PROGETTO jPokeBattle

Corso: Metodologie di programmazione

Simone Descontus

Matricola : 2116496

INDICE

1. Feature inserite

1.1 Feature comuni:

Minimo

- Implementate le prime 13 linee evolutive di prima generazione per un totale di 34 Pokemon. Ognuno di questi è dotato di almeno un moveSet di base più la possibilità di apprendere nuove mosse durante la sua crescita attraverso l'aumento di livello.

- Ogni Pokemon è dotato delle caratteristiche 6 statistiche: Ps, attacco, attacco speciale, difesa, difesa speciale, velocità, inoltre, anche le statistiche nascoste di precisione ed elusione sono state implementate. Tuttle le caratteristiche appena citate, sono in grado di essere alterate durante le battaglie attraverso l'uso di mosse specifiche, con anche la possibilità di ripristinarle a fine battaglia. In ultimo, le prime sei in lista si incrementano con la crescita del Pokemon attraverso l'aumento di livello e evoluzione.

- Attraverso l'utilizzo di javaFX, sono state implementate 10 schermate che verranno discusse in seguito: titolo, selezione dei salvataggi per uno scontro 1VS1, riorganizzazione della squadra prima di iniziare una battaglia, selezione del salvataggio per uno scontro di allenamento, nuovo salvataggio, selezione Pokemon per il nuovo salvataggio, battaglia, cambio Pokemon in battaglia, apprendimento mosse dopo l'aumento di livello, evoluzione a fine battaglia.

Tipico

- Implementate un gran numero di mosse ognuna in grado di rispettare la relazione elementale in base alla creatura con cui si sta scontrando. Ogni mossa, inoltre, possiede un base power, una precisione e il numero di utilizzi. Ci sono anche alcune mosse che non arrecano danno ma in grado di alterare le statistiche in battaglia dei Pokemon in campo finché questi non vengono sostituiti

Extra

- Set crescita:
  - Ogni Pokemon è dotato di punti esperienza che aumentano ogni volta che quest'ultimo sconfigge un avversario. Arrivato ad un valore prestabilito, questo salirà di livello, migliorando le sue statistiche ed eventualmente, apprendendo mosse o evolvendosi.
  - Implementate le meccaniche di apprendimento mosse: qualora ci dovessero essere slot vuoti al momento dell'apprendimento, questo verrebbe riempito in automatico con la nuova mossa. In caso contrario, una schermata con la richiesta di sostituzione della mossa verrebbe mostrata con una finestra, lasciando all'utente la scelta se insegnare al proprio Pokemon la nuova abilità o se ignorarla.
  - Implementato il sistema di evoluzione dei Pokemon, il quale si innesca al raggiungimento di un livello specifico delle varie creature. A fine di uno scontro, qualora un'evoluzione fosse possibile, questa verrebbe mostrata a schermo attraverso un'apposita finestra, qui l'utente può accettare o rifiutare l'evoluzione, che gli verrebbe comunque riproposta al prossimo aumento di livello del Pokemon.

1.2 Feature due sviluppatori:

Minimo
- Implementato un sistema di lotte alla meglio delle 3, con ripristino dello stato del Pokemon tra una lotta e l'altra. In questo contesto, i Pokemon possono salire di livello sconfiggendo gli avversari, imparare mosse ed evolversi.
- L'intero sviluppo è stato svolto sulla piattaforma Eclipse con l'integrazione di un repository github: https://github.com/PatatineFritte0/PokeJBattle.git

Tipico
- Ogni utente può crearsi un profilo, il quale terrà conto delle vittorie e delle sconfitte
- Ogni profilo utente viene salvato su file, così che lo stato della sua partita possa essere ripristinato in qualunque momento.

Extra
- Implementato un sistema di allenamento contro un NPC, che consente di rafforzare i propri Pokemon in viste di lotte future contro altri umani. Questi scontri non influenzano i counter di vittorie e sconfitte.

