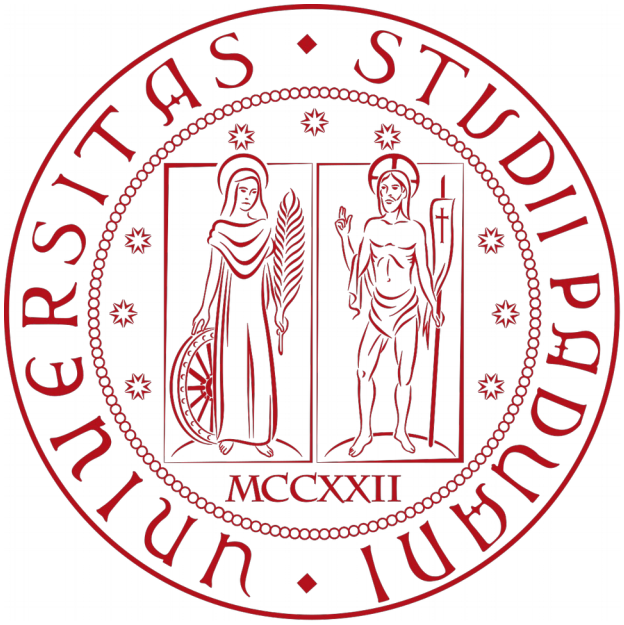


Università degli Studi di Padova

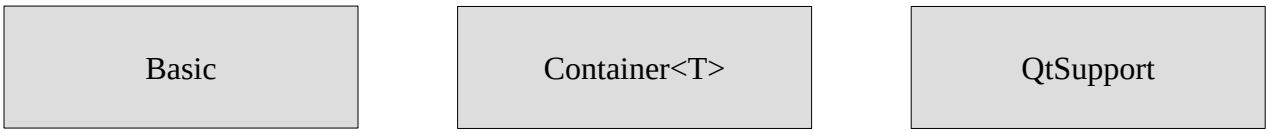
Progetto di Programazione ad Oggetti “Qcontainer”
a.a. 2018/2019

