

Programmazione ad oggetti

Progetto anno: 2018/2019

Rizzo Ilaria

Matricola: 1126073



Indice

1	Introduzione	3
1.1	Descrizione del contesto generale	3
1.2	Gerarchia	3
1.2.1	Classe base: Personaggio	3
1.2.2	Attacco	4
1.2.3	Difesa	4
1.2.4	Eroe	4
1.3	Polimorfismo	4
1.4	Contenitore	5
1.5	Smart Pointer	5
1.6	Gui	5
1.7	Manuale utente	6
1.7.1	Visualizzare i dati da file	6
1.7.2	Scrivere su file	6
1.7.3	Modifica e rimozione	6
1.7.4	Ricerca	7
1.8	Formato file	7
1.9	Tempo di lavoro	8
1.10	Ambiente di sviluppo e test	8
1.11	Comandi per la compilazione ed esecuzione	8



Elenco delle figure

1	Gerarchia	3
2	GUI	6
3	Struttura file JSON	7

1 Introduzione

1.1 Descrizione del contesto generale

Clash of Clans è un gioco di strategia online disponibile per dispositivi mobili. Si tratta di un videogame dove lo scopo principale è creare un proprio villaggio e accumulare risorse per migliorarlo rubandole ai villaggi degli avversari. Gli attacchi vengono fatti creando degli eserciti scegliendo tra i vari personaggi a disposizione per ogni livello e il villaggio viene costruito con le difese disponibili per il proprio livello. Prendendo ispirazione da questo gioco ho deciso di creare questo progetto per descrivere le caratteristiche dei vari personaggi del gioco.

1.2 Gerarchia

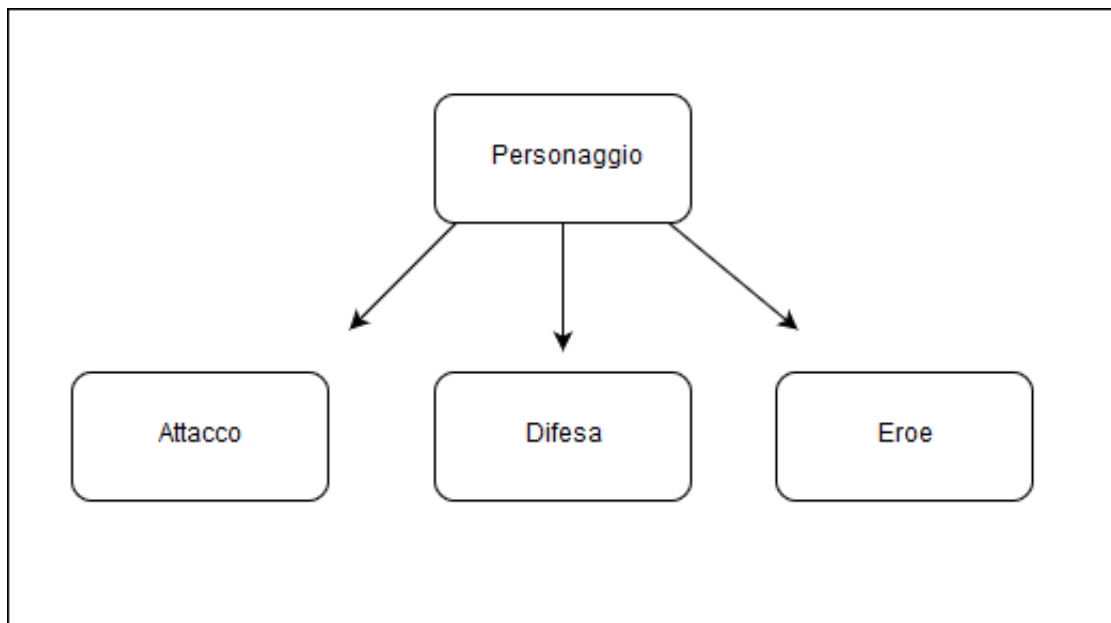


Figura 1: Gerarchia

La gerarchia si compone nel seguente modo: una classe base di nome "Personaggio" e tre classi derivate (tutte derivate dalla classe base) di nome "Attacco", "Difesa", "Eroe".

