Progetto PAO 2017

Informazioni

Studente	Victor Dabija
Matricola	1050229
Data Consegna	10-07-2017

Indice

Elenco delle figure

1 Introduzione

La relazione esposta contiene l'analisi dell'applicazione fornita come proggetto per il corso di Programmazione ad Oggetti. Sono stati analizzati e dettagliati gli elementi che possono emanare obreggiature maggiori. Una visione completa di tutte le funzionalità e la loro implmentazione si può avere consultando il codice che è stato integrato con commenti dove necessario.

1.1 Considerazioni personali

Tale proggetto soddisfa i requisiti posti (spero) ma si incentra sulla sperimentazione e l'applicazione prattica delle varie funzionalità della libreria qt nel modo più ampio. Ragione per cui alcune funzionalità del programma finito sono più curate, altre invece meno. Inoltre in alcune situazioni si possono notare divergenze sulla progettazione: in alcune situazioni eredito le funzionalità di un widget definendo un oggetto personalizzato, in altre uso il metodo *addWidget()* fornito dal framework. In breve: **ho cercato di non annoiarmi :)**. Le 60 ore previste per il progetto non mi hanno permesso di sviluppare alcune funzionalità previste nella fase di progettazione, oppure in alcune situazioni, ho dovuto rinunciare ad alcune funzionalità parzialmente implementate. Tali problemi sono dovuti a una curva di aprendimento maggiore di quella prevista per quello che riguarda il framework qt.

Separazione tra parte grafica e logica Dato che si tratta di un gioco in fase di progettazione ho valutato attentamente l'attuazione di tale separazione. In alcune sezioni l'ho ritenuta superflua dato che una mela e..una mela e appunto un eventuale cambio della sua rappresentazione graffica dovrebbe cambiare il suo comportamento. Di fatto ho adottato un metodo virtuale *SetGraphics()* che setta la graffica dell'oggetto (Mela verde, mela gialla o bomba). Un'eventuale modifica della rappresentazione graffica dei drop comunque non implica per forza una modifica del codice ma una semplice sostituzione dell'immagine o del path.
In altre situazioni mi sono impegnato di più a mantenere tale separazione ma comunque cercando di rendere il codice il meno verboso possible e mantenerne la leggibilità. Ne subirò le consegunze sulla valutazione nel caso in cui tale ragionamento si rilevasse completamente sbagliato.

2 Scopo del Progetto

Il programma rappresenta un piccolo gioco realizzato utilizzando l'editor QT. Per muovere il personaggio vengono usati i **tasti DESTRA e SINISTRA**. Lo scopo è quello di raggiungere il massimo punteggio possibile raccogliendo le mele e schivando le bombe.

Figura 1: ScreenShot della schermata di gioco.

- **Mele Verdi:** aumentano il punteggio del giocatore in base al moltiplicatore- minimo 1X.
- **Mele Gialle:** come le mele verdi, aumentano il punteggio del giocatore e inoltre, vanno ad incrementare il moltiplicatore del punteggio.
- **Bombe:** azzerrano il moltiplicatore dei punti e diminuiscono la vita di 1.

La velocità di caduta degli oggetti è casuale e aumenta con il proseguire del gioco. perse le 3 vite a disposizione viene visualizzata la classifica dei giocatori e se il giocatore rientra tra i primi dieci verrà inserito all'interno il suo nickname e il relativo punteggio.

3 Utenti

All'avvio del gioco viene chiesto all'utente di autenticarsi o di creare un'account se lo desidera. L'utente può anche saltare questo procedimento proseguendo come Guest e accedendo lo stesso al gioco. Sono stati resi disponibili 3 tipi di utenti:

- **Guest:** Ha a disponibilità le funzionalità del gioco, può vedere la classifica dei migliori punteggi ma non può modificarla, anche se ha avuto un punteggio migliore di quelli presenti all'interno.
- **Utente Registrato:** Utente che ha seguito la procedura di **SignUp** quindi è munito di username e password. Se a fine partita il punteggio dell'utente registrato rientra tra i migliori 10, il suo username verrà inserito all'interno della classifica.
- **Admin:** Oltre alle funzionalità dell'utente registrato l'admin può decidere di eliminare la classifica dei punteggi oppure eliminare il database degli utenti registrati. Le credenziali dell'utente admin sono: **username: "admin", password: "admin"**.

