Progetto del corso di Programmazione ad Oggetti
2015-2016

YourQtArmy

Eugen Saraci - 1073559

Versione QtCreator: 3.3.1
Versione Qt: 5.3.0 64bit
Versione GCC: 4.9 64bit
O.S.: elementary OS Luna 64bit

1) Abstract

Il progetto si propone di rappresentare un mini gioco gestionale in cui l'utente ricopre i panni di un generale il cui scopo è quello di comprare varie tipologie di soldati dal Marketplace per poi scontrarsi con altri utenti ed accumulare denaro. Ci sono due tipologie di giocatori: Free e Premium. I giocatori Premium hanno accesso ad unità di combattimento esclusive e nel caso in cui vogliano vendere un' unità del loro esercito ottengono oltre la metà del loro prezzo d'acquisto. Nel caso di uno scontro fra un utente Premium e un utente Free, se il primo dovesse risultare perdente esso ha il 10% di probabilità di ribaltare l'esito della battaglia. Un utente Premium inoltre ha ATK (attacco) e DEF (difesa) totale aumentati. Il MarketPlace è il "luogo" in cui gli utenti possono comprare le varie unità, un utente Free può vedere le unità dedicate ai giocatori Premium, ma non può acquistarle. Non è prevista una parte di amministrazione per cui non è possibile aggiungere o rimuovere utenti o unità dal Marketplace. Il gioco si presenta in una versione molto base e non offre tantissime funzionalità tuttavia rappresenta un ottimo contesto nel quale provare e sperimentare gran parte degli argomenti visti durante il corso.

2) Parte Logica

Per evitare qualsiasi confusione fra parte grafica e parte logica il progetto è stato sviluppato non tenendo conto dell'eventuale interfaccia grafica e cercando di rispettare il più possibile i princìpi della programmazione ad oggetti. Solamente dopo la conclusione del codice "logico" si è iniziato a costruire la parte grafica.

2.1) Gerarchia di Utenti

La classe User è una classe concreta, ma con costruttore protected in modo da premettere un' istanziazione solamente alle classi che la ereditano, essa ha due campi dati privati (username e password) e i relativi metodi pubblici getter. User rappresenta un sottooggetto necessario per qualsiasi classe che abbia necessità di accedere come utente nell'applicazione. La classe Player è una classe astratta che eredita da User, essa rappresenta un utente che accede all'applicazione come giocatore, fornisce i campi dati necessari per tutti i giocatori (atk, def, etc...), i relativi getter ed i metodi virtuali puri che verranno implementati dalle altre classi. FreePlayer e PremiumPlayer rappresentano le classi "finali", essi sono utenti ed in particolare sono giocatori, per cui ereditano da Player implementando in maniera differente i vari metodi virtuali e virtuali puri della classe base.

2.2) Gerarchia di unità

La classe Unit è una classe concreta, essa contiene tutti i campi dati e metodi necessari per rappresentare un'unità di combattimento, essa contiene anche dei metodi virtuali. Si è scelto tuttavia di impostare il costruttore come protected e permettere alle classi che ereditano da Unit di implementare in maniera consona i metodi virtuali e specializzare quindi la classe base, ne è un esempio la classe Soldier, che implementa in maniera personalizzata il metodo che ritorna il valore dell'attacco (ATK) dell'unità. L'idea è quella di favorire l'estensibilità del codice permettendo l'aggiunta di classi come veicoli, carri armati ed aerei che dovrebbero implementare diversamente i metodi virtuali della classe base Unit.
Le unità hanno un campo booleano per specificare se sono premium o meno, è stato scelto di utilizzare solamente un campo dati e non una classe intera in quanto le unità premium hanno attacco base e difesa base maggiori rispetto alle unità normali impostate nei file XML, non hanno nessun metodo da implementare in maniera particolare.

2.3) Contenitori

E' stato definito un contenitore per il Marketplace (MarketList) ed un contenitore derivato dal Marketplace per contenere gli eserciti di ogni utente (Army). Entrambi sono liste singolarmente linkate, quasi identiche all'esempio visto durante il corso. La seconda aggiunge le funzionalità di calcolo dell'attacco e della difesa totali dell'esercito.
La classe Node è impostata come protected nella prima in modo che la classe Army possa ereditarla. Il contenitore rende disponibile anche un iteratore che offre le funzionalità base di confronto (== e !=) e l'incremento prefisso (++it). La classe nodo contiene due campi dati puntatore, il puntatore al prossimo nodo e il puntatore all'unità. Per il contenitore di utenti è stato utilizzato un un QVector di puntatori a Player.

2.4) BattleManager e sistema di combattimento

Il sistema di combattimento del progetto è elementare, attacco e difesa hanno lo stesso valore, un unità con ATK=200 e DEF=0 ha lo stesso valore di un unità con i valori invertiti (modulo i metodi virtuali implementati dal tipo di unità). Per decidere chi vince fra due combattenti viene fatta la somma degli attacchi e delle difese degli eserciti, l'utente che ottiene un punteggio maggiore vincerà la battaglia. Se un utente Premium perde contro un utente Free, il primo ha il 10% di probabilità di ribaltare l'esito. Una vittoria di un utente comporta un guadagno in termini di denaro per il vincitore e una perdita per il perdente, la quantità di denaro dipende dal tipo di Player. Per rendere il sistema di combattimento più interessante si potrebbe aggiungere un sistema di punti vita o di punti azione per evitare rapidi attacchi in sequenza. La classe che implementa il sistema di combattimento è chiamata BattleManager.

2.5) AppManager

AppManager rappresenta la base dell’applicazione, effettua il caricamento e salvataggio dei dati su file xml e linka a tutti i giocatori il Marketplace. La funzione funge in un certo senso da controller (design pattern MVC) in quanto offrendo pubblicamente un puntatore al QVector di Player* e un puntatore al MarketPlace fa da collegamento fra la GUI e la parte di codice vero e proprio. Il caricamento dei dati da file XML avviene solo all’avvio dell’applicazione, il salvataggio solo alla chiusura. Tutte le modifiche runtime vengono mantenute dai rispettivi contenitori (MarketList e Army), se l’applicazione dovesse crashasre inaspettatamente i cambiamenti effettuati durante la sessione verrebbero persi.

3) GUI

L’interfaccia grafica è rappresentata da due finestre, la finestra di login e la finestra principale. La finestra di login è stata creata senza l’utilizzo del Designer offerto da Qt. Per la finestra principale sono stati provati tanti modelli di layout, la scelta finale è stata guidata dalla necessità di rendere immediatamente disponibili le (poche) informazioni necessarie all’utente per giocare. Questa finestra è stata disegnata col Designer di Qt, ma il codice è stato ritoccato manualmente dalla prima all’ultima riga per rimuovere le righe inutili generate dal Designer e per aggiungere proprietà utili. Per lo sviluppo di questo progetto si è preferito unire il Controller con la View, conseguenza di questa scelta è la presenza dei vari slot nel codice della finestra che si occupano di mostrare i dati e di interagire con le parti della GUI.

4) Istruzioni per l’uso

Il programma risulta funzionante compile e run time dando I seguenti comandi dai computer di laboratorio sfruttando il file .pro generato da Qt Creator. All'interno della cartella del progetto è comunque presente il MAKEFILE generato dai primi due comandi per permettere la compilazione partendo dal passo 3 della lista sottostante.

1) qt-532.sh
2) qmake
3) make
4) ./YourQtArmy

5) Credenziali applicazione

FreePlayer1:pwd
PremiumPlayer1:password
NoArmyPlayer:pwd