Monopoly

Davide Rossi, Alessio Minniti, Lucas Prati, Martina Ravaioli

3 aprile 2025

Indice

Capitolo 1

Analisi

1.1 Descrizione e requisiti

Il software mira alla alla costruzione di una versione virtuale del famoso gioco da tavolo "Monopoly". Quest'ultimo è un gioco di società nel quale più giocatori si muovono a turni su di un tabellone composto da caselle differenti con lo scopo di creare un monopolio. Durante la partita i giocatori si scambiano denaro a vicenda sotto forma di pagamenti e possono acquistare e migliorare proprietà. L'obiettivo è mandare gli altri giocatori in bancarotta, vince l'ultimo giocatore rimasto.

Requisiti funzionali

- Il gioco potrà essere giocato su uno stesso dispositivo da un minimo di 2 giocatori ed un massimo di 6.
- Il giocatore potrà tirare i dadi per far muovere la propria pedina sul tabellone.
- Durante il proprio turno ciascun giocatore avrà la possibilità di acquistare la proprietà sulla quale si trova se di proprietà della banca, altrimenti pagare l'affitto al proprietario. Ci saranno anche delle caselle speciali con effetti specifici.
- Nel corso della partita un giocatore, quando capita sulle sue proprietà, le potrà migliorare spendendo del denaro per posizionare delle case che elevano il costo dell'affitto.
- Durante il proprio turno, se lo desidera, il giocatore potrà vendere alla banca le proprietà o eventuali case presenti su esse.

Requisiti non funzionali

- modularità azioni
- salvare su file lo storico dei risultati delle partite precedenti
- interfaccia grafica responsive
- caricare metadata (dati delle caselle) da file locale
- rendere l'applicazione portabile su più SO

1.2 Modello del dominio

Il sistema (TurnationManager) deve gestire l'avvicendarsi dei vari turni dei giocatori. A ogni giocatore (Player) viene concesso periodicamente il proprio turno d'azione sulla base di una rotazione ciclica. Durante il turno il giocatore tira i dadi e si muove sul tabellone (Board) finendo sempre su una casella (Card). Questa casella può essere una proprietà (Property) oppure una casella speciale (Special), e sulla base di ciò cambia radicalmente l'interazione dell'utente. Se capita su una proprietà che non è ancora posseduta da alcun giocatore allora il giocatore può decidere di acquistarla. Se invece la proprietà è già stata comprata si possono verificare due casistiche: il proprietario è un altro giocatore, il proprietario è il giocatore che sta svolgendo il proprio turno. Nel primo caso il giocatore è obbligato a pagare l'affitto al proprietario, nel secondo caso può decidere se migliorare la sua proprietà aggiungendo case. Se capita su una casella speciale invece si attiverà un effetto caratteristico per ognuna di loro che avrà ripercussioni sullo svolgimento del gioco (guadagno/perdita denaro, saltare un determinato numero di turni, affrontare sfide) Una volta che il giocatore avrà svolto tutte le azioni obbligatorie potrà decidere di terminare il proprio turno. Fatto questo se il suo saldo è in negativo perde la partita e tutte le sue proprietà tornano disponibili per l'acquisto.

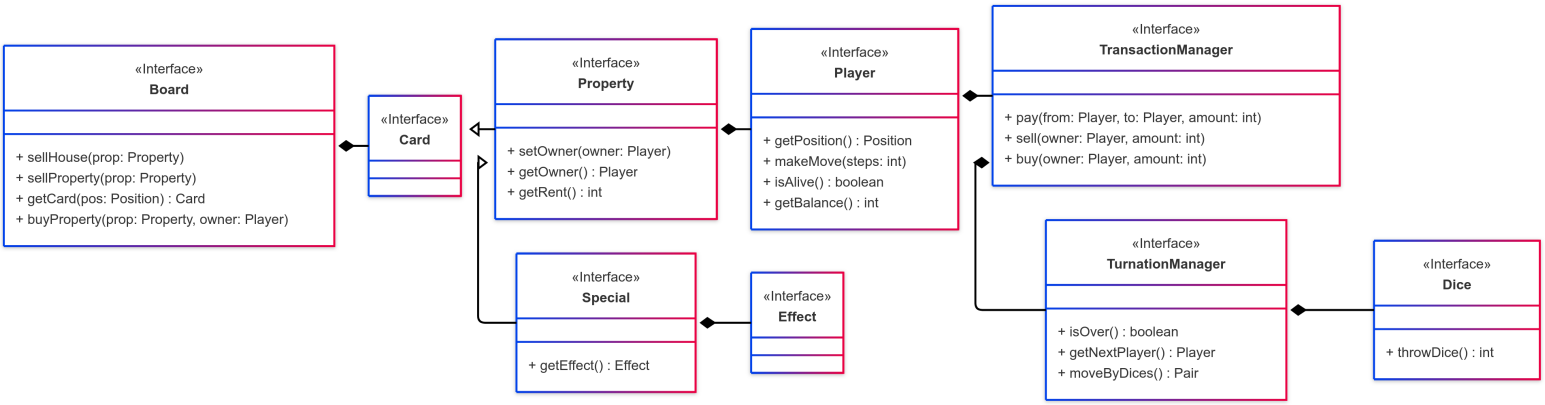

Figura 1.1: Schema UML dell'analisi del problema, con rappresentate le entità principali ed i rapporti fra loro

