AlbanesiAngelo123406_PDF_Progetto

1. Introduzione

Il presente documento descrive il processo di sviluppo, le decisioni progettuali e i dettagli di implementazione del progetto di un gioco vettoriale di Formula 1. Il gioco simula una gara di Formula 1 basata su vettori, con giocatori umani e bot che competono su un circuito disegnato su una griglia. Lo sviluppo del progetto ha seguito il pattern architetturale MVC, rispettando le specifiche richieste e principi come SOLID e i design pattern.

2. Informazioni Generali

- Il circuito viene caricato da un file di testo (.txt), in cui ogni carattere rappresenta un componente specifico del tracciato.
- Il progetto include 3 circuiti disponibili, selezionabili dall'utente all'avvio del gioco.
- Vince il primo giocatore che raggiunge una cella di arrivo, facendo terminare immediatamente la partita.
- I movimenti dei bot sono determinati da due algoritmi: un adattamento dell'algoritmo A* e l'algoritmo BFS.
- Il gioco è fruibile sia tramite interfaccia a riga di comando (CLI) sia con un'interfaccia grafica (GUI).
- I giocatori, umani e bot, vengono caricati da un file di testo separato.

3. Responsabilità del Sistema

1. Gestione del Dominio e delle Entità di Base

Responsabilità: Modellare il dominio applicativo, definendo le entità fondamentali (posizioni, vettori, celle, tracciati e relativi caricamenti).

Componenti coinvolti:

- **model.core.Position:** Rappresenta una posizione spaziale per tracciare i movimenti.
- **Vector:** Gestisce le direzioni e gli spostamenti.
- **CellType** e **model.core.AccelerationType:** Definiscono rispettivamente le tipologie di cella e le dinamiche di accelerazione.
- **Track** e **model.core.TrackLoader:** Si occupano della rappresentazione e del caricamento dei tracciati.

2. Coordinamento del Ciclo di Gioco e Validazione delle Regole

Responsabilità: Coordinare lo stato della partita, la gestione dei turni e la validazione dei movimenti (incluso il rilevamento di collisioni).

Componenti coinvolti:

- **model.game.GameState:** Rappresenta lo stato attuale della partita.

- **MovementManager** e **TurnManager:** Coordinano rispettivamente i movimenti e i turni.
- **model.game.validators.MoveValidator, PlayerCollisionValidator, WallCollisionValidator:** Implementano la logica di validazione dei movimenti e il controllo delle collisioni.

3. Gestione dell'Intelligenza Artificiale e delle Strategie

Responsabilità: Calcolare percorsi e strategie ottimali per i bot tramite pathfinding e strategie predefinite.

Componenti coinvolti:

- **model.ai.algorithms.astar.AStarNode:** Implementa il pathfinding con l'algoritmo A*.
- **model.ai.checkpoint.ICheckpointMap, ICheckpointStrategy, ICheckpointTracker:** Gestiscono checkpoint e strategie di movimento.
- **model.ai.services.IFinishLocator, IMoveValidator, IReservationService:** Forniscono servizi di supporto come localizzazione della cella di arrivo e validazione dei movimenti.
- **model.ai.strategies.IPathFinder:** Interfaccia per la ricerca dei percorsi.
- **AIStrategy:** Implementa le strategie di intelligenza artificiale.

4. Gestione dei Giocatori

Responsabilità: Gestire i partecipanti, includendo parsing, creazione e assegnazione delle strategie.

Componenti coinvolti:

- **model.players.Player:** Classe base per i giocatori.
- **BotPlayer:** Specializzazione per i bot.
- **PlayerFactory:** Crea e inizializza i giocatori.
- **PlayerParser:** Si occupa del parsing dei file di configurazione.

5. Gestione dell'Interfaccia Utente

Responsabilità: Gestire la comunicazione con l'utente attraverso interfacce testuali e grafiche.

Componenti coinvolti:

- **view.CLIView e CLIMenuManager:** Gestiscono l'interfaccia testuale.
- **view.gui.GUIView e GUIComponentFactory:** Implementano la GUI.

6. Coordinamento del Flusso Applicativo

Responsabilità: Assicurare la corretta interazione tra le componenti.

Componenti coinvolti:

- **controller.GameController:** Coordina le operazioni tra i moduli.

