Relazione per “Programmazione ad Oggetti”

Fabio Pazzini
Aldo Dushku

20 Febbraio 2015

Indice:

Capitolo 1
Analisi
1.1 Requisiti

L’applicazione consiste in un videogame di tipo platform bidimensionale, a cui abbiamo dato il nome Ninja Platform: lo scopo, è riuscire a superare tutti i livelli ed i relativi ostacoli fino al suo completamento.
La meccanica di gioco è molto semplice, in quanto prevede il movimento verso destra/sinistra, alto/basso.
Un livello, genericamente parlando, è composto da nemici, ostacoli, e piattaforme su cui muoversi.

1.2 Problema

Come principale problema di un gioco di questo genere troviamo l’interazione tra i vari componenti di un livello, ovvero tra i nemici, personaggio, e la mappa stessa. Un altro problema concerne lo spostamento della telecamera mentre il personaggio si muove, tenendolo al centro della schermata.

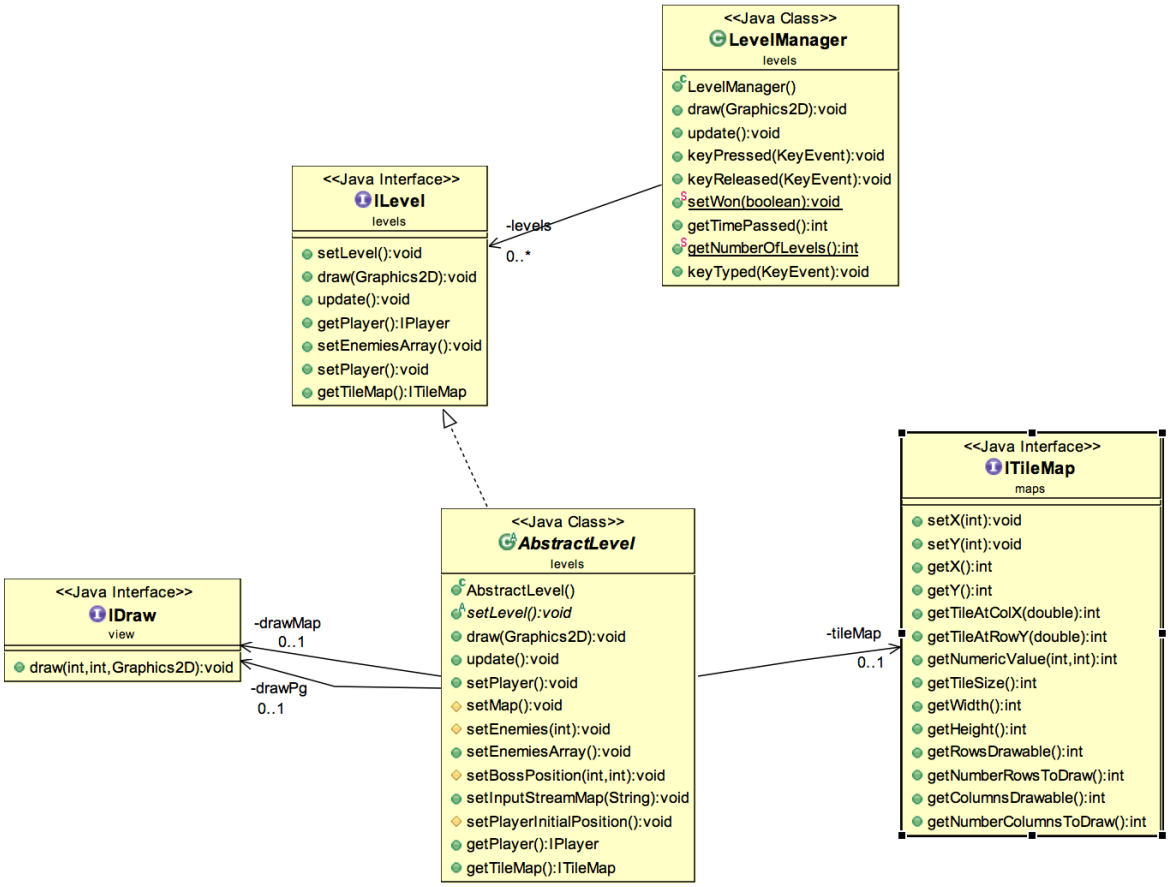
Capitolo 2

2.1 Design

Il progetto è stato realizzato in ottica Model-View-Controller.

