di esecuzione con e senza OpenMP

Come si può osservare dalla tabella, l'effetto della parallelizzazione diventa particolarmente evidente con un numero di Boids superiore a 500.

5 Conclusioni

In questa relazione abbiamo descritto l'implementazione di una simulazione di Boids usando SFML e OpenMP. L'uso della parallelizzazione ha dimostrato un miglioramento significativo nei tempi di calcolo, in particolare per un grande numero di Boids. Questo approccio potrebbe essere esteso ulteriormente ottimizzando altre parti dell'algoritmo.