

Relazione progetto Programmazione Avanzata

Introduzione

Il progetto sviluppato simula una gara di Formula 1, ispirata al classico gioco di carta e matita, utilizzando bot che corrono su un tracciato. Il progetto è organizzato in modo tale che ogni componente svolga un ruolo specifico e collabori con gli altri componenti per una simulazione completa e dinamica. Attualmente, il progetto raggiunge il livello medio di implementazione, fornendo una semplice interfaccia grafica dove viene mostrato l'avanzamento dei bot in modo interattivo.

Responsabilità Assegnate e Implementazione

Gestione e creazione del Tracciato: La gestione del tracciato è fondamentale per la simulazione. Il tracciato viene caricato da un file di configurazione JSON tramite la classe *TrackFactory*, che crea un'istanza di *DefaultTrack*. Questa componente garantisce che il tracciato sia configurato correttamente e che le posizioni di partenza e arrivo siano definite chiaramente. La classe *DefaultTrack* è progettata per rappresentare un tracciato immutabile all'interno di un gioco, gestendo la disposizione spaziale del tracciato e definendo le posizioni di partenza e arrivo.

Operazioni sul Tracciato: Le operazioni sul tracciato, come la verifica delle posizioni e la gestione delle mosse, sono gestite tramite la classe *DefaultTrackOperation*. Questa componente assicura che i bot possano muoversi correttamente sul tracciato, verificando la validità e la percorribilità delle posizioni e del tracciato e calcolando le mosse vicine (gli 8 vicini di una data posizione).

Gestione e creazione dei Bot: I bot rappresentano i concorrenti nella gara e sono gestiti tramite la classe *DefaultBot* e l'interfaccia *Bot*. La classe *DefaultBot* permette di calcolare la mossa successiva che un bot dovrà fare e permette di aggiornare correttamente le posizioni relative ad un bot. La classe *BotFactory* è responsabile della creazione dei bot tramite un file di configurazione JSON, consentendo una configurazione flessibile e personalizzata dei bot.

Movimento dei Bot: I bot hanno una strategia di movimento, definita nell'interfaccia *Movement* e implementata dalla classe *DefaultMovement*, che permette ai bot di muoversi in modo coerente e realistico durante la gara, permettendo quindi di poter accelerare, decelerare e calcolare il punto principale per il movimento. Per semplicità, è stata introdotta una *velocità massima* predefinita nella classe *DefaultMovement* per simulare la massima velocità che un'auto può raggiungere.

Visualizzazione Gara: La visualizzazione della gara è gestita dal motore di gioco, implementato nella classe *DefaultGameEngine*. Questa componente è responsabile per l'avvio della gara, aggiornamento dello stato della gara, visualizzazione dello stato attuale e verifica se la gara è terminata. Il motore di gioco interagisce con i bot per calcolare le loro prossime mosse, eliminare i bot che hanno subito un incidente (bot che sono finiti fuori pista poiché non hanno mosse valide in cui spostarsi, esempio quando accelerano troppo in prossimità di un ostacolo o quando si muovono verso un angolo) e determinare il vincitore della gara. Il costruttore *DefaultGameEngine* accetta come parametro il percorso del file JSON, per la creazione del tracciato e dei bot. Inoltre, *DefaultGameEngine* utilizza uno *ScheduledExecutorService* per gestire l'aggiornamento periodico della gara e permette di impostare un tempo di delay tra un turno e l'altro.

Gestione delle Eccezioni: Le eccezioni specifiche del bot e del tracciato sono gestite dalle classi *BotException* e *TrackException*. Queste classi rappresentano gli errori relativi alla configurazione dei bot e del tracciato, permettendo di identificare e gestire problemi che possono sorgere durante la configurazione e/o l'esecuzione della gara.

Posizioni nel tracciato: La classe `Position` rappresenta una posizione immutabile con coordinate `x` e `y`, utilizzata per tracciare le posizioni dei bot sul tracciato.