2. Pattern adottati

2.1 Model-view-controller:

Come soluzione per la gestione delle interfacce utente abbiamo scelto di utilizzare un pattern MVC, con relativa suddivisione anche visibile attraverso i package del progetto. Il model contiene le classi e costanti di base, che verranno discussi in seguito, di cui i controller hanno bisogno per gestire le varie feature implementate. Negli 11 controller presenti vengono gestite le logiche per delle varie interfacce grafiche, contenute nel view. Gli aspetti grafici di questi ultimi sono stati reperiti nella loro totalità in rete, prendendo ispirazione principalmente dalle interfacce dei giochi Pokemon diamante/perla/platino per rimanere coerenti con lo stile 2d del progetto. Tra questi troviamo ad esempio front e back sprite dei Pokemon nonché sfondi e design dei pulsanti.

2.2 Simple factory:

Per quanto non sia in tutto e per tutto un pattern, ritengo doveroso citare l'utilizzo di questa tecnica. Per la gestione della generazione dei vari Pokemon, infatti abbiamo creato una classe apposita che istanziasse su richiesta le varie creature specificandone solo il nome o l'indice sotto forma di stringa.

3. Risorse esterne

3.1 JavaFX:

Per l'implementazione dell'interfaccia utente, la nostra scelta è ricaduta su JavaFX con l'ausilio del software SceneBuilder per una creazione più maneggevole delle schermate e dei relativi file fxml.

3.2 Gson:

Riflettendo su come gestire i salvataggi abbiamo optato per la scrittura su un file in formato Json, così che fosse più facilmente maneggiabile per quanto riguarda la scrittura e lettura dei dati. Per implementare in maniera agevole una conversione degli oggetti di tipo Allenatore in formato Json e viceversa, la libreria Gson di google ci è risultata particolarmente utile in quanto versatile e in grado di convertire gli oggetti passando come parametri soltanto questi ultimi e le loro classi.

4. Discussione costanti e classi

4.1 Costanti:

Tipo:

Contiene tutti i tipi che i Pokemon e mosse possono possedere. Ogni tipo ha tre liste come parametri associate rispettivamente ai tipi contro cui quello corrente è forte, debole o inefficace. Affinché queste liste contenessero correttamente i tipi, una porzione di codice statica è stata inserita per evitare elementi null che si sarebbero presentati semplicemente per l'ordine di dichiarazione dei tipi stessi. Infine, l'enumerazione Tipo contiene un metodo in grado di calcolare la relazione tra il tipo chiamante e un altro passato come parametro, ad esempio Tipo.ACQUA.calcolaRelazioneTipi(Tipo.FUOCO) restituirebbe 0.5 in quanto acqua è poco efficace su fuoco.

Categoria:

Contiene i tre formati di mossa disponibili: fisica, speciale o stato.

Statistica:

Ogni Pokemon possiede cinque statistiche più due disponibili solo in battaglia, queste sono contenute qui. Le statistiche sono le seguenti: attacco, attacco speciale, difesa, difesa speciale, velocità, precisione ed elusione.

Mossa:

Una mossa è una delle azioni che possono essere eseguite da un Pokemon durante la battaglia. Queste sono caratterizzate da diversi aspetti: abbiamo ovviamente un nome nonché il base power, corrispondente alla potenza intrinseca della mossa. Alti parametri sono il numero di utilizzi (PP) o la precisione, che corrisponde a uno degli elementi che concorrono al calcolo della probabilità che un attacco non vada a segno. Abbiamo poi il campo categoria che, se fisico o speciale interrogherà le corrispettive statistiche dei Pokemon coinvolti nello scontro per ottenere un corretto calcolo dei danni, d'altra parte se la categoria fosse di stato staremmo parlando di un tipo di mossa particolare, che non fa danni e altera una statistica di sé stessi o dell'avversario durante la battaglia. Nel caso si tratti di una mossa di stato, perciò, il base power sarà uguale a zero, e altri tre parametri saranno portati all'attenzione per l'esecuzione della mossa: statBoostNerf specifica su quale statistica va applicata l'alterazione, lvlBoostNerf specifica di quanto va alterata la appena definita statistica, e in ultimo, un parametro onSelf booleano chiarirà se il bersaglio è sé stessi.

