Eriantys - Prova Finale di Ingegneria del Software

Paolo Pertino [10729600] paolo.pertino@mail.polimi.it
Leonardo Pesce [10659489] leonardo.pesce@mail.polimi.it
Alberto Paddeu [10729194] alberto.paddeu@mail.polimi.it

15 marzo 2022

Indice

1 Introduzione

La **Prova Finale di Ingegneria del Software** dell'anno scolastico 2021-2022 prevede lo sviluppo di una versione software del gioco da tavolo *Eriantys*, un prodotto *Cranio Creations*[1] che si ispira e tenta di rinnovare il già affermato *Carolus Magnus*[2].

Il prodotto finale dovrà soddisfare i requisiti *Game-Specific* e *Game-Agnostic* indicati nel documento *requirements.pdf*. In particolare è richiesto l'utilizzo del design pattern *Model-View-Controller* di cui a breve forniremo una concisa descrizione.
Per incrementare il punteggio ottenuto, il team si concentrerà nell'implementazione delle regole complete del gioco, nel fornire la possibilità ai giocatori di connettersi al server e giocare tramite un'interfaccia a linea di comando (CLI) oppure mediante l'interazione con un'interfaccia grafica (GUI). Infine si darà spazio all'implementazione di quante più possibili delle seguente funzionalità aggiuntive: *12 carte personaggio, partite a 4 giocatori, partite multiple e persistenza.*

1.1 Model-View-Controller

Il *Model-View-Controller* è un design pattern per la progettazione di un'architettura software. Esso permette di separare la logica di presentazione dell'applicativo da quella applicativa(o detta di business).

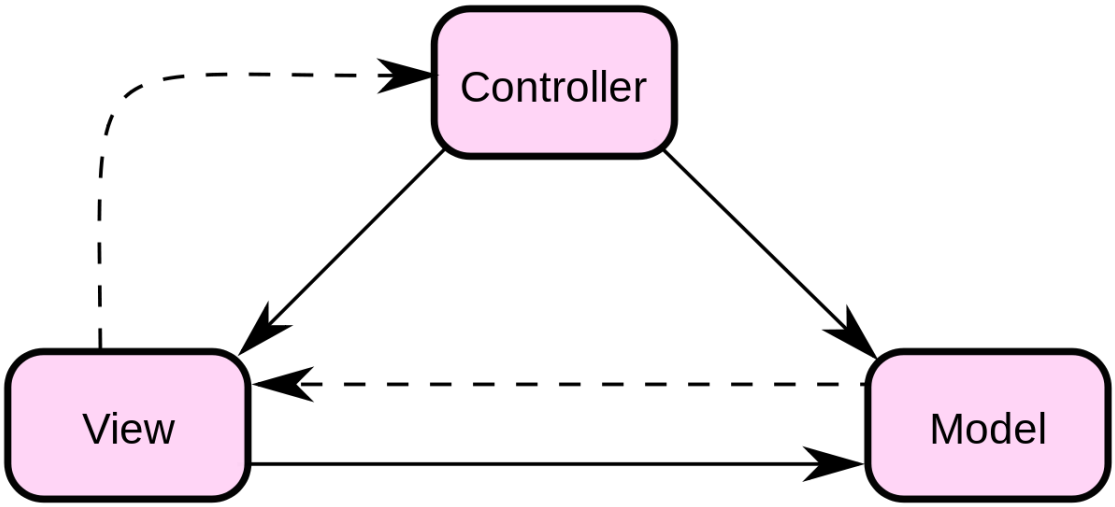

Figura 1: Model-View-Controller Pattern

Come deducibile dal nome, tale architettura è strutturata su 3 differenti layer:

1. *Model* - gestione diretta dei dati, della logica e delle regole del programma. Si noti come l'utente non modifica in modo diretto lo stato del model, bensì si interfaccia con il controller il quale gestisce in separata sede l'interazione con lo stato interno del sistema;
2. *View* - permette la visualizzazione dello stato del model e gestisce l'interazione con gli utenti e agenti esterni;
3. *Controller* - riceve i comandi dell'utente attraverso la view e li attua modificando gli stati degli altri due layers.

Sono possibili viste multiple di uno stesso modello. Nel nostro caso, infatti, saranno implementate due viste: il gioco sarà pertanto accessibile sia attraverso linea di comando sia mediante interfaccia grafica.

2 Devlog

Nella seguente sezione riportiamo settimana per settimana i progressi effettuati dal team, evidenziando, ove necessario, eventuali diagrammi e gli snodi del ragionamento.