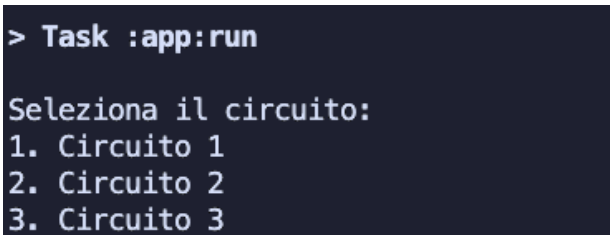
4. Avvio del Progetto

Il progetto può essere eseguito in modalità CLI o GUI.

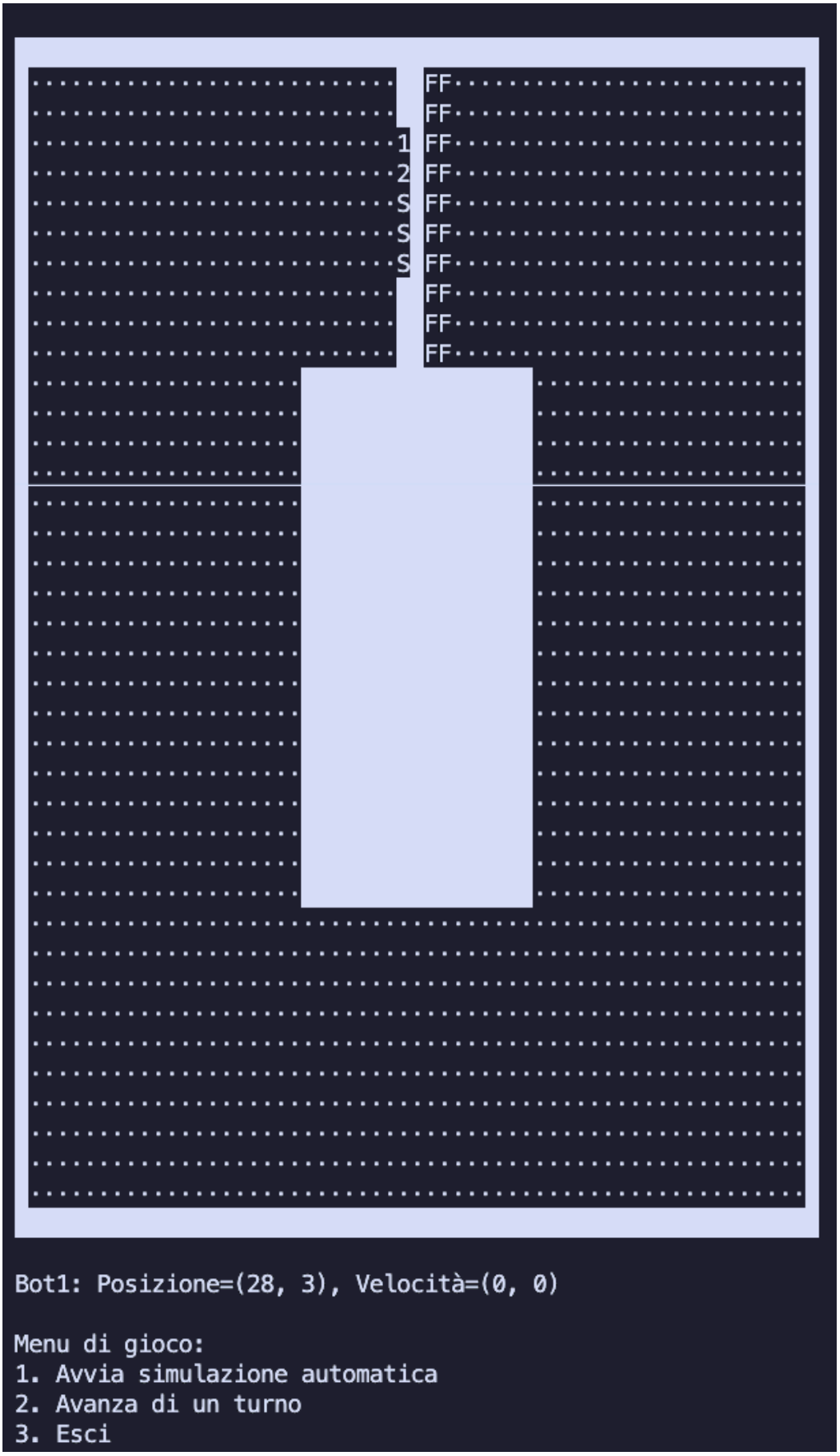
1. Modalità CLI

1. Aprire un terminale nella directory principale del progetto.
2. Eseguire `gradle build` per compilare il progetto.
3. Avviare l'applicazione con `gradle run`.
4. Seguire le istruzioni per selezionare il circuito e avviare la partita.

Nota: In modalità CLI possono partecipare solo giocatori bot.