4 Gerarchie Rilevanti

4.1 Drops

Informazioni

File:	Drops.h
Elementi:	*QObject*
	QGraphicsPixmapItem
	Drop
	Bomb
	Apple
	Multiplier
Utilizzo:	Rappresenta gli oggetti che cadono e con cui il giocatore deve/non deve creare collisioni

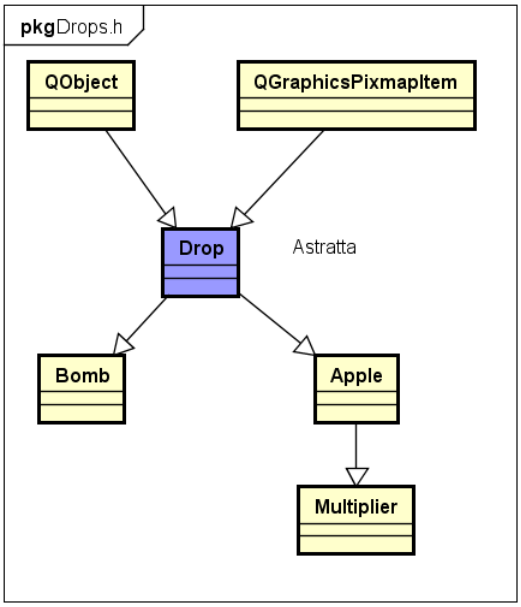

Figura 2: Gerarchia file Drops.h.

Descrizione La Classe Drop è astratta e fornisce i metodi virtuali *Collision()* e *SetGraphics()* per le classi Bomb, Apple e Multiplier. L'implementazione successiva di tali metodi ha la seguente funzione:

- *Collision():* Il Comportamento del oggetto quando collide con il giocatore, oviamente diverso con efetti opposti per il giocatore se si tratta di Apple, Multiplier o Bomb.
- *SetGraphics():* setta la rappresentazione graffica del oggetto in base al tipo, ovviamente Bomb ha una rappresentazione diversa da Apple.

Durante la partita tali metodi verranno invocati su un **puntatore polimorfo** a un oggetto di tipo *Drop** e si comporteranno in base al tipo dinamico del oggetto di invocazione.
Gli altri metodi Sono abbastanza generici, va nominato però *Drop::Update()*:
Si tratta di uno *slot* publico che viene **connesso al timer statico di Game**. Lo scopo di tale slot è quello modificare la posizione dell'oggetto a ogni tick del timer in modo da simulare la gravità.

4.2 User e UserDB

Informazioni	
File:	users.h
Elementi:	User
	Admin
	UserDB
Utilizzo:	Rappresenta i tipi di utenti registrati e Il modulo che li gestisce nella ram e sul hard disk.

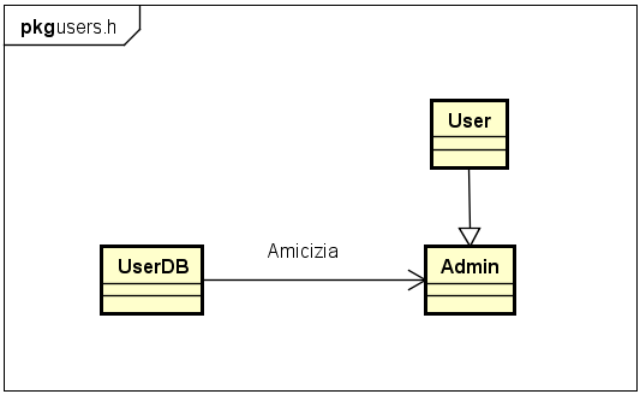

Figura 3: Gerarchia file users.h.

Descrizione User e' una banalissima classe rappresentata da un username e password integrata con metodi per gestirla e serve a rappresentare le credenziali degli utenti.
La classe Admin eredita le funzionalità di User ma ottiene la possibilità di gestire i moduli UserDB e ScoreManager grazie alle amicizie dichiarate di tali classi.
UserDB contiene gli utenti registrati nel sistema e informazioni sull'utente corrente se presente. Ovviamente tale modulo è stato munito di tutte le funzionalità necessarie per la loro gestione. Il login come admin e' stato limitato al hard coding per esigenze di tempo ma ovviamente è facilmente estendibile quindi va specificato che nella versione attuale UserDB contiene soloo utenti registrati.

- *Collision():* Il comportamento del oggetto quando collide con il giocatore, oviamente diverso con efetti opposti per il giocatore se si tratta di Apple, Multiplier o Bomb.
- *SetGraphics():* setta la rappresentazione graffica del oggetto in base al tipo, ovviamente Bomb ha una rappresentazione diversa da Apple.

Durante la partita tali metodi verranno invocati su un **puntatore polimorfo** a un oggetto di tipo *Drop** da un oggetto *Player* e si comporteranno in base al tipo dinamico del oggetto di invocazione.
Gli altri metodi Sono abbastanza generici, va nominato però *Drop::Update()*:
Si tratta di uno *slot* publico che viene **connesso al timer statico di Game**. Lo scopo di tale slot è quello modificare la posizione dell'oggetto a ogni tick del timer in modo da simulare la gravità.

