Università Degli Studi Di Camerino

Scuola Di Scienze e Tecnologie

Corso Di Laurea in Informatica

Vector Race

Relazione

Studente:

Necula Robert Gabriel

Matricola: 123390

Docente:

Prof. Michele Loreti

Data:

03/09/2024

Anno Accademico:

2023/2024

SOMMARIO

ANALISI DEL PROGETTO

L'obiettivo di questo progetto è ricreare il gioco "Formula 1", un gioco in cui viene disegnato un circuito e i giocatori competono per vincere la gara. La mia implementazione si basa sul terzo livello di sviluppo proposto.

In questo livello, viene richiesto di caricare un file contenente tutti i dati necessari per lo svolgimento della gara, che comprende la forma del circuito e i giocatori che vi partecipano.

Inoltre, dovrà essere sviluppata un'interfaccia grafica che consentirà agli utenti di interagire con il gioco. Attraverso questa interfaccia, i giocatori potranno selezionare le loro mosse e seguire in tempo reale l'andamento della gara, visualizzando i progressi sul circuito man mano che prendono decisioni.

Per facilitare lo sviluppo del progetto c'è stata una suddivisione in macroaree, ciascuna delle quali rappresenta una parte chiave del progetto.

Successivamente è stata effettuata una divisione più dettagliata per definire le responsabilità principali di ciascuna parte del gioco precedentemente individuata. Partendo con la prima suddivisione, sono state individuate quattro aree chiave del gioco:

1. *Gestione del file in input*: Questa area si occupa di tutte le responsabilità legate alla gestione del file contenente i dati della gara. Ciò comprende la lettura del file, la sua conversione in un formato supportato e la sua validazione per verificare che non ci siano errori nella struttura.
2. *Creazione della gara*: Quest'area riguarda tutte le responsabilità legate alla configurazione della gara, utilizzando le informazioni estratte dal file di input. Comprende la generazione del circuito, dei giocatori e delle regole che coordineranno la gara.
3. *Renderizzazione della gara*: Quest'area è responsabile dell'inizializzazione e gestione dell'interfaccia grafica. Si occupa di visualizzare il circuito e i giocatori, aggiornando graficamente l'andamento della gara in base alle mosse degli utenti.
4. *Gestione della gara*: Quest'area copre tutte le responsabilità relative alla gestione delle dinamiche di gioco. Si occupa di gestire gli input degli utenti, aggiornando i progressi dei giocatori e le condizioni del circuito, verificando che le regole siano rispettate e, nel caso non lo fossero, prendere altre decisioni.

ANALISI DELLE RESPONSABILITA

Per ogni punto chiave precedente, sono state individuate diverse responsabilità.

GESTIONE DEL FILE IN INPUT

Per questa parte, le responsabilità individuate sono:

1. *Lettura del file*: caricamento di un file scelto dall'utente, che dovrà avere un'estensione supportata dal motore di gioco.
   Per questa responsabilità è stato usato un FileChooser offerto da JavaFx, che apre una finestra di dialogo dove l'utente può scegliere un file a piacimento tra quelli supportati.
2. *Conversione del file*: una volta caricato, il file viene convertito (in base all'estensione) in un oggetto che contiene tutte le sue informazioni.
   Per questa responsabilità è stata creata un'interfaccia chiamata **FileParser**, che mette a disposizione il metodo *parseFile()* per convertire il contenuto del file in un oggetto di tipo **ParsedData**. Questo oggetto permette di accedere ai dati del file tramite il metodo *getData()*.
   La classe che implementa FileParser per i file con estensione ".txt" è TXTFileParser, la quale restituisce un oggetto di tipo TXTParsedData, che contiene i dati estratti dal file.
3. *Validazione del file*: dall'oggetto che contiene le informazioni del file, si procederà ad una validazione della struttura che verificherà che il file sia stato correttamente scritto e non ci siano errori.
   Per questa responsabilità è stata creata un'interfaccia **ParsedDataValidator**, che mette a disposizione il metodo *validateData()* che prende come parametro un oggetto di tipo ParsedData e definisce tutte le regole da rispettare per la struttura del file. Anche in questo caso, la classe che la implementa è TXTParsedDataValidator, che valida la struttura di un file ".txt"

Nel caso si volesse supportare un'altra estensione, tutte le interfacce precedentemente elencate dovranno essere implementate.

CREAZIONE (LOGICA) DELLA GARA

Per la creazione (logica) della gara, le responsabilità individuate sono:

1. *Creazione del circuito*: partendo dai dati del file, estrarre la parte relativa al circuito e convertila per mantenere solo le parti che fanno effettivamente parte del circuito, escludendo tutte le altre.
   Per questa responsabilità è stata creata un'interfaccia **CircuitSetup**: essa mette a disposizione il metodo *setup()* che prende come parametro un oggetto di tipo ParsedData ed estrae tutta la parte relativa al circuito, interpretando tutti i simboli all'interno e convertendoli in nodi, che rappresentano un punto all'interno del circuito; questa classe fornisce un oggetto **Circuit** che conterrà tutte le informazioni riguardanti il circuito.
   La classe che implementa quest'interfaccia è TXTCircuitSetup, che converte la parte del circuito di un file ".txt".

2. *Creazione dei giocatori*: partendo dai dati del file, estrarre la parte relativa ai giocatori per convertili in una lista di giocatori che parteciperanno alla gara.
   Per questa responsabilità è stata creata un'interfaccia **PlayerSetup**: anch'essa mette a disposizione il metodo *setup()* che prende come parametro un oggetto di tipo ParsedData ed estrae tutta la parte relativa ai dati dei giocatori, fornendo in una lista di giocatori. La classe che implementa quest'interfaccia è TXTPlayerSetup, che converte la parte de giocatori in un file ".txt".

