RELAZIONE ANALISI E FUNZIONAMENTO DEL GIOCO:

FORZA 4

INDICE:

1 INTRODUZIONE

2 FASI DI PROGETTO

3 DESCRIZIONE DELLE CLASSI

4 REFERENTIDI SVILUPPO

INTRODUZIONE

Finalità ultima della presente relazione è volta allo scopo di introdurre, spiegare e analizzare il processo dietro il progetto del gioco digitale “Forza 4”, tratto dall’omonimo gioco da tavolo. Il gioco consiste in un incontro tra 2 giocatori su una griglia 6x7, nonché 6 righe e 7 colonne, all’interno delle quali deve essere eseguita una sequenza di 4 pedine consecutive dello stesso colore, in qualsiasi direzione, per vincere la partita, le possibili direzioni devono essere: Orizzontale, Diagonale (sia verso destra che verso sinistra), Verticale. I giocatori a turno scelgono di posizionare una pedina del proprio colore in una colonna; pertanto, non è concesso al giocatore di scegliere la riga, poiché essa verrà decisa automaticamente dal gioco in base alla presenza o meno di pedine all’interno delle righe sottostanti della stessa colonna. Se la colonna risulterà già piena, verrà chiesto al giocatore di optare per un’altra. Non essendo stata implementata un’interfaccia grafica, ai 2 giocatori verrà mostrata la griglia a inizio della partita e dopo ogni loro turno. Qualora la griglia fosse totalmente piena, ovvero che non si potrà più fare mosse per allineare 4 pedine, si arriva incontro a uno stato di pareggio, ovvero viene dichiarata la patta.

Il progetto ha un solo referente di sviluppo: Francesco De Angelis che ha eseguito tutto il progetto.

DESCRIZIONE E FUNZIONALITA’ DELLE CLASSI

All’interno di questa sezione verranno elencate le relative classi create nel progetto. Verranno spiegati i motivi della loro creazione e il loro utilizzo. Tutto ciò deriva da un’analisi tecnica avvenuta durante le fasi di sviluppo del progetto.

Esse sono nati da questi bisogni tecnici:

-Agli utenti finali serve la possibilità di avere ciascuno delle pedine con un colore ed un nickname da usare durante la partita per capire a quale giocatore spetta il turno e la possibilità ovviamente di posizionare le pedine. Quindi è stata necessaria la creazione della classe Player e della classe Mappa.

-Un altro bisogno è quello di avere una classe che gestisca: la creazione di una nuova partita, la ripresa da un salvataggio, turni. Per questo è stata creata la classe Game.

-Infine, si ha avuto la necessità di un sistema che gestisca i controlli delle pedine (per verificarne la loro consecutività, in caso ci sia) per dichiarare la vittoria di un giocatore o la patta di entrambi.

Per ogni classe verranno spiegati metodi e attributi:

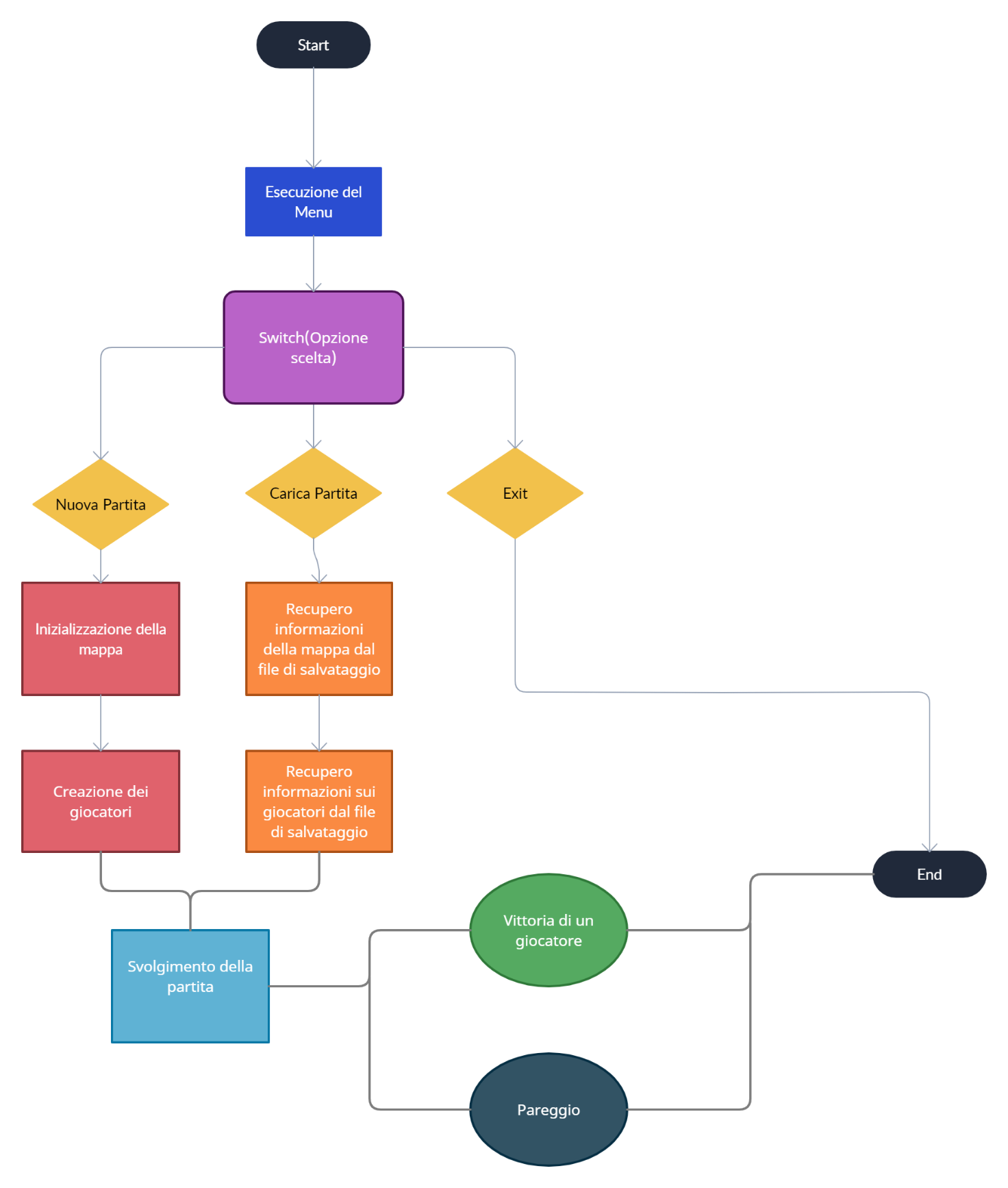
-Player: la finalità della classe è quella di rappresentare un giocatore che ha come attributi una Stringa con il nickname e un colore per identificare le sue pedine. Essa ha due getter che permettono di ritornare il nome e il colore e un metodo che ritorna una variabile booleana per verificare se viene aggiunto o meno la propria pedina nella griglia.

-Mappa: la sua funzione è quella di creare una mappa 6x7 (6 righe e 7 colonne), attraverso una matrice di Stringhe. Ogni cella verrà inizializzata con la parola empty per indicare che sono vuote, mentre per sostituire i “colori” nelle celle è stato implementato il metodo addColours con il quale, dando come argomento: riga, colonna e colore è possibile appunto modificare il colore di una cella, quindi aggiungere una pedina. È stato implementato un metodo atto a ricostruire la matrice dal file di salvataggio. Il metodo showMatrix permette una stampa a video della griglia per aiutare il giocatore a scegliere dove posizionare la pedina. Infine, ha un getter che permette di ricavare il colore della cella dando come argomento due interi che indicano la riga e la colonna.

Winner: la sua funzione è quella di impostare lo stato di vittoria del gioco, ovvero ha 2 attributi: uno booleano e uno stringa, i quali rispettivamente indicano la condizione di vittoria e il nome del vincitore. Ha un metodo principale: checkWin che controlla se uno dei due giocatori ha eseguito una sequenza in diagonale o orizzontale o verticale di 4 pedine, salvando il nome del giocatore che ha conseguito la vittoria. Il metodo checkWin esegue altri metodi che controllano se la vittoria è stata eseguita in direzione orizzontale(checkHorizzontal), diagonale (checkDiagonal che a sua volta esegue due metodi che controllano se la direzione della diagonale è verso destra(checkDiagonalRight) o verso sinistra(checkDiagonalLeft)) e infine verticale(checkVertical). Sono stati usati dei controlli per evitare inutili controlli. Infine, il metodo checkTie controlla lo stato di pareggio della partita.

Game: infine la funziona della classe Game è quello di gestire un meno di gioco che permette la scelta di 3 opzioni: Nuova partita, Carica partita, Uscita. Nella prima viene implementato il metodo newGame che istanzia i costrutti della classe Winner, Mappa e Player. Di quest’ultima ne permette di scegliere il nickname tramite input dei due giocatori. La seconda permette il caricamento di una partita precedentemente salvata, istanziato tramite dei dati ricavati dal file di testo di salvataggio, i dati della griglia e dei giocatori. Entrambi le classi eseguono alla fine il metodo playGame che svolge la partita facendo giocare, un turno alla volta, entrambi i giocatori e salvando dopo ogni mossa la griglia di gioco, permettendo così di interrompere in qualsiasi momento il gioco senza perdere dati.

DIAGRAMMA UML



REFERENTI DI SVILUPPO

Lo sviluppatore del progetto è solo uno: Francesco De Angelis.