5 La classe Game

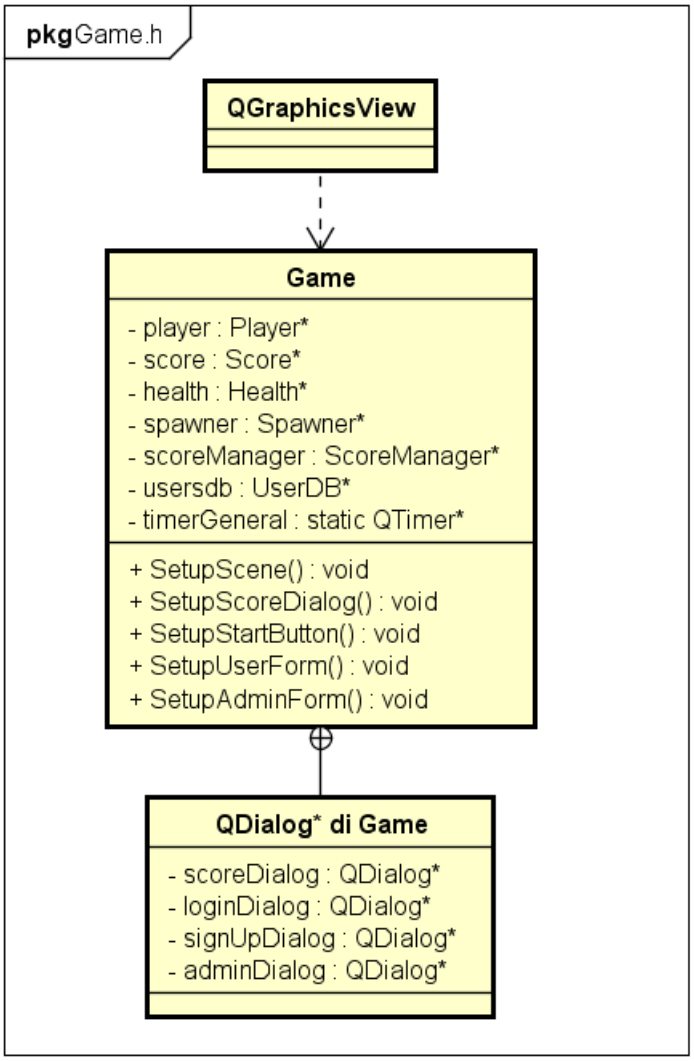

Figura 4: Parziale Rappresentazione Graffica della classe Game.

Descrizione La classe *Game* viene creata dal *main* e racchiude a cascata tutti i moduli per il funzionamento del sistema.
La *Figura 4* rappresenta approssimativamente la sua struttura. I vari componenti interagiscono tra loro in relazione alle funzionalità e al accesso fornito all'esterno di ciascuno di essi. I vari metodi Setup forniti vengono invocati alla creazione del oggetto Game oppure con chiamate specifiche quando necessario.

Dialogs I vari *QDialog* permettono di interagire con il modulo che li rappresenta attraverso vari slot e funzionalità fornite.

6 Altre sezioni e notazioni

6.1 Distruttori profondi:

Dove necessario sono stati ridefini i distruttori di alcune classi onde evitare di lasciare garbage.

6.2 Classi Annidate:

Vi sono presenti alcune classi annidate con visibilità diversa in base alle esigenze di progettazione: la classe *nodo* di Container ad esempio non è visible all'esterno dato che serve a rappresentare e organizzare elementi all'interno della classe parent.

6.3 Alcune Funzionalità non usate:

In alcuni casi le classi sono state progettate fuori dal contesto e di conseguenza dotate di alcuni metodi non utilizzati. Tali metodi non sono stati rimossi per aiutare e facilitare l'eventuale aggiunta di nuove funzionalità.

6.4 Sviluppo Incrementale:

Il programma è stato sviluppato adottando un approcio incrementale. Per prime sono state implementate le funzionalità che riguardano il gioco, successivamente sono stati aggiunti i moduli per la gestione del punteggio e degli utenti. Nella seconda fase inoltre, sono state sperimentate maggiormente le varie funzioni e annidamenti dei QWidget fornite dal frameWork QT. Tali considerazioni dovrebbero aiutare a rafforzare l'asse del **Open World Assumption**. Quindi rendere il codice facilmente estendibile.

6.5 Ore di Lavoro:

60h circa, gran parte impiegate a studiare le funzionalità e non per la codifica.