DESCRIZIONE INTEGRATIVA

UML MODEL – GROUP GC48

INTRODUCTION

Abbiamo scelto di creare un Fat Model che avesse già gran parte della logica di funzionamento del gioco al suo interno.

Il futuro Controller sarà Thin e si interfaccerà con il Model attraverso 3 classi principali: Game, Table e Player per gestire rispettivamente la successione dei turni e l'ordine dei giocatori, il tavolo da gioco e tutto ciò che vi viene posto sopra e il giocatore e le azioni che può eseguire.

Di seguito abbiamo spiegato in poche righe ogni classe aggiungendo anche alcuni commenti ai metodi che ritenevamo più articolati.

I colori dei paragrafi sono relativi a quelli utilizzati nell'UML.

PART RELATED TO GAME

WaitingRoom

WaitingRoom rappresenta la lobby in cui i giocatori si trovano in attesa dello start del Game (gestito dal futuro Controller). Questa classe mantiene memoria dei giocatori (e quindi delle pedine già scelte) entrati in lobby e, a momento debito, crea un game.

Game

In questa classe è gestito l'ordine delle diverse fasi di gioco: l'inizio del gioco (instanziazione di Table e dei Player), i turni dei giocatori (controllati in comunicazione con Player), la fine della partita e la decretazione di un vincitore in base agli obiettivi e ai punti accumulati. In particolare, gli attributi "first 20" e "lastRound" controllano e comunicano l'eventuale arrivo ai 20 punti (definito dalla Scoreboard) ed il successivo inizio dell'ultimo round.

Grazie all'interazione con Player e Table, il metodo endGame() valuta i vincitori o l'eventuale parità.

PART RELATED TO TABLE

Table

La classe Table gestisce tutto ciò che, nel corso della partita, si trova nella zona “comune” del tavolo di gioco: i mazzi (risorsa, oro, obiettivo) e le coppie di carte visibili a tutti i giocatori. Il Table fornisce le carte al giocatore che pesca durante il proprio turno e, successivamente, grazie all’interazione con Player, aggiorna il punteggio chiamando un apposito metodo della classe Scoreboard.

Il Table verifica anche l’eventuale fine dei mazzi, per notificare la fine del gioco alla classe Game.

Scoreboard

Scoreboard rappresenta il tabellone di gioco della specifica partita. Tramite la Map pointsMap, sono memorizzati i punti relativi a ciascun giocatore e su questa avviene l’effettivo controllo del raggiungimento della “soglia fine partita” (20 punti).

PART RELATED TO PLAYER

Player

In questa classe sono descritti i metodi legati alle funzionalità del singolo giocatore. Gli oggetti di tipo Player sono istanziati nella classe Game: esistono tanti oggetti di tipo Player quanti sono i giocatori della partita.

In particolare, in Player si trovano i metodi che verranno chiamati dal controller a seguito delle scelte dell’utente arrivate dalla view; per questo motivo, nella classe si trovano gli specifici metodi “show…” utili a mostrare al singolo utente le carte a disposizione (in mano) e quelle presenti sul tavolo di gioco.

Inoltre, in Player si trova il metodo che consente il piazzamento della carta, preceduto da un controllo specifico delle condizioni di piazzamento di quest’ultima (posizione nella mappa, coerenza in angoli/carte da coprire ed eventuale condizione “Gold”); tutti questi metodi si interfacciano con l’oggetto di tipo Station corrispondente allo specifico Player (la sua postazione di gioco).

In questa classe viene inoltre gestito il pescaggio a fine turno tramite i metodi "getCard"(getVisibleGold(…), getRandomGold(),…), legati ai corrispettivi metodi "giveCard"(giveVisibleGold(…), giveRandomGold(),…) della classe Table(dove sono presenti mazzi e carti visibili), il conteggio dei punti della carta piazzata ed il controllo finale degli obiettivi (sempre comunicando con la classe Table).

Station

Nella classe Station è gestita la postazione di gioco del singolo giocatore in cui vengono piazzate le carte giocabili. L'oggetto di tipo Station viene istanziato nella classe Player: ciascun giocatore ha una Station in cui sono mostrate le sue carte piazzate.

La posizione di una carta viene gesita dalla Map "cardPosition", che ha come chiave la singola carta e come valore le sue coordinate X Y (utili per comprendere la possibilità di piazzare carte e la loro reciproca posizione, oltre che per la valutazione degli obiettivi finali).

La lista "placedCard" è utilizzata per sapere l'ordine cronologico di piazzamento delle carte. Inoltre, per facilitare controlli e valutazioni di condizioni, in Station sono presenti le Map "myResource" e "myObject" attraverso le quali si mantiene memoria del numero di risorse e oggetti dei vari tipi Enum: la chiave è il tipo di risorsa/oggetto ed il valore è il numero di tale risorsa/oggetto visibili nella postazione.