SkyrimQt

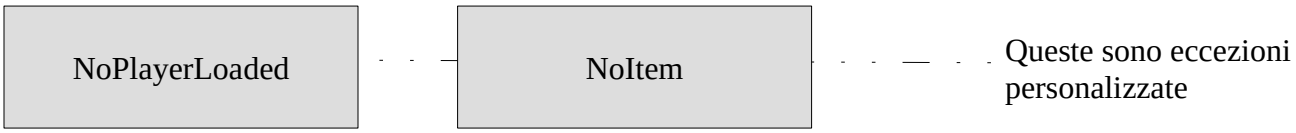
Sorge Francesco 1170715

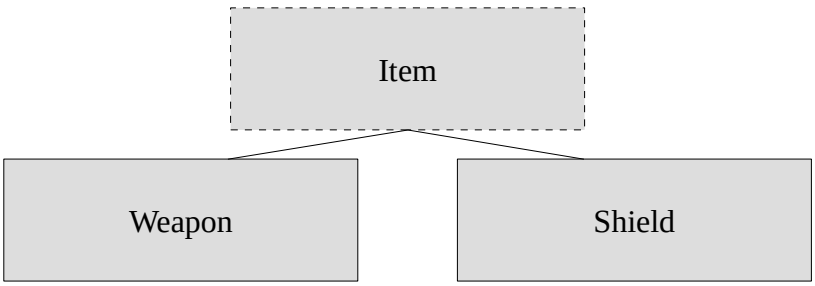
TIPI DEFINITI DALL'UTENTE

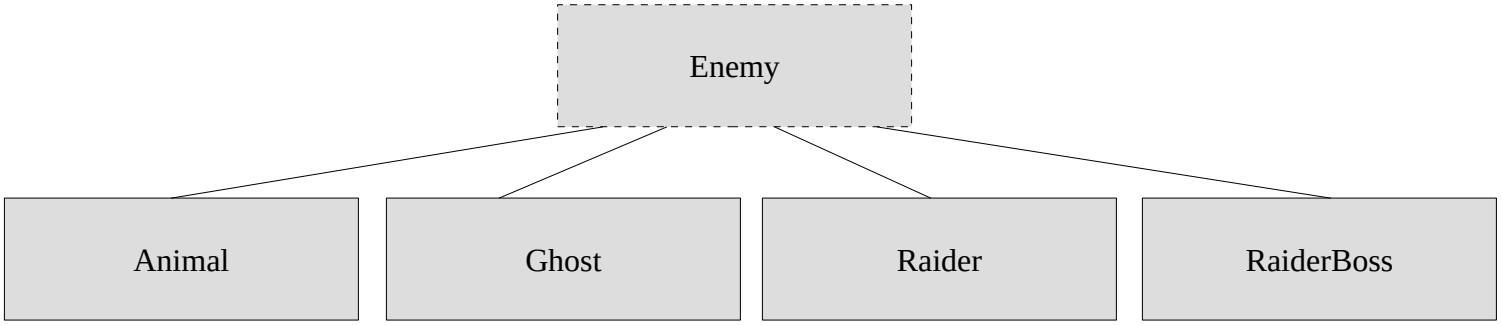
Namespace: FrancescoSorge



Namespace: Skyrim







Classi delle finestre e widget di Qt

CHIAMATE POLIMORFE

```
MatchWindow::on_dyamicButton_clicked()
223    game->getEnemy()->attack()
```

ushort attack() è un metodo virtuale puro, ovvero non è definito nella classe base, pertanto ogni classe che intende derivare da essa, deve definire questo metodo. In virtù di ciò, il compilatore non conosce, a compile time, la funzione che verrà chiamata. Questo è chiamato “late-binding” e attiva il meccanismo del polimorfismo per le chiamate all’interno di una gerarchia di classi.
Ogni nemico può attaccare, ma ogni nemico ha capacità di danno diverse dagli altri nemici. E’ importante che il metodo sia virtuale per costringere ogni classe figlia a dover dichiare quanto danno (secondo parametri) il nemico deve infliggere al giocatore.

```
MatchWindow::updateInventory(const Qstring&)
151    game->getPlayer()->getWeapon()->getImage();
152    game->getPlayer()->getShield()->getImage();
```

string getImage() è un metodo virtuale puro. Ritorna la path (che può essere una path relativa ad un file risorsa di Qt, oppure ad una Pixmap esterna al programma) dell’immagine, dell’elemento, da mostrare nel campo weaponImage oppure shieldImage in base alla tipologia dell’elemento.

FORMATO DI SALVATAGGIO E CARICAMENTO CONTAINER

Nel contesto di SkyrimQt, la classe `Container<T>` ha il template instanziato a `Item*`, ovvero un puntatore alla classe base `Item`, pertanto il contenitore conterrà una sequenza di puntatori ad `Item`.
`Container<Item*>` viene utilizzato per memorizzare il contenuto dell’inventario del giocatore, che potrà avere più armi e scudi per attaccare e difendersi dai nemici.

Al momento del salvataggio della partita, viene creato un nuovo file chiamato [nome-giocatore].gamesave, il cui contenuto è un JSON valido creato grazie ad una `QVariantMap`
che poi si trasforma in un `QjsonObject` che viene poi salvato su disco.
All'interno del Json, vengono salvati i dati del giocatore, il nemico attuale (se presente) e il contenuto dell'inventario, ovvero del `Container<Item*>`.

UTILIZZO DI `Container<T>`

Inserimento:
Alla fine di ogni combattimento vinto, il nemico ha la possibilità di "droppare" un'arma o uno scudo, di vario tipo, livello e punti danno/assorbimento. Il giocatore, ha la possibilità di raccogliere l'elemento (se ha spazio nell'inventario), oppure se lasciarlo a terra e proseguire.

Modifica:
Per la natura di ciò che è rappresentato nell'inventario, non è possibile modificare arbitrariamente i parametri di armi e scudi, ma è solamente possibile rinominarli, per riconoscerli facilmente nell'inventario.

Rimozione:
Si possono rimuovere singolarmente dall'inventario, armi e scudi ritenuti superflui (tranne quelli equipaggiati. In tal caso, bisogna prima equipaggiare un'elemento diverso).
E' anche disponibile uno strumento di pulizia automatico dell'inventario, che permette di rimuovere armi e scudi che sono inferiori al livello del giocatore. Gli elementi equipaggiati non verranno comunque toccati, anche se di livello inferiore.