La parte di Model è costituita da mappe, personaggio e nemici.
La parte di View è costituita dalla rappresentazione grafica degli stessi.
La parte di Controller è costituita dalla gestione del gioco nel suo insieme, coordinando e facendo comunicare le due parti precedenti.

UML che mostra l'applicazione del pattern MVC nel progetto:



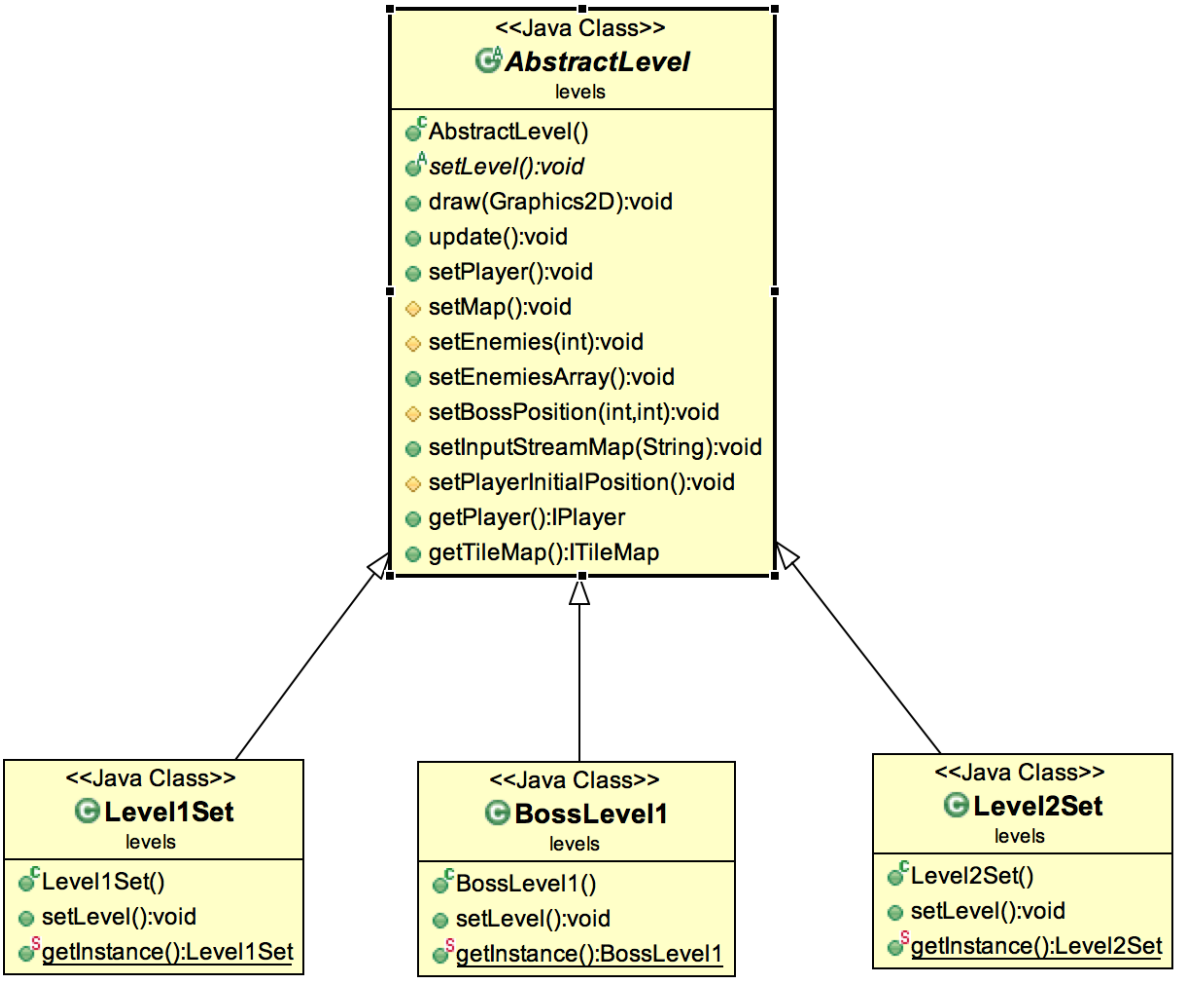
La parte di Controller è destinata prima a LevelManager, che può avere uno o più istanze di AbstractLevel il quale implementa ILevel e gestisce la parte di Model identificata da ITileMap, IPlayer ed IEnemy, facendola comunicare con la View rappresentata da IDraw.
Sostituire in blocco la View, in questo modo, non causa enormi problemi né alla parte di Controller né a quella di Model.

