Relazione Progetto Programmazione ad Oggetti

Baldo Leonardo, mat.2042372

Dragon Ball Q

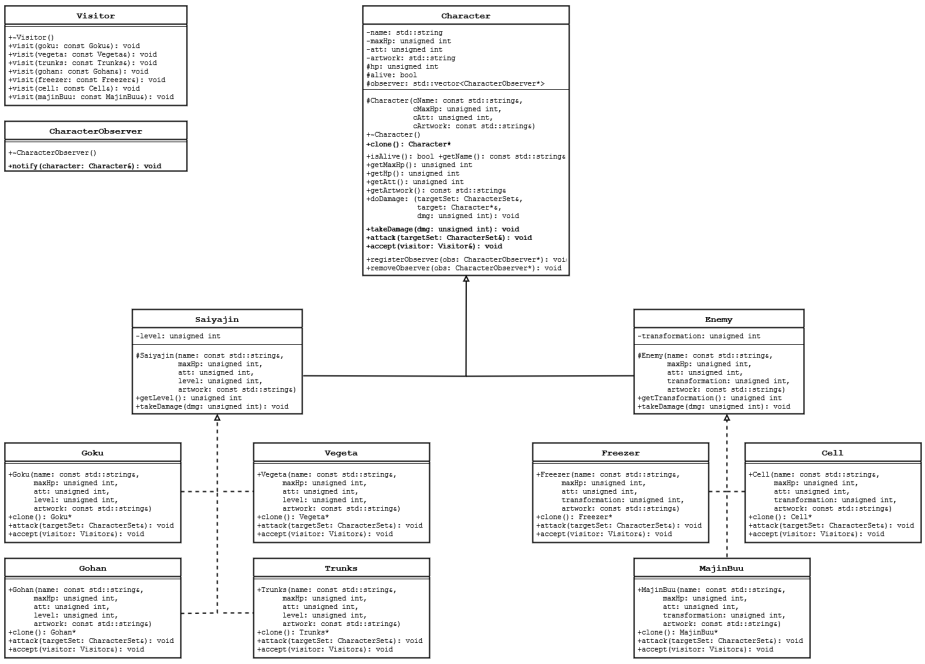
Contents

1 Introduzione

Dragon Ball Q è un videogioco molto semplice, che consente di simulare combattimenti tra personaggi della famosa opera di Akira Toriyama "Dragon Ball". Il titolo deriva dal gioco di parole tra Dragon Ball Z e il framework Qt.
Il gioco consiste nella creazione di due squadre, una comandata dal giocatore che attaccherà l'altra, ovvero quella nemica. Le squadre si possono creare da zero, ma c'è anche la possibilità di caricarle da file json o di salvarle per essere riutilizzate. I personaggi disponibili sono: Goku, Vegeta, Trunks, Gohan, Freezer, Cell e MajinBuu; ogni personaggio attacca e si difende in modo diverso dagli altri.
Il principale punto di forza è il combattimento a turni, realizzato da zero prendendo spunto da alcuni dei più famosi videogiochi in questo ambito (per esempio Pokémon), ma anche altri giochi di Dragon Ball, come la serie di Dragon Ball Budokai Tenkaichi.
Ovviamente con questo progetto non si intende avere un gioco completo, ma abbastanza minimale. Le principali funzionalità però sono presenti, come la gestione degli attacchi a turni, la gestione del cambio di personaggio nella squadra e della morte.
Il principale motivo per cui ho deciso di scegliere questo tema per il progetto è sicuramente la mia passione per la serie di Dragon Ball e per i videogiochi. La sfida più grande è stata studiare e capire l'utilizzo di Qt da zero, senza mai aver fatto qualcosa di simile prima.