Ricerca:
E' disponibile una casella di ricerca che permette di individuare velocemente l'equipaggiamento che si cerca, scrivendo il nome dell'elemento oppure il suo livello. Es: "Iron Shield" oppure "Livello 10" (per elencare tutti gli elementi di livello 10).

Visualizzazione:
Solamente i due elementi equipaggiati, l'arma e lo scudo, vengono mostrati con la loro icona sulla parte sinistra della finestra di gioco. Gli altri vengono elencati nella QlistView che rappresenta tutti gli elementi del contenitore.

AMBIENTEDI SVILUPPO

Per realizzare il modello e la versione terminale di SkyrimQt è stato realizzato utilizzando Ubuntu 19.04 Gnome con Clion 2019 e gcc 8.3.0.
Per la realizzazione della GUI tramite Qt, ho impiegato, sempre su Ubuntu 19.04 Gnome, QtCreator e QtDesigner alla versione 4.5.2 basata sulla versione 5.9.5 di Qt e con gcc 5.3.1.

TEMPO RICHIESTO

- Analisi preliminare del problema: 5 ore
- Progettazione modello: 15 ore
- Apprendimento Qt: 5 ore
- Realizzazione GUI: 12 ore
- Debugging: 5 ore
- Testing: 3 ore

QMAKE

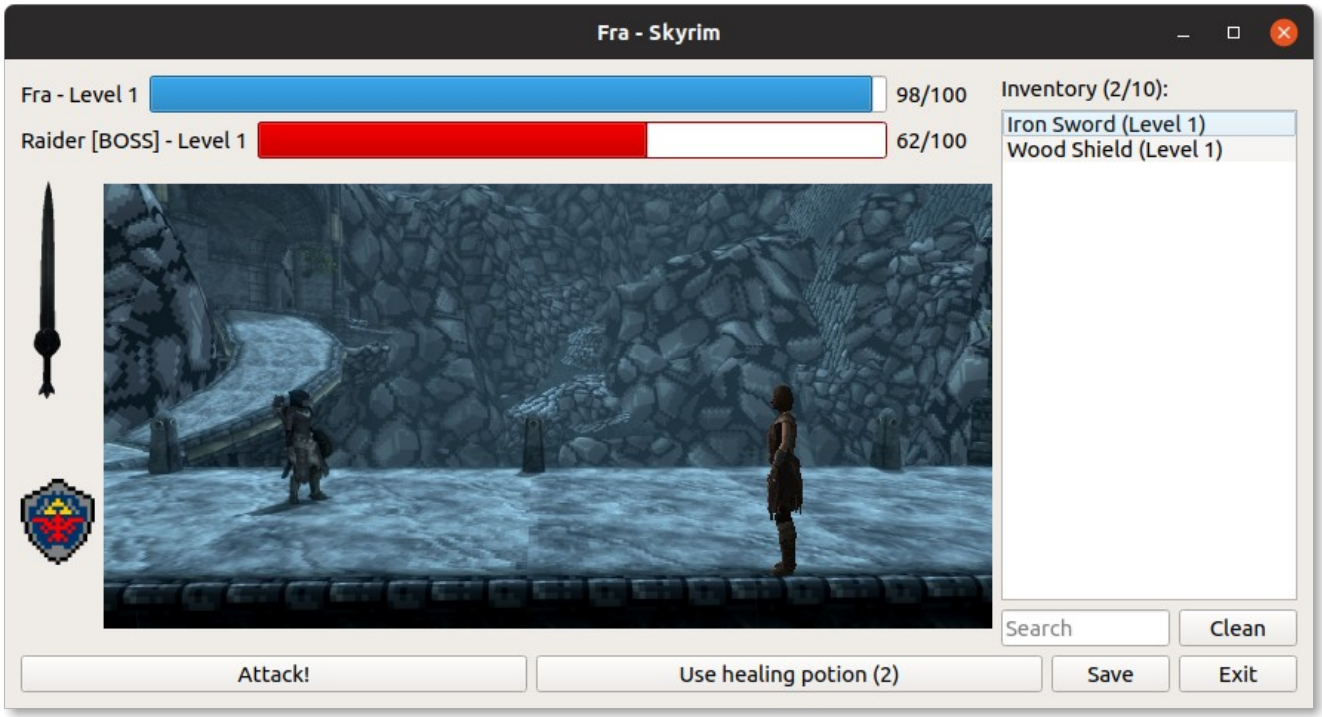
E' necessario compilare utilizzando lo standard C++11 e per questo, ed unico motivo, è fornito il file progetto.pro. Questo perché la libreria FrancescoSorge::Basic fa uso di un meccanismo di numeri random.