Capitolo 2

Design

2.1 Architettura

L'architettura di MPoly è basata sul pattern architetturale MVC. Si è scelto di implementare l'architettura nella sua forma classica in quanto il gioco ha una modalità di interazione prettamente statica: l'utente interagisce con la view, facendo scaturire un evento; a seguito di questo evento si eseguono una serie di operazioni specifiche per la sua gestione con conseguente aggiornamento della view, che mostra all'utente che cosa è successo. È stato predisposto un controller principale denominato GameManager. Questo implementa il pattern observer in quanto si mette in ascolto degli eventi che vengono catturati dalla view. Nel momento in cui viene notificato di un evento il controller interroga il model, nello specifico facendo delle chiamate a Board, che si occupa di gestire la struttura del tabellone, TurnationManager, che si occupa di gestire l'avanzamento della partita, e TransactionManager, per gestire lo scambio di denaro. Questi sono i punti d' entrata del model e offrono delle primitive che incapsulano la logica di funzionamento delle principali azioni che si possono compiere durante il gioco. Queste azioni sono caratteristiche del gioco stesso Monopoly. Con questa architettura il modello è perfettamente scorporabile e utilizzabile per costruire un software diverso con lo stesso principio di funzionamento. Una volta finita l'interrogazione del model il GameManager si occuperà di chiamare delle funzioni sulla view che aggiornano il contenuto di quest'ultima. Il controller agisce su model e view mediante delle interfacce che sono completamente indipendenti dall'implementazione di quest'ultimi. Questo fa sì che la modalità di implementazione della view non determini cambiamenti sul controller o sul model in alcun caso. Il software prevede anche un menù iniziale di configurazione della partita. Questo menù è a sua volta costituito da una sua architettura MVC più

ridotta, modellata tenendo in mente gli stessi principi descritti sovrastante. L'entità GameBuilder è colei che si occupa di creare poi l'MVC principale del software e avviare effettivamente il gioco.

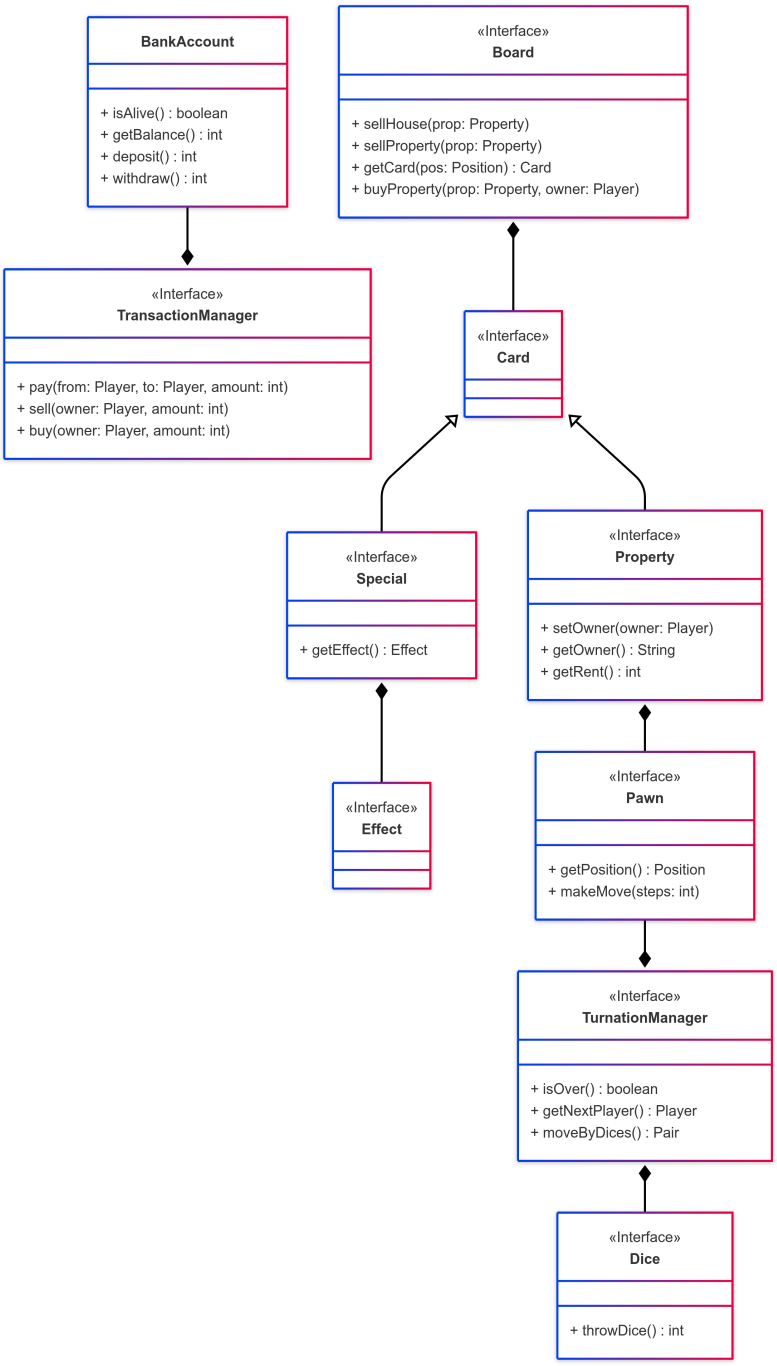

Figura 2.1: Schema UML dell'analisi del problema, con rappresentate le entità principali ed i rapporti fra loro