Implementazione interfaccia grafica ([JavaFX](#))

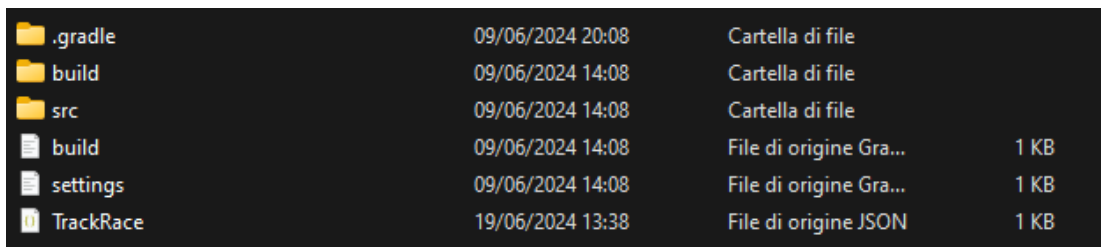
Simulazione della Gara: La visualizzazione della simulazione della gara è gestita dall'interfaccia `RaceDisplay` e dalla sua implementazione `DefaultRaceDisplay`. Queste componenti sono responsabili della rappresentazione visiva della gara compreso il disegno del tracciato e l'aggiornamento delle posizioni dei bot in tempo reale. La classe `App` inizializza la GUI e avvia la simulazione, fornendo un'interfaccia utente interattiva per monitorare la gara.

Gestione della Gara: Il controllo e la gestione della gara sono supportati dall'interfaccia `RaceManager` e dalla sua implementazione `DefaultRaceManager`. Questi componenti gestiscono l'avvio della gara, controllano il completamento della gara e determinano se la gara è terminata (facendo riferimento al motore di gioco principale "`GameEngine`"). Il `DefaultRaceManager` utilizza uno `ScheduledExecutorService` per gestire l'aggiornamento periodico della gara e permette di impostare un tempo di delay tra un turno e l'altro.

Le classi e le interfacce relative all'implementazione con JavaFX si trovano nella cartella "`app`" del progetto, mentre le restanti si trovano nella cartella "`api`".

File JSON

Nella cartella "`buildSrc`" del progetto è presente un file JSON (salvato "`TrackRace.json`", come mostrato in figura) con una configurazione base della gara. Questo file può essere modificato a piacimento per adattare il tracciato, le posizioni di partenza e arrivo, e i bot partecipanti, ciò permette di personalizzare la gara secondo le proprie esigenze, facilitando l'aggiunta di nuovi bot o la modifica del layout del tracciato per creare nuovi scenari di gara.



.gradle	09/06/2024 20:08	Cartella di file	
build	09/06/2024 14:08	Cartella di file	
src	09/06/2024 14:08	Cartella di file	
build	09/06/2024 14:08	File di origine Gra...	1 KB
settings	09/06/2024 14:08	File di origine Gra...	1 KB
TrackRace	19/06/2024 13:38	File di origine JSON	1 KB

Struttura del file JSON per la configurazione del tracciato

Il file di configurazione JSON per il tracciato deve includere due elementi principali: il layout del tracciato e la lista dei bot che gareggiano. Il layout del tracciato è rappresentato da una matrice bidimensionale dove ogni cella ha un valore specifico:

1. Non percorribile (indicato con 0)
2. Percorribile (indicato con 1)
3. Posizioni di partenza (indicate con 2)
4. Posizioni di arrivo (indicate con 3)

Esempio:

```
Json
{
  "track": [
    "00000000000000000000000000000000",
    "0211111111111111111111111111110",
    "0211111111111111111111111111110",
    "00000000000000000000000011111110",
    "000000000000111111111111111110",
    "000000001111111111111111111110",
    "011111111111100000000000000000",
    "011111111111111111111111111130",
    "011111111111111111111111111130",
    "000000000000000000000000000000"
  ],
  "bots": [{"name": "Bot1"}, {"name": "Bot2"}]
}
```

Considerazioni finali

In futuro, ci sono diverse migliorie che potrebbero essere implementate per rendere il codice della simulazione più realistico e sofisticato. Un esempio di miglioramento è l'ottimizzazione del comportamento dei bot quando si trovano di fronte a un vicolo cieco: invece di scegliere la prossima posizione in modo casuale, i bot potrebbero essere dotati di una logica più avanzata che consente loro di determinare il percorso più efficiente per avvicinarsi al traguardo. Un'altra miglioria riguarda il controllo delle collisioni tra i bot. Attualmente, le collisioni tra bot non vengono gestite, ma l'implementazione di questa funzionalità potrebbe contribuire a una simulazione più realistica.

Inoltre, sarebbe interessante implementare diverse strategie di movimento per ciascun bot. Attualmente, tutti i bot seguono la stessa strategia, ma in futuro si potrebbe introdurre una varietà di comportamenti. Ad esempio, un bot potrebbe adottare una strategia più aggressiva accelerando spesso, mentre un altro potrebbe essere più conservativo. Questo aggiungerebbe varietà e complessità alla simulazione, rendendo la gara più dinamica e interessante.