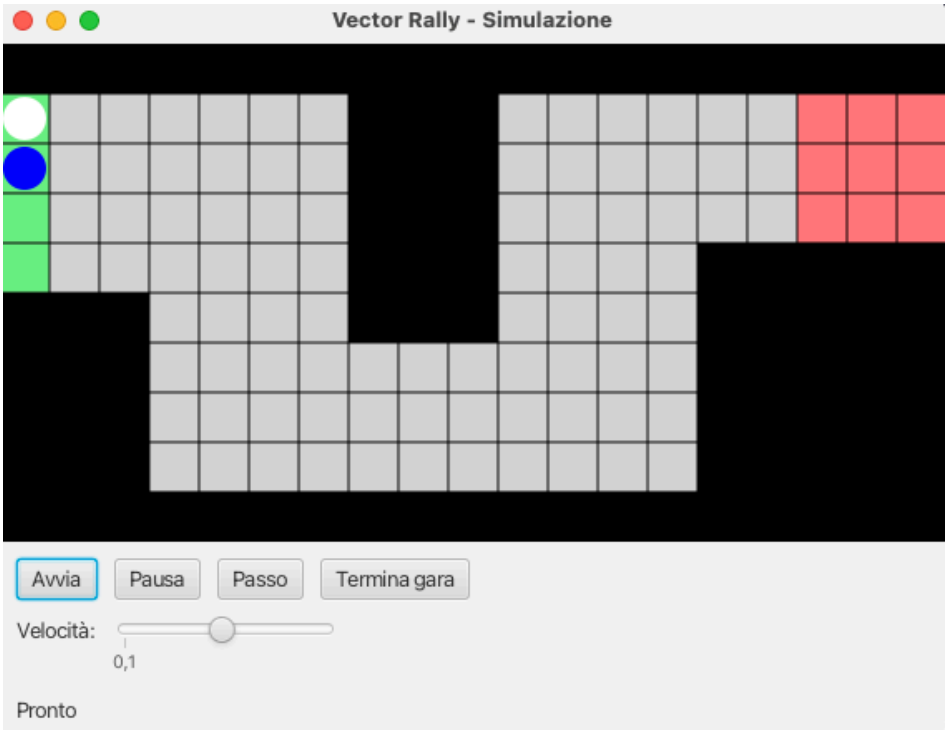
Esempio con il circuit1


FF
1
2
S
S
S
Bot1: Posizione=(28, 3), Velocità=(0, 0)
Menu di gioco:
1. Avvia simulazione automatica
2. Avanza di un turno
3. Esci

2. Modalità GUI

1. Aprire un terminale nella directory principale del progetto.
2. Eseguire `gradle build`.
3. Avviare l'applicazione con `gradle run --args="gui"`.
4. Selezionare il circuito dalla finestra grafica e avviare la gara.

Esempio con il circuit2



3. Configurazione dei Giocatori

I giocatori vengono caricati dai file:

- `playersCLI.txt` per la modalità CLI (solo bot).
- `playersGUI.txt` per la modalità GUI (bot e umani).

Il formato del file è il seguente:

```
Tipo;Nome;Colore (HEX);Strategia (1=BFS, 2=PureAStarStrategy)
```

Esempio per playersGUI.txt:

```
human;Human1;#FF0000
human;Human2;#00FF00
Bot;Bot3;#FFFFFF;1
Bot;Bot4;#0000F0;2
```

Esempio per playersCLI.txt:

```
Bot;Bot1;1
Bot;Bot2;2
```

Nota: Per circuit1 e circuit3 il numero massimo di giocatori è 5, mentre per circuit2 è 4.

5. Strategie dei Bot

Strategie Implementate

- **BFS (Breadth-First Search):** Algoritmo che esplora i nodi in ampiezza, valutando tutti i percorsi a una certa profondità prima di passare al livello successivo.
- **PureAStarStrategy:** Utilizza l'algoritmo A* per calcolare il percorso ottimale combinando costo già percorso (g) e costo stimato rimanente (h).

Estensione delle Strategie

Per aggiungere nuove strategie è necessario:

1. Creare una nuova classe nel package `model/ai/strategies` implementando l'interfaccia `AIStrategy`.
2. Implementare il metodo `getNextAcceleration(Player player, GameState gameState)`.
3. Aggiornare il metodo `createStrategy(StrategyType strategyType)` della classe `PlayerFactory` per includere la nuova strategia.
4. Configurare il nuovo tipo di strategia nei file `playersGUI.txt` o `playersCLI.txt`.