4.2 Classi:

BaseStat:

Questa è una classe annidata internamente a Pokemon, e consente di prendere la normale enumerazione di statistiche e trasformale in oggetti alterabili durante le battaglie. Per raggiungere questo scopo BaseStat è caratterizzata dalla Statistica associata e da un valore che varia da Pokemon in Pokemon. Questo, viene poi assegnato a un parametro mainValue che non cambierà e resterà come riferimento principale, il battleValue d'altra parte sarà l'elemento modificabile in battaglia e al quale il Pokemon fa riferimento durante gli scontri. In ultimo, un parametro modifyLvl identifica di quanto una statistica è stata alterata durante la battaglia, con un minimo di -6 e un massimo di +6. Attraverso il metodo modify(int val) la statistica può essere alterata di N stadi corrispondenti al parametro N, sia positivamente che negativamente.

UsableMove:

Analogamente a BaseStat, anche questa classe ha lo scopo di trasformare una costante in un oggetto effettivamente modificabile durante le battaglie. Una UsableMove è caratterizzata in primo luogo dalla sua costante Mossa associata, poi i suoi utilizzi vengono assegnati a due parametri: ppMax, il quale valore non cambierà mai, e pp che memorizza gli utilizzi correnti e che verranno decrementati a ogni utilizzo dopo la chiamata del metodo useMove().

Pokemon:

Uno dei protagonisti del progetto, classe astratta contenente una grande quantità di parametri nonché di metodi. Partendo dal nome, un Pokemon è caratterizzato da due tipi, di cui il secondo può essere null, un array UsableMove[4], ossia le mosse utilizzabili, e una HashMap<Integer, Mossa> corrispondete al parco mosse al quale la creatura ha accesso, in questo, la chiave corrisponde al livello al quale si può apprendere la mossa specifica, contenuta nel valore corrispondente. Andando avanti ogni Pokemon ha dell'esperienza accumulabile, un livello e un livelloEvo raggiunto il quale il parametro booleano tryToEvolve verrà impostato a true rendendo possibile l'evoluzione, contenuta in un parametro evo di tipo Pokemon. Per quanto riguarda la battaglia i campi coinvolti sono maxPs e ps, che memorizzano rispettivamente i punti salute massimi e correnti, e le sette statistiche totali (attacco, attacco speciale, difesa, difesa speciale, velocità, precisione ed elusione) tutte di tipo BaseStat.

Riguardo ai metodi di Pokemon abbiamo attacca() che identifica la categoria della mossa utilizzata, hit() che effettua i calcoli dei danni con anche eventuali colpi non andati a segno, e incassa() che fa subire dei danni alla creatura. Per la crescita del Pokemon, abbiamo creato un'interfaccia omonima con i metodi levelUp(), evolve() e gainExp(). Il primo dei tre appena citati aumenta le statistiche con l'aiuto dei metodi privati calcPs() e calcStat() che si occupano di calcoli matematici e se lvl dovesse essere uguale a evoLvl chiamerebbe il metodo evolve(). Anche il secondo fa uso dei metodi privati per aumentare le statistiche, ma lo fa in percentuale maggiore e assegnando al Pokemon le principali caratteristiche della propria evoluzione. Infine, l'ultimo metodo del trio contenuto nell'interfaccia crescita si occupa di far accumulare l'esperienza in base al nemico sconfitto con delle apposite forumle e, se necessario, chiamare il metodo levelUp().