1.2.1 Classe base: Personaggio

La classe base "Personaggio" è composta dai campi comuni a tutte le classi derivate:

- Nome;
- Bersaglio preferito;

- Costo;
- Vita;
- Tempo di costruzione/preparazione;
- Tipo di danno;
- Bersagli;
- Danno;

1.2.2 Attacco

La classe derivata "Attacco" si riferisce ai personaggi attaccanti che presentano i seguenti campi in più rispetto alla classe base:

- Spazio occupato;
- Velocità di movimento;
- Tipo (incantesimo o truppa);

1.2.3 Difesa

La classe derivata "Difesa" si riferisce ai personaggi difensori che presentano i seguenti campi in più rispetto alla classe base:

- Distanza;

1.2.4 Eroe

La classe derivata "Eroe" si riferisce ai personaggi eroi che presentano i seguenti campi in più rispetto alla classe base:

- Livello massimo;
- Tempo di rigenerazione;

1.3 Polimorfismo

Nel programma il polimorfismo è stato utilizzato per i metodi dichiarati come virtuali e definiti nella classe base e ridefiniti nelle classi derivate:

- `virtual bool operator==(const Personaggio) const;`
- `virtual bool operator!=(const Personaggio) const;`
- `virtual Personaggio() = default;`

Infine son presenti altri due metodi polimorfi dichiarati nella classe base come virtuali puri e ridefiniti quindi solo nelle classi derivate

- `virtual Personaggio* clone() const = 0;`
- `virtual unsigned int incremento() const=0;`

1.4 Contenitore

Viene dichiarata e implementata una classe `Container` che rappresenta il contenitore di tipo array del quale vengono implementati i seguenti metodi:

- overloading degli operatori `[]`, `<`, `!=`, `=`, `==`
- costruttore, distruttore;
- vuoto: controlla se il contenitore è vuoto;
- agg: aggiunge un elemento al contenitore;
- elimina: rimuove un elemento dal contenitore;
- begin e end: ritornano un iteratore che punta all'inizio e uno che punta alla fine del contenitore;
- getCapacity e getSize: ritornano i due campi della classe;
- clear: svuota il contenitore;

1.5 Smart Pointer

La classe `SmartP` rappresenta gli smart pointer templetizzati utilizzati, di cui vengono ridefiniti gli operatori `=`, `*`, `->`, `==`, `!=`; implementati il costruttore, costruttore di copia, e il distruttore.

1.6 Gui

È stato scelto di utilizzare il pattern Model-View quindi nel file `window.cpp` è contenuto tutto il codice che serve per mostrare l'interfaccia grafica. Nella parte di sinistra son presenti i campi di compilazione utilizzati per inserire un nuovo personaggio, visualizzarne i dettagli, e modificarli. Nella parte centrale son presenti i filtri di ricerca e il relativo pulsante "cerca", infine nella parte di destra è presente la sezione dove viene visualizzata la lista degli oggetti (ordinati per nome) grazie al click sul pulsante "leggi da file", e sotto i pulsanti per aggiungere, salvare su file, rimuovere, modificare un personaggio.




Figura 2: GUI

1.7 Manuale utente

1.7.1 Visualizzare i dati da file

Per leggere i dati contenuti nel file è necessario cliccare su "Leggi da file" e selezionare il file. a questo punto comparirà una lista di personaggio ordinati per nome (univoco).