MANUALE UTENTE

La schermata principale permette all'utente di iniziare una nuova partita cliccando sul pulsante "New game". Verrà richiesto il nome del giocatore, che è obbligatorio. ATTENZIONE! Se si sceglie un nome che è già stato usato e si salva la partita, il precedente salvataggio verrà sovrascritto!

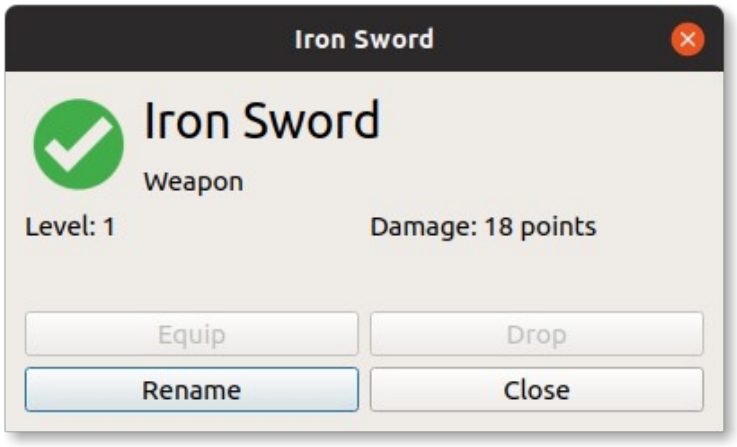
Altrimenti, si può riprendere la partita interrotta precedentemente cliccando sul pulsante "Load game". Verrà presentata una lista con i salvataggi rilevati. Sarà sufficiente cliccare una volta su un salvataggio in lista per riprendere a subito da dove si era rimasti.

Infine, si può uscire dal gioco.



La finestra di gioco principale.
Vengono rappresentate le statistiche del giocatore: il nome, il livello, la barra della salute, la salute attuale e la salute totale raggiungibile.
Alla seconda riga, ci sono le statistiche del nemico (se presente nella scena) con le stesse informazioni.

Nei due riquadri disposti verticalmente vengono rappresentati l’arma e lo scudo equipaggiati. Si possono scambiare con altri oggetti dal proprio inventario, raffigurato sulla destra come una lista.



Cliccando una volta su una riga, viene aperta una finestra per equipaggiare, lasciare e rinominare l’oggetto selezionato. Non è possibile lasciare un oggetto equipaggiato.

E’ inoltre possibile cercare (per nome o per livello) un oggetto nell’inventario e pulire l’inventario dagli oggetti non equipaggiati e che sono di livello inferiore a quello del giocatore.
L’inventario ha un limite massimo di capienza, rappresentato dalla riga “Inventory ([elementi che si stanno trasportando] / [elementi totali che si possono trasportare]).

Nella parte bassa della finestra, ci sono i pulsanti più importanti:

- Il pulsante di attacco “Attack!” permette di attaccare l’avversario con l’arma equipaggiata.
- Il pulsante “Use healing potion ([numero di pozioni trasportate])” attiva il meccanismo di cura che ripristina una parte della salute. Le pozioni curative vengono rilasciate, alle volte, dai nemici una volta sconfitti.
- “Save” salva la partita
- “Exit” esce dalla partita. Se ci sono modifiche non salvate, chiede se salvare, scartare le modifiche o annullare e tornare alla partita.

Quando i nemici vengono sconfitti, c’è una possibilità che lascino a terra armi e scudi, rendendo possibile per il giocatore, decidere se raccoglierli o meno. Una volta raccolti, possono essere equipaggiati. Sfrutta questo meccanismo per diventare sempre più forte ed aumentare le tue difese, con scudi sempre migliori. Ma ricorda, non sempre un oggetto di livello superiore è più forte dell’equipaggiamento che stai usando!

Ad ogni nemico sconfitto, la scena cambia e si va avanti nel livello senza fine!