2.2 Design dettagliato

Il design pattern utilizzato, di cui abbiamo notato un particolare vantaggio nell'utilizzo è il Template Method.

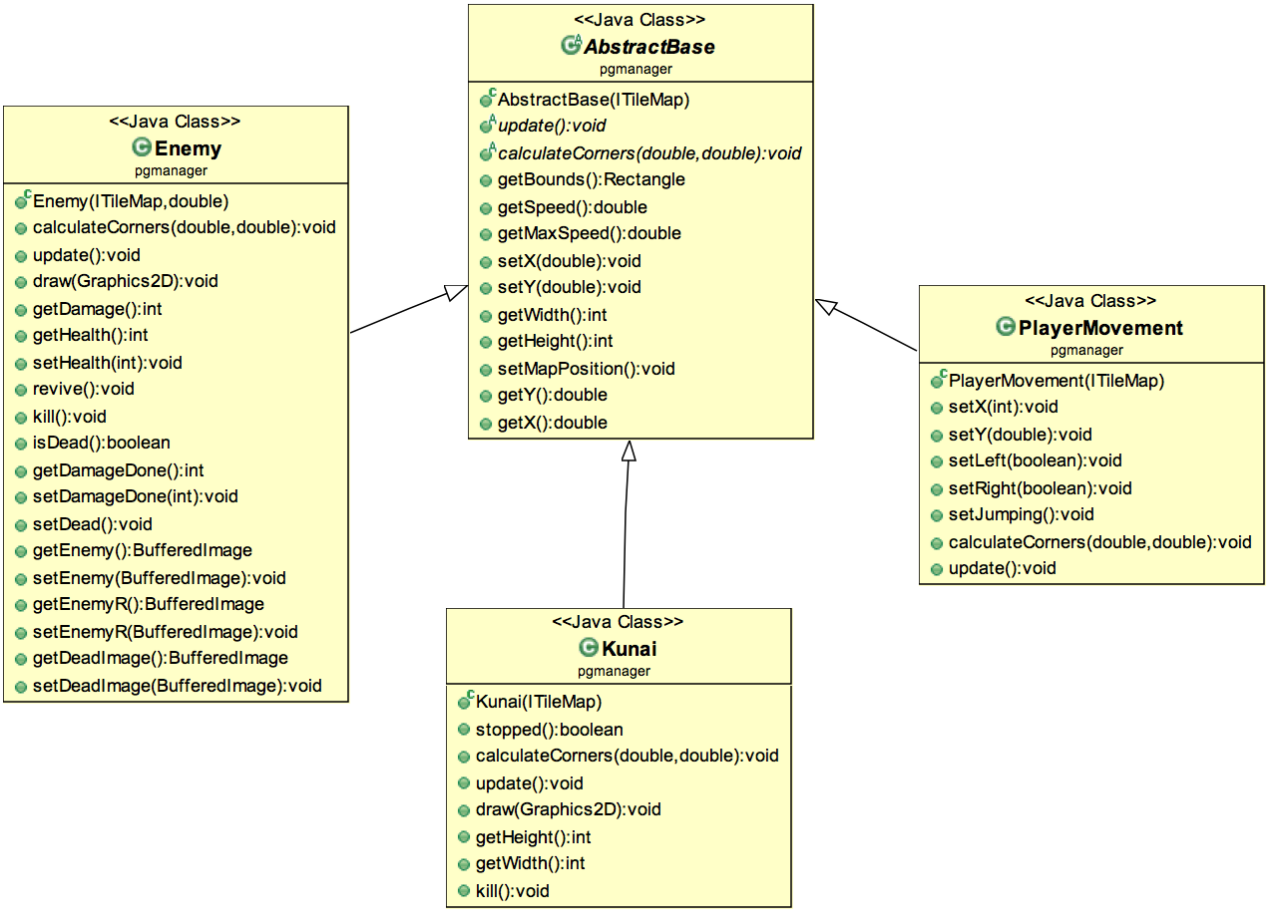
Identifichiamo due parti del progetto in cui è stato applicato:

Livelli:



AbstractLevel rappresenta un livello generico. Possiede metodi che sono in comune tra ogni livello, ed un metodo astratto che viene implementato dalle singole specializzazioni, ovvero Level1Set, Level2Set e BossLevel1. Queste, presenteranno diverse implementazioni di tale metodo dal momento che rappresentano i livelli del gioco e che quindi dovranno possedere caratteristiche differenti.