2 Descrizione del Modello

Il modello logico si articola in: la classe CharacterSet che serve per gestire le squadre di Character; la gerarchia di classi dei personaggi e dalle classi per la gestione degli Observer e Visitor.
La classe CharacterSet serve per gestire le squadre, ovvero è un array di puntatori a Character. Questa classe oltre ai pirciplai operator e getter, possiede degli attributi per gestire la dimensione delle squadre e lo spostamento dei personaggi durante il gioco. Infatti i principali metodi forniti da questa classe servono per muovere i personaggi all'interno della squadra per gestire il cambio personaggio e la morte di quest'ultimi.



La gerarchia ha come padre la classe astratta Character contenente le principali informazioni come nome, vita, attacco, artwork, un indicatore bolleano per lo stato del personaggio e infine un vettore di Observer. Sono presenti inoltre i metodi getter per ognuna delle informazioni elencate in precedenza. Per la gestione degli attacchi sono presenti due classi: doDamage e takeDamage, che sono virtuali dato che hanno funzionalità diverse in base al personaggio.
La classe figlio Saiyajin, composta da Goku, Vegeta, Trunks e Gohan; possiede l'attributo level che aumenta la potenza dei loro attacchi. Nello specifico Goku attacca tutti gli avversari facendo sempre più danno; Vegeta attacca il primo avversario con danni aumentati; Trunks infligge meno danni, ma a tutti i componenti della squadra avversaria; Gohan attacca il primo avversario con un normale attacco.
La classe figlio Enemy, composta da Freezer, Cell e MajinBuu; possiede l'attributo transformation, che indica lo stato della loro trasformazione e aumenta la difesa dagli attacchi altrui. Nello specifico Freezer attacca tutti gli avversari facendo sempre più danno; Cell infligge meno danni, ma a tutti i componenti della squadra avversaria; MajinBuu attacca il primo avversario con un normale attacco.

3 Polimorfismo

Il principale uso di poliformismo non banale è presente nella gerarchia dei Character con i visitor che vengono utilizzati per tenere aggiornati gli attributi del pannello Info e degli artwork a schermo. Viene utilizzato nella persistenza dei dati in formato json. Inoltre viene usato anche nei metodi doDamage e takeDamage accennati in precedenza, fornendo un'implementazione diversa in base alla classe.

4 Persistenza dei Dati

La persistenza dei dati avviene attraverso file in formato .json come suggerito nella specifica del progetto. La directory "Json" contiene il codice, utile per la parte di persistenza, e alcuni file .json di esempio, pronti per essere caricati nel gioco. Questi file servono per caricare squadre di personaggi già pronte all'utilizzo, o per salvare squadre direttamente dall'interfaccia di gioco, così da poter essere pronte all'utilizzo quando si vuole.

5 Funzionalità implementate

Funzionalità implementate:

- conversione, caricamento e salvataggio tramite file .json
- attacco del nemico, attraverso l'apposito pulsante
- cambio del personaggio della propria squadra, attraverso l'apposito pulsante
- gestione dell'aggiornamento delle statistiche dei personaggi

Funzionalità grafiche:

- barra del menù, con relativi pulsanti per le funzionalità principali: caricamento da file, salvataggio su file, avvio di una partita, chiusura del gioco
- utilizzo di una toolbar attivabile/disattivabile (tramite menù)
- gestione di due schermate in contemporanea per le due squadre
- gestione della fine della partita, con possibilità di uscire o ritornare al menù
- ridimensionamento dell'artwork alla morte di un personaggio
- controlli e relativi messaggi d'errore

Le funzionalità elencate sono intese in aggiunta a quanto richiesto dalle specifiche del progetto.

6 Rendicontazione Ore

Attività	**Ore Previste**	**Ore Effettuate**
Studio e progettazione	10	10
Sviluppo del codice e del modello	10	11
Studio del framework Qt	10	12
Sviluppo del codice della GUI	10	14
Test e debug	5	8
Stesura della relazione	5	4
Totale	50	59

Il monte ore è stato leggermente superato in quanto lo studio del framework Qt e lo sviluppo del codice della GUI hanno richiesto più tempo del previsto, portando anche a maggiori problemi ed errori da risolvere a lato test e debug.