2.1 Settimana 1: Un primo sguardo al class diagram del Modello

Durante la prima settimana di corso abbiamo analizzato i componenti fisici del gioco e le sue regole, cercando di riprodurre uno schema logico di tali elementi attraverso un *Class Diagram UML*. Esso contiene una prima bozza della struttura del Model:

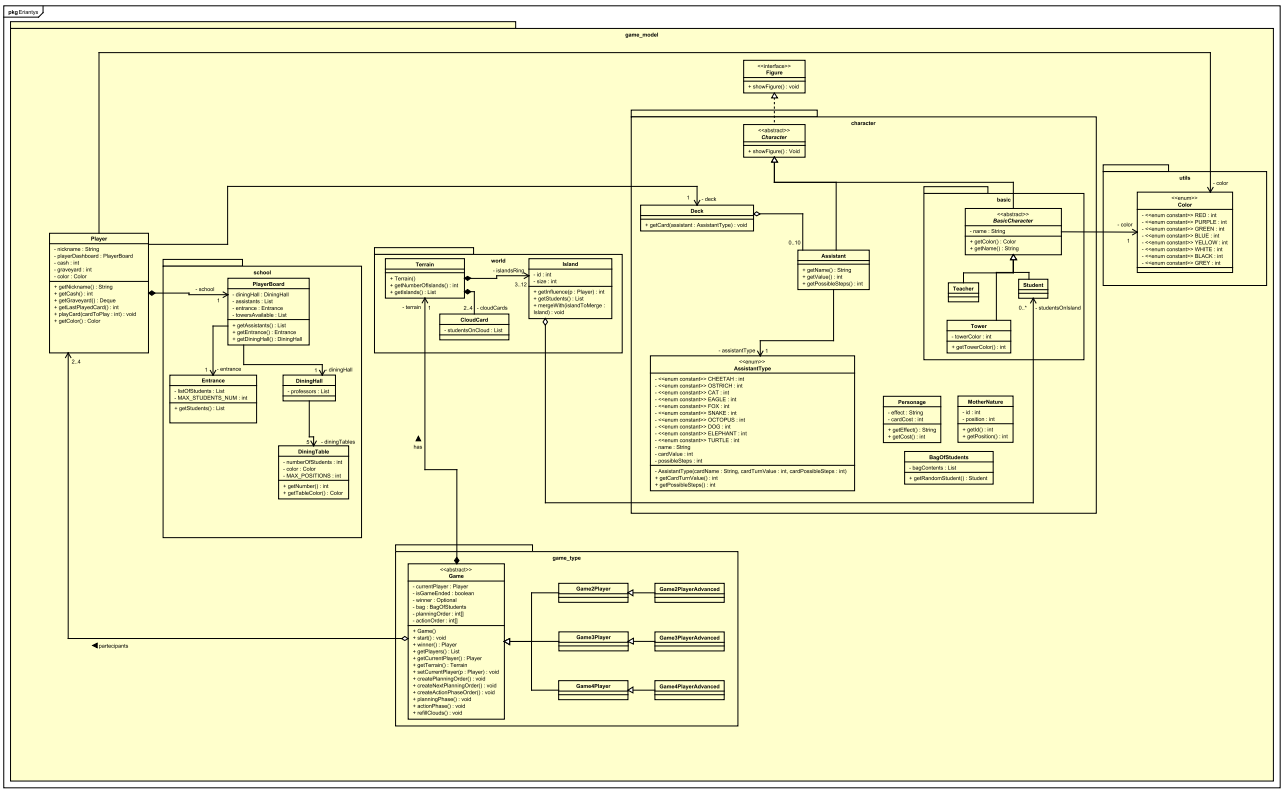

Figura 2: Class Diagram del Model - Bozza

Come indicato, lo schema sopra riportato è una bozza primitiva e di seguito riportiamo i principali ragionamenti effettuati:

- Tutti i componenti fisici, in futuro, avranno una loro grafica che dovrà essere mostrata. Pertanto implementano l'interfaccia GameObject che prevede l'implementazione di un metodo specifico per conseguire tale obiettivo.
- Il cerchio di isole che costituisce la board di gioco è stato pensato come *Doubly Circular Linked List*[3]. Con tale rappresentazione sarà più agevole lo spostamento di madre natura e l'operazione di merge di 2 isole consecutive a valle della loro conquista da parte di un giocatore.
- Volendo implementare le regole complete, quindi prevedendo la possibilità di giocare una partita seguendo la modalità per esperti, e tenendo conto della possibilità di implementare partite multiple sullo stesso server, ci siamo interrogati su come far impattare tale scelta sui diversi metodi delle varie classi. L'idea è quindi quella di utilizzare un Template

Pattern creando una classe astratta di gioco dalla quale saranno derivate le 3 versioni di gioco (per 2, 3 e 4 giocatori). Da esse discenderanno successivamente le corrispettive versioni a 2, 3 e 4 giocatori in modalità per esperti.

- In vista dell'implementazione della funzionalità di persistenza della partita, è stato brevemente analizzato il pattern *Memento* ed il suo funzionamento per capire se esso possa essere utile in futuro.

3 Strumenti utilizzati

Nella seguente sezione verranno indicati i principali strumenti di sviluppo utilizzati:

- *IntelliJ IDEA Ultimate 2021.3.2* - Principale IDE utilizzato.
- *Maven* - Gestione dello sviluppo del progetto software e di tutte le sue fasi.
- *JUnit* - Framework principale di unit testing.
- *AstahUML* - Creazione di diagrammi UML.
- *GitKraken* - Git GUI per visualizzare il workflow di sviluppo ed utilizzare efficientemente Git.
- *TEXStudio* - Gestione e aggiornamento del report.

Riferimenti bibliografici

[1] Eriantys, Cranio Creations

[2] Carolvs Magnvs, Winning Moves

[3] Circular Doubly Linked List