In questa classe, oltre a utili metodi getter e setter, è presente il metodo che, chiamato all'interno del metodo PlayCard(...) di Player che definisce la decisione dell'utente, consente l'effettivo piazzamento della carta e l'aggiornamento del numero di risorse e oggetti visibili.

ENUMERATIONS USED

Color – elenca i possibili colori della pedina.

Resource – elenca le risorse esistenti.

Object – elenca gli oggetti esistenti.

CardType – elenca i possibili tipi di carta.

Position – elenca le possibili posizioni di un angolo nel fronte o retro di una carta

PART RELATED TO DECK AND CARD

Deck

Per gestire i diversi mazzi abbiamo utilizzato una Design Pattern Strategy: abbiamo un'interfaccia Deck generica e 4 classi (1 per ogni tipo di carta) che la implementano. In questo modo , con un metodo pick(), siamo in grado di pescare i diversi tipi di carte.

Card

Interfaccia che rappresenta la carta generica che viene poi implementata separatamente come PlayingCard o ObjectiveCard data la sostanziale differenza nella loro struttura e funzioni.

PART RELATED TO OBJECTIVE CARDS

ObjectiveCard

Classe che rappresenta le Carte Obiettivo con i punti e la condizione associate.

Il metodo evaluate (Station) permette di valutare quante volte la condizione si verifica nella stazione (area di gioco del giocatore) passata come parametro.

Condition

Per le diverse condizioni, abbiamo scelto di utilizzare una Design Pattern Strategy: abbiamo un'interfaccia Condition generica che è implementata da 4 classi:

- **StairCondition** rappresenta la condizione in cui le carte formano una scala.
- **LCondition** rappresenta la condizione in cui le carte sono riconducibili ad una forma di L.
- **ResourceCondition** rappresenta la condizione con un tris di una singola risorsa.
- **ObjectiveCondition** rappresenta la condizione con un trio di un singolo oggetto o un trio di 1 oggetto di ogni tipo.

PART RELATED TO PLAYING CARDS

PlayingCard

Classe utilizzata per gestire tutte le carte giocabili nella Station (tranne quindi le carte obiettivo). Qui sono salvate le informazioni che non appartengono ad una faccia specifica ma sono proprie della carta come Id, tipo di carta e per le carte risorsa e oro, il Colore (riconducibile ad una delle 4 Resource).

La carta ha un riferimento al Front e Back corrispondente e un boolean per sapere quale lato è stato piazzato nella station. Viste le differenze tra le diverse carte, la carta è poi ereditata da:

- **ResourceCard** rappresenta la carta risorsa.
- **GoldCard** rappresenta la carta oro.
- **StarterCard** rappresenta la carta iniziale (attualmente la classe è vuota ma è inserita per completezza e per future necessità legate alla grafica)

Front

Classe utilizzata per rappresentare la parte anteriore di una carta giocabile, contiene una mappa per associare la posizione nella carta (in alto a sinistra, in alto a destra, ecc.) con l'angolo corretto della classe Corner. Dato che il fronte di una carta oro e di una carta iniziale, ad esempio, sono diversi, abbiamo deciso di separare tre classi specializzate che ereditano dalla classe Front. Queste sono:

- **FrontResource** il front di una carta risorsa, in cui salviamo i punti ottenuti al suo piazzamento.
- **FrontGold** il front di una carta oro, qui salviamo informazioni come quante e quali risorse sono necessarie per il suo piazzamento e le condizioni da rispettare per ottenere i punti indicati.
- **FrontStarter** il front di una carta iniziale(attualmente la classe è vuota ma è inserita per completezza e per future necessità legate alla grafica)

Back

Classe utilizzata per rappresentare il retro di una carta giocabile, contiene una mappa per associare la posizione nella carta (in alto a sinistra, in alto a destra, ecc.) con l'angolo corretto della classe Corner e un riferimento alla classe Center associata.

Center:

Classe utilizzata per rappresentare il centro del retro di una carta in cui sono salvate le risorse permanenti ottenute al suo piazzamento.

Corner:

Classe utilizzata per rappresentare l'angolo di un fronte o retro di una carta.

Contiene informazioni quali la sua posizione nella carta e se esso è copribile o no / vuoto o no. Per gli angoli copribili e non vuoti, a seconda che l'angolo contenga un Oggetto (Quill, Inkwell, Manuscript) o una Risorsa (plant, animal, ecc.) abbiamo creato due classi che estendono la classe Corner:

- **Resource Corner** salva la Resource contenuta nell'angolo.
- **Object Corner** salva l'Object contenuto nell'angolo.