1.7.2 Scrivere su file

Per inserire un nuovo personaggio è necessario cliccare "aggiungi", vengono puliti i campi della sezione di sinistra per permetterne la compilazione con nuovi dati, e compare il pulsante "salva su file". È necessario precisare che per permettere l'inserimento si è deciso di rendere obbligatorio il tipo e il nome del personaggio (che dovrà essere univoco) in caso non vengano rispettate queste condizioni vengono visualizzati dei messaggi di errore che spiegano qual'è il problema. Infine per permette l'inserimento su file bisogna concludere cliccando il pulsante apposito e a questo punto viene richiesto di scegliere il file su cui salvare,

1.7.3 Modifica e rimozione

Per rimuovere un personaggio ovviamente prima è necessario leggere gli elementi contenuti in un file, quindi dopo la procedura descritta nella sezione "leggi da file" si può procedere cliccando semplicemente su un nome della lista. A questo punto compariranno i suoi dettagli nel lato di sinistra e in basso i due pulsanti rimuovi e modifica. La rimozione è molto semplice basta cliccare il pulsante apposito e viene

eliminato il personaggio selezionato. Per la modifica ci sono delle precisazioni da fare. Per prima cosa il nome non è modificabile quindi in caso si voglia cambiare si obbliga l'utente ad eliminare il personaggio selezionato e crearne uno nuovo per praticità. "Incremento" non è un dato che inserisce l'utente in quanto consiste in un potenziamento del danno in base al personaggio (es. l'incremento di un attaccante è il danno+2) per questo motivo la sua SpinBox è non modificabile, ci penserà il programma a calcolarlo da solo. Una volta modificati i campi si clicca su modifica per dare conferma. A questo punto i campi vengono ripuliti e se si riclicca sul nome del personaggio modificato si potranno vedere i nuovi dettagli.

1.7.4 Ricerca

La sezione di ricerca è nella parte centrale della schermata, la ricerca può avvenire per nome, tipo, bersagli, bersaglio preferito, tipo di danno. Sono settati come "qualsiasi" quindi se non si esprime una preferenza la ricerca restituirà tutti gli elementi che corrispondono alle possibilità di quel filtro (es. se non si esprime preferenza sul tipo di attacco verranno mostrati personaggi con un tipo di danno ad area e a bersaglio singolo). Per la ricerca per nome è stato deciso di aggiungere una nota in più che consiste in un messaggio "lista vuota" in caso non fosse presente. Ovviamente prima di procedere con la ricerca è necessario leggere da file i dati.

1.8 Formato file

La scelta è ricaduta sui file Json in quanto considerati più semplici nella comprensione, gli oggetti all'interno del file vengono visualizzati nel seguente modo:

```
{
  "Personaggi": [
    {
      "bersagli": "Terra",
      "bf": "Indifferente",
      "costo": 20,
      "danno": 26,
      "nome": "Barbaro",
      "spazio": 1,
      "tempo": 5,
      "tipo": "truppa",
      "tipodanno": "Bersaglio singolo",
      "type": "Attacco",
      "velocita": 16,
      "vita": 90
    },
    {
      "bersagli": "Terra",
      "bf": "Indifferente",
      "costo": 50,
      "danno": 25,
      "nome": "Arciere",
      "spazio": 1,
      "tempo": 6,
      "tipo": "truppa",
      "tipodanno": "Bersaglio singolo",
      "type": "Attacco",
      "velocita": 24,
      "vita": 48
    }
  ]
}
```

Figura 3: Struttura file JSON

1.9 Tempo di lavoro

- analisi del problema(3h);
- progettazione GUI(2h);
- apprendimento libreria Qt(7h);
- codifica modello e GUI(25);
- debugging(8h);
- testing(8h)

1.10 Ambiente di sviluppo e test

Sistema operativo ubuntu 16.04, compilatore gcc 5.4.0, libreria Qt 5.5.1;

IMPORTANTE: il progetto utilizza un file .pro personalizzato, non sovrascrivere il file .pro

1.11 Comandi per la compilazione ed esecuzione

- qmake
- make
- ./progp2