Relazione Progetto Metodologie di Programmazione Giugno/Settembre 2024

Lorenzo Marcantognini
Matricola: 122754

Corso di laurea triennale in Informatica (L-31)
Università degli Studi di Camerino
Anno Accademico 23/24

Specifica del Progetto:

Formula 1 è un gioco di carta e matita che si gioca su un foglio di carta quadrettata, sul quale viene disegnato un circuito automobilistico di fantasia, con una linea di partenza e una linea di arrivo (che possono anche coincidere se il circuito è circolare). Il gioco simula una gara tenendo conto dell'inerzia dei veicoli (per esempio, è necessario frenare quando si affronta una curva).

Una completa descrizione del gioco e delle sue regole è disponibile al seguente link.

L'obiettivo del progetto è quello di realizzare le classi che permettano di gestire una gara dove *giocatori interattivi* (umani) e *giocatori bot* (programmati) concorrono per vincere la gara.

E' possibile consegnare il progetto a tre diversi livelli di sviluppo: base, media, avanzata.

Sviluppo Base (Valutazione massima 24)

Il progetto consegnato a questo livello di sviluppo dovrà mettere a disposizione le interfacce e le classi che implementano i seguenti elementi:

- Il *tracciato di gara*;
- I *giocatori bot*;
- Il *motore di gioco.*

L'applicazione sviluppata dovrà essere in grado di:

- Caricare da file la configurazione del tracciato insieme alla lista dei giocatori bot presenti (il formato deve essere definito dallo studente);
- Fornire il supporto alla *simulazione* di una gara;
- Mostrare tramite *console* l'avanzamento della gara (posizione dei giocatori nel tracciato e loro stato).

Implementazione Media (Valutazione massima 27)

In questo livello di implementazione, oltre alle funzionalità dello *sviluppo base*, dovrà essere sviluppata una (semplice) interfaccia grafica che consenta di visualizzare in modo interattivo l'evoluzione della simulazione.

Implementazione Avanzata (Valutazione massima 30 e Lode)

In questo livello di implementazione, oltre alle funzionalità dello *sviluppo base* e *medio*, dovrà essere estesa l'interfaccia, ed il motore di simulazione, in modo da poter considerare anche *giocatori interattivi* che si muovono secondo le indicazioni dell'utente.

Introduzione:

Questa relazione fornisce in modo abbastanza dettagliato una panoramica riguardante questo progetto basato sul gioco Formula 1.
Il documento descrive le istruzioni su come avviare il progetto e le responsabilità con le varie interfacce e classi che implementano queste ultime.

Responsabilità e Implementazioni:

1. Gestione dei Simboli del Tracciato

Gestire e riconoscere i diversi simboli nel tracciato, come la partenza, l'arrivo, i bordi e il tracciato stesso. Implementato da:

- ★ **TrackSymbol** (Enum per i vari simboli)
- ★ **iSymbol** (Interfaccia)

2. Gestione della Tracciato di Gara

Creare, gestire e rappresentare il tracciato di gara dentro il programma. Implementato da:

- ★ **RaceTrack** (Classe)
- ★ **RaceTrackReader** (Classe per la lettura del tracciato da file)
- ★ **iTrack** (Interfaccia)
- ★ **iTrackReader** (Interfaccia)
- ★ **iFileReader** (Interfaccia che trasmuta il file in lista di stringhe)

3. Gestione dei Giocatori Bot

Definire e gestire i giocatori bot, inclusa la lettura delle configurazioni dei bot da file e la gestione delle loro mosse durante la gara. Simulare il comportamento dei bot in base alla loro posizione, velocità e inerzia. Implementato da:

- ★ **Player** (Classe astratta per un giocatore generico)
- ★ **BotPlayer** (Classe)
- ★ **BasicBotManager** (Classe per calcolare la prossima mossa eseguita dai bot)
- ★ **BotReader** (Classe per la lettura dei bot da file)
- ★ **iBotManager** (Interfaccia)
- ★ **iBotReader** (Interfaccia)
- ★ **iPlayer** (Interfaccia)
- ★ **iFileReader** (Interfaccia che trasmuta il file in lista di stringhe)

4. Gestione delle Posizioni e Vettori di Movimento

Gestire le posizioni dei giocatori sulla traccia e calcolare le direzioni di movimento. Implementare la logica per calcolare i vettori di movimento durante la gara. Implementato da:

- ★ **Position** (Classe)
- ★ **Vector** (Classe)
- ★ **iPosition** (Interfaccia)
- ★ **iVector** (Interfaccia)
- ★ **MoveDirection** (Enum per le possibili direzioni)

5. Gestione delle Auto

Gestire le auto dei giocatori tenendo conto dell'accelerazione e del proprio colore rappresentativo. Implementato da:

- ★ **Car** (Classe)
- ★ **iCar** (Interfaccia)
- ★ **Color** (Enum per i possibili colori delle auto)

6. Gestione dei Movimenti

Definire e gestire le mosse che i giocatori possono effettuare, definendole in base alla posizione attuale e al vettore che le compone.Implementato da:

- ★ **Move** (Classe)
- ★ **iMove** (Interfaccia)

7. Gestione delle Mosse

Calcolare le successive mosse disponibili per i giocatori in base alla loro posizione attuale, al tracciato e agli altri giocatori. Considerare le regole fisiche del gioco, come evitare le collisioni. Implementato da:

- ★ **MoveCalculator** (Classe)
- ★ **iMoveCalculator** (Interfaccia)

8. Gestione dei File di Configurazione per i Giocatori e il Tracciato

Gestire il caricamento e la lettura dei file di configurazione per i bot e il tracciato di gara. Assicurarsi che i file di configurazione siano letti correttamente e che le informazioni siano utilizzate per inizializzare il gioco. Segnalare eventuali errori se i file di configurazione non sono trovati o sono malformati.Implementato da:

- ★ **AbstractFileTracker** (Classe astratta che trova il path di un file)
- ★ **TrackFileTracker** (Classe)
- ★ **BotFileTracker** (Classe)
- ★ **ResourceDirectoryFinder** (Classe che trova la cartella risorse)
- ★ **iFileTracker** (Interfaccia)
- ★ **iDirectoryFinder** (Interfaccia)

9. Configurazione del Gioco

Caricare e gestire le configurazioni del tracciato e dei giocatori bot dai file di configurazione. Assicurarsi che il tracciato e i giocatori siano configurati correttamente prima dell'inizio della gara. Implementare la logica per inizializzare la partita, piazzare i giocatori sulla linea di partenza e avviare la simulazione. Implementato da:

- ★ **GameConfigurator** (Classe)
- ★ **iGameConfigurator** (Interfaccia)
- ★ **RaceTrackReader** (Classe)
- ★ **BotReader** (Classe)

10. Motore di Gioco

Gestire la logica di gioco, inclusa la simulazione della gara, la gestione dei turni e la determinazione dei vincitori. Implementare la sequenza di gioco, garantendo che ogni giocatore effettui la propria mossa in ordine. Verificare il rispetto delle regole fisiche del gioco durante la simulazione ed in caso monitorare e gestire lo stato di gioco, come l'eliminazione dei giocatori che escono dal tracciato o che non rispettano queste ultime. Implementato da:

- ★ **GameEngine** (Classe)
- ★ **GameController** (Classe)
- ★ **iGameEngine** (Interfaccia)
- ★ **iGameController** (Interfaccia)

11. Utilità di Supporto

Fornire metodi ausiliari generali utilizzati in tutto il progetto, come la stampa di messaggi di debug e informazioni. Implementato da:

- ★ **iUtils** (Interfaccia)

12. Gestione della Visualizzazione della Gara (Console)

Visualizzare l'avanzamento della gara sulla console, inclusi aggiornamenti sulle posizioni dei giocatori e messaggi di stato. Mostrare informazioni rilevanti come traccia con le posizioni aggiornate, le velocità dei giocatori, gli incidenti e l’eventuale vittoria. Implementato da:

- ★ **UIRaceController** (Classe)
- ★ **iUIRaceController** (Interfaccia)
- ★ **iUIController** (Interfaccia)

Istruzioni riguardo il funzionamento del Progetto:

Per utilizzare il progetto installare Gradle e seguire i seguenti comandi da terminale all'interno della directory del progetto stesso: *Gradle build* e successivamente *Gradle run.*

L'applicazione ottiene i file .txt di configurazione di gioco situati dentro la cartella resources del progetto stesso, di base vi sono alcune tracce e lista di bot preconfigurate, ma nel caso si volesse provare a crearne un altro basta seguire le linee generali facilmente intuibili vedendo come sono strutturati gli altri file già presenti.

- | = Muro
- - = Linea di partenza
- _ = Traguardo
- . = Circuito

L'applicazione sceglie poi i file in base agli indici che l'utente inserisce ad inizio gioco così da poter configurare tutto correttamente.