3. *Creazione delle regole*: le regole vengono impostate partendo dall'interfaccia grafica, avendo a disposizione varie scelte per ogni tipo di regola.
   Per questa responsabilità è stata creata un'interfaccia **RaceHandler**: essa mette a disposizione il metodo *handle()* che gestisce una particolare situazione nel gioco nel caso questa situazione si verifichi. Le classi che implementano quest'interfaccia sono **CollisionHandler**, **CrashHandler** e **WinHandler**, delle classi astratte che raggruppano gli handler per categoria e che fungono da base per creare i veri e propri handler, che gestiscono effettivamente la gara.
   Ogni nuovo handler deve estendere una di queste classi astratte.

RENDERIZZAZIONE DELLA GARA

Per la renderizzazione della gara, le responsabilità individuate sono:

1. *Inizializzazione grafica della gara*: partendo dalla gara precedentemente impostata, viene creata la forma del circuito e i giocatori vengono messi sulla linea di partenza.
2. *Aggiornamento della grafica*: ogni volta che un giocatore fa una mossa, in base all'esito della mossa verranno effettuati vari aggiornamenti grafici sul circuito e sui giocatori, mostrando così l'andamento della gara.

Per entrambe le parti, è stata creata una classe astratta **SceneBuilder** che offre un base per un qualsiasi componente grafico. Ogni componente grafico deve avere un contenitore e un insieme di dati che saranno visualizzati al suo interno. Le classi che la implementano sono:

- **CircuitSceneBuilder**, che permette la visualizzazione della forma del circuito e offre metodi che permettono di visualizzare le mosse dei giocatori.
- **PlayerSceneBuilder**, che permette la visualizzazione della lista di giocatori che partecipano alla gara. Offre un metodo che permette la rimozione di un giocatore dalla lista di giocatori una volta che viene eliminato.
- **MovesGridSceneBuilder**, che permette la visualizzazione delle mosse possibili che il giocatore corrente può compiere in quel momento. Offre metodi che modificano questi valori in base al giocatore corrente.

GESTIONE DELLA GARA

Tutte queste responsabilità sono state divise in classi diverse, unite poi da un gestore della gara che racchiude e permette l'utilizzo dei vari gestori. Questo gestore è una classe chiamata **RaceManager**.

Per la gestione della gara, le responsabilità individuate sono:

1. *Gestione dei giocatori*: per ogni mossa fatta dai giocatori, le loro informazioni vengono aggiornate.

   Per questa responsabilità è stata creata una classe **PlayersManager**, che mette a disposizione metodi per la gestione e l'aggiornamento dei dati dei giocatori.

2. *Gestione del circuito*: per ogni mossa fatta, lo stato del circuito viene aggiornato.
   Per questa responsabilità è stata creata una classe **CircuitManager**, che mette a disposizione metodi per la gestione e l'aggiornamento dei dati del circuito.
3. *Controllo delle regole*: per ogni mossa fatta, le informazioni relative al circuito e al giocatore vengono utilizzate per verificare che le regole siano rispettate. Nel caso non lo siano, verranno eseguite altre azioni.
   Per questa responsabilità, per ogni mossa fatta, i dati del RaceManager vengono passati a tutti i RaceHandler precedentemente creati, che verificheranno lo stato del giocatore e del circuito dopo aver fatto quella mossa. Se nessun RaceHandler ha dovuto gestire la mossa, significa che sono state rispettate tutte le regole della gara.

ISTRUZIONI PER IL GIOCO

Il formato di file momentaneamente supportato è il file ".txt". Questo file, per essere valido, deve avere un formato ben preciso.

La prima riga dovrà essere ::CIRCUIT, che segnala l'inizio della sezione relativa al circuito.

Nelle successive 35 righe (dalla riga 2 alla riga 36 estremi compresi) verrà definita la forma del circuito. Ogni riga deve contenere esattamente 63 caratteri.

I caratteri momentaneamente supportati sono:

- "#": quel punto non fa parte del circuito.
- "@": quel punto fa parte del circuito.
- "+": quel punto fa parte della linea di partenza.
- "-": quel punto fa parte della linea di arrivo.

Utilizzando una combinazione di questi simboli si potrà creare una qualsiasi forma.

Per quanto riguarda i giocatori, la loro sezione comincia nella linea immediatamente successiva (linea 37): questa linea dovrà essere ::PLAYERS.

Nella linea successiva sono definiti il numero di player che partecipano alla gara: il formato di questa riga è "*xB yH*", dove x e y sono numeri non negativi che identificano rispettivamente il numero di bot e il numero di persone reali che giocano.

Infine, nell'ultima riga, si dovranno specificare le informazioni dei giocatori umani (se presenti).

Per ogni giocatore umano, deve essere definita una coppia "nome:colore", tutti separati da una virgola (se più di uno).

Una volta impostato correttamente il file, dovrà essere caricato attraverso il bottone "LOAD FILE" presente nell'interfaccia. Successivamente, si dovranno scegliere le regole disponibili per la gara. Una volta scelte anche le regole, si dovrà premere il bottone "START RACE". Se non ci sono errori nella struttura del file, verrà visualizzata la forma del circuito coi giocatori dentro.

Se ci sono errori, nella parte "LOG" verrà segnalata la loro presenza, specificando il tipo e la posizione nel file dell'errore.

In basso, al centro dello schermo, apparirà una griglia con nove bottoni, ognuno dei quali indica una mossa specifica che il giocatore attuale può fare. Per ogni bottone è specificato lo spostamento che subirà il giocatore scegliendo quella specifica mossa.

Un file già pronto all'uso è stato messo a disposizione: esso si chiama *ovalCircuitForSimulation.txt* e si trova nella cartella principale NeculaRobertGabriel123390.