Pokemon specifici:

Ogni creatura specifica ha corrispondente una classe che eredita da Pokemon i cui parametri sono già inseriti nei singoli costruttori. Inoltre, come dettaglio extra abbiamo inserito MissingNo come Pokemon evoluzione comune al livello 101 (irraggiungibile poiché il massimo è 100) di tutte le creature che normalmente non evolvono.

Allenatore:

Un allenatore corrisponde all'utente. È caratterizzato da un nickname, una squadra di Pokemon[6], un mainPokemon che corrisponde alla creatura attualmente in campo, il numero di Pokemon in squadra, il numero di vittorie, sconfitte, il winRatio e l'ultimo accesso in formato stringa. Di questa classe, i metodi degni di nota sono setMainPokemon() che imposta la creatura in campo da una di quelle in squadra, setSquadra() utile per riorganizzare il team, getPokemonId() per ottenere in quale posizione è un determinato Pokemon nella squadra e, specularmente, getPokemonById() per ottenere un Pokemon dalla sua posizione nella squadra.

Franco:

Questo simpatico ragazzo è una classe figlia di allenatore e svolge il ruolo di NPC contro il quale è possibile allenare i propri Pokemon. È dotato di un campo strategia di tipo Random che lo aiuta a prendere le decisioni, un campo enemy di tipo Allenatore per calibrare il livello della propria squadra su quella dell'avversario con tanto di mainEnemy che contiene il Pokemon avversario attualmente in campo. Poi, i parametri scambioConsentito e statusCounter permettono a Franco di non cadere in brutti loop inconcludenti.
Con il metodo agisci() Franco decide se cambiare Pokemon o attaccare, in questo contesto, scambioConsentito sarà impostato a true solo dopo un cambio da parte dell'utente e sarà impostato nuovamente a false dopo un cambio da parte dell'NPC attraverso il metodo cambia(). Per quanto riguarda il caso del cambio, Franco controllerà sempre se ha in squadra creature superefficaci contro l'avversario, ma il cambio strategico avverrà solo col 30% di probabilità col metodo controllaSquadra(). Nel caso in cui, invece, la scelta dovesse ricadere su un attacco, Franco tenterà sempre di usare mosse superefficaci sul suo avversario o mosse di stato, statusCounter gli impedisce di usarne più di 6 dallo stesso Pokemon.

FactoryPkmn:

Classe contenente solo metodi statici con l'onere di generare istanze di tutti i Pokemon attraverso il metodo crea(String nome) che con il parametro nome identifica il Pokemon da creare utilizzando il polimorfismo delle classi figlie di Pokemon. Questa factory è anche in possesso di un metodo random() per generare Pokemon casuali per Franco e di un metodo allPokemon() che restituisce una lista di tutte le creature implementate.

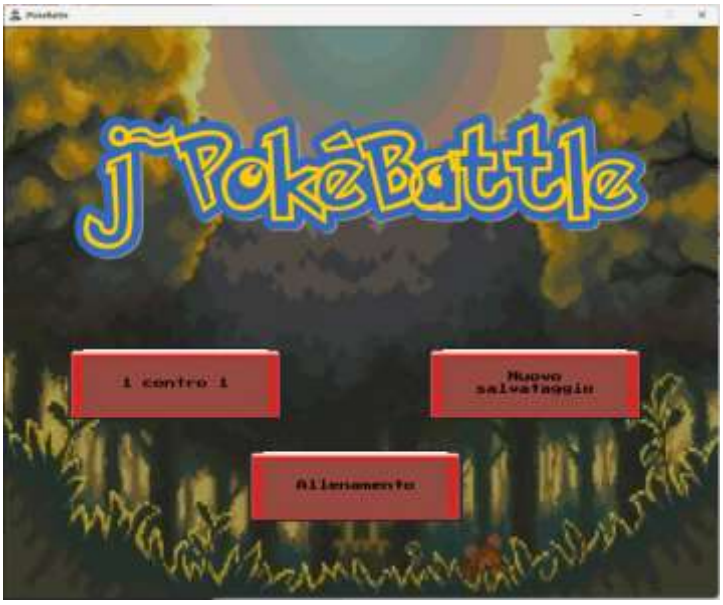
SaveManager:

Come la classe factory di cui sopra, anche il SaveManager ha solo metodi statici i quali si interfacciano con la libreria Gson per convertire gli oggetti Allenatori in formato Json.
Il metodo newSave() legge dal file l'array di allenatori già presenti e ne aggiunge uno nuovo in coda. loadSave(String nickname) legge tutti gli Allenatori nell'array e, se trova il salvataggio corrispondente al nickname, ne genera un'istanza.
Il metodo save(Allenatore t) aggiorna l'array inserendo una nuova versione dell'allenatore passato come parametro e lo scrive su file. Infine, getSave() fornisce una lista di tutti gli allenatori presenti sul file dei salvataggi.

5. Discussione interfacce

5.1 Title screen:

La prima schermata è molto semplice, lascia all'utente la possibilità di scegliere tra creare un nuovo salvataggio, allenare i propri Pokemon contro un NPC o iniziare uno scontro con un altro umano.



5.2 1VS1:

All'interno della schermata 1VS1 gli utenti possono selezionare i loro salvataggi per poi iniziare uno scontro

5.2.1 Riorganizza squadra:

Una volta selezionato un salvataggio, cliccando l'ingranaggio è possibile riorganizzare la propria squadra.

5.3 Allenamento:

La schermata di allenamento è analoga alla schermata per gli scontri a due giocatori, ma qui è selezionabile solo un salvataggio, poiché l'avversario sarà sempre Franco.

5.4 Nuovo salvataggio

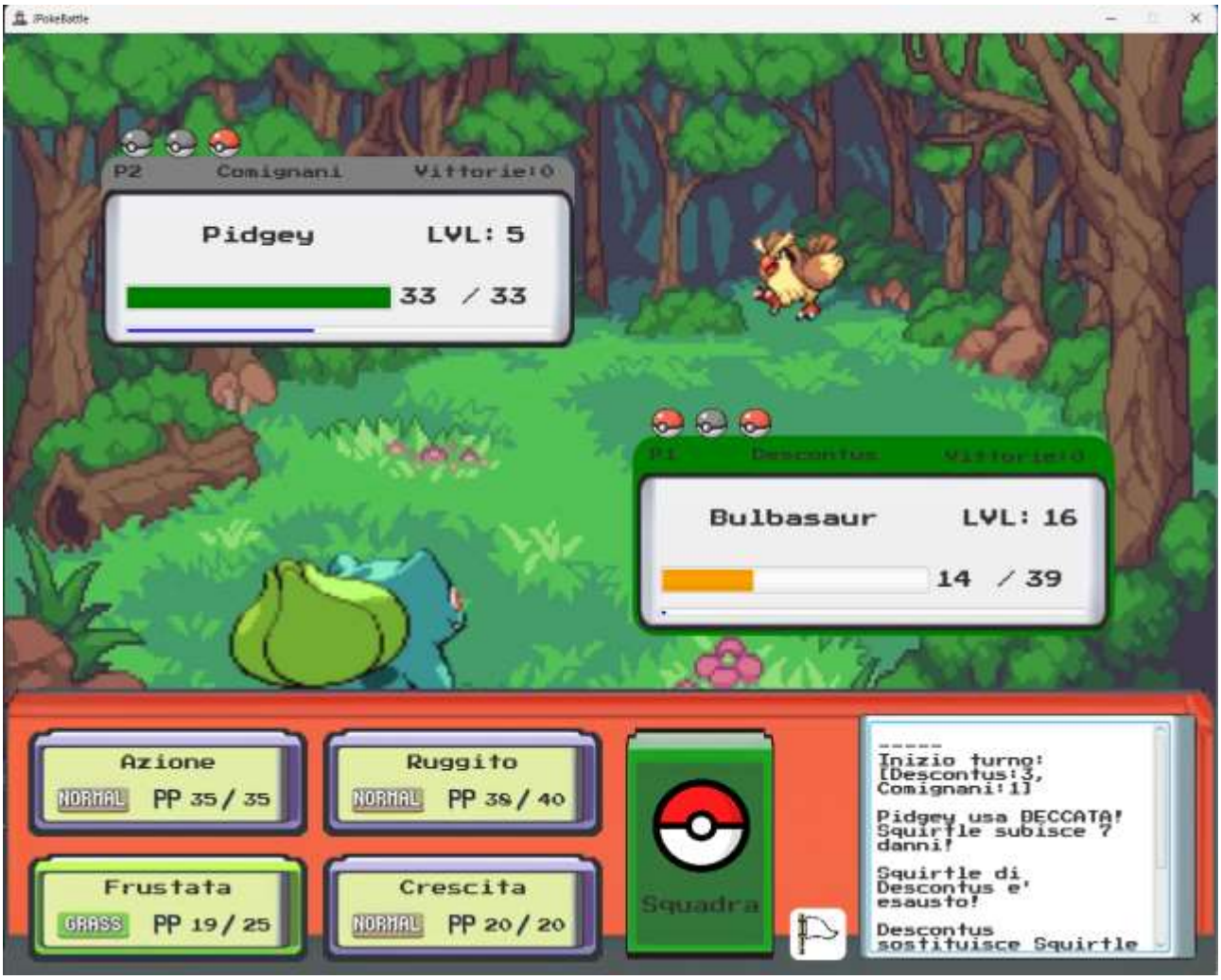
Dalla schermata del nuovo salvataggio è possibile creare il proprio allenatore con un nickname e fino a 6 Pokemon con il minimo di uno per squadra.

5.4.1 Aggiungi Pokemon

Selezionando uno spazio "+" è possibile ottenere un Pokemon dalla lista di quelli implementati.

5.5 Battaglia

In questa schermata gli utenti possono scontrarsi tra di loro o contro Franco per un allenamento. I riquadri degli utenti contengono il loro Nickname e il numero di vittorie della best of three, il riquadro verde segnala di chi è il turno al momento. Il numero di Pokeball sopra al riquadro equivale al numero di Pokemon in possesso dall'allenatore, queste diventeranno grigie per ogni Pokemon KO. Tra i dati dei Pokemon nei riquadri possiamo notare il loro nome, il loro livello, la loro barra della vita, con anche valore numerico associato, e la piccola barra dell'esperienza.



Tra le scelte attuabili dal giocatore abbiamo la selezione di una tra le quattro mosse possedute dal Pokemon, la possibilità di effettuare un cambio o la resa, che farà perdere la partita (ma non la best of three) a chi la seleziona. È inoltre presente anche un log degli eventi a destra.

Durante uno scontro, un cambio avviene sempre come prima azione, per quanto riguarda le altre mosse, viene considerata la velocità corrente dei due Pokemon per capire l'ordine con il quale queste verranno sferrate, in secondo luogo vengono calcolate le relazioni tra i tipi e i fattori casuali per poi giungere all'esecuzione degli eventi.

5.5.1 Cambio Pokemon

La seguente schermata viene mostrata quando un giocatore vuole cambiare Pokemon in campo

5.5.2 Apprendimento mosse

Qualora un Pokemon salisse di livello conoscendo già quattro mosse, la seguente schermata verrà mostrata. Cliccare una delle quattro mosse già imparate la farà sostituire con quella nuova, mentre cliccare la quinta porterà ad un annullamento dell'apprendimento.

5.5.3 Evoluzione

A fine partita, se un Pokemon ha raggiunto il livello necessario ad evolversi, la seguente schermata viene mostrata.