Personaggi:



Il cuore di questo UML è la classe AbstractBase, che accomuna i movimenti ed i metodi principali di qualsiasi oggetto che deve interagire con la mappa. Le sue specializzazioni sono Enemy, PlayerMovement e Kunai. A loro volta, le singole classi possono ulteriormente specializzarsi per consentire una buona riusabilità del codice, come nel caso delle classi Dragon e Boss (non presenti nell'UML).

Capitolo 3
Sviluppo
3.1 Testing automatizzato

Dal momento che la maggior parte del progetto si sviluppa graficamente, lasciando poco spazio all'interazione da parte dell'utente (eccetto la pressione dei tasti per il movimento), non abbiamo eseguito test automatizzati ma prettamente manuali.

Il team ha programmato su due sistemi operativi Mac OS X (Yosemite) ; di conseguenza, per testarne il corretto funzionamento su ogni SO (Windows 7/8.1, Linux) , abbiamo eseguito test su macchine virtuali e fisiche non nostre. In particolare, le caratteristiche hardware su cui abbiamo testato sono:
1) RAM: da 4GB a 16GB;
2) CPU: Dual Core, i5 (2,6GHz), i7;
3) Risoluzioni: 800x600, 1900x1600, 2560x1600.

3.2 Divisione dei compiti e metodologia di lavoro

Il team è composto da due studenti, Aldo Dushku e Fabio Pazzini.

Dushku si è occupato della parte di progettazione ed interazione dei personaggi con la mappa e tra di loro, del menù principale di gioco (Launcher) e della gestione del punteggio totalizzato a fine gioco.
Pazzini, invece, ha realizzato la struttura sottostante, ovvero gestione di livelli, mappe, grafica e risoluzione, e i thread di gioco.

Per aggiornare il nostro lavoro abbiamo utilizzato BitBucket e Mercurial committando le modifiche che venivano applicate.

Ci sono parti che necessitavano di un'integrazione da parte dell'altro membro del team, per esempio le collisioni tra personaggio e mappa, e la rappresentazione grafica degli oggetti in essa. Per integrare le parti ci siamo incontrati tre volte ed abbiamo trovato la soluzione che ci risultava maggiormente appropriata e riutilizzabile.

La divisione dei compiti è stata equa ed entrambi siamo rientrati circa nelle ore prestabilite.

3.3 Note di sviluppo

Particolari difficoltà a livello di codice sono state incontrate soprattutto nella prima fase del progetto per capire come si articoli un gioco, in quanto non avevamo mai realizzato nulla di simile.
Inoltre, un'altra importante difficoltà è stata riscontrata nell'adattamento della dimensione del frame di gioco a seconda della risoluzione dello schermo su cui viene disegnato: infatti, se avessimo trattato questo punto in maniera sbrigativa, testandolo prima su uno schermo a bassa risoluzione e poi su un 4K (o più) la differenza sarebbe stata esagerata ed assolutamente non appropriata per il progetto (e, più in generale, per un videogioco). Dunque, per come è stato realizzato, nel frame di gioco si vedrà sempre la stessa porzione di mappa ed ogni oggetto sarà adattato secondo la risoluzione del monitor.

Per quanto riguarda la strutturazione di un gioco, abbiamo ricercato sul Web risorse quali video e tutorial che spiegassero come procedere in maniera molto generica: inoltre, ci siamo rivolti a Stack Overflow qualora avessimo trovato dei problemi a livello pratico. In particolare, linkiamo due video a cui abbiamo fatto riferimento:
1) https://www.youtube.com/watch?v=FUgn-PA7yzc (telecamera)
2) https://www.youtube.com/watch?v=sLKniK9xZw8 (personaggio)

Ed un link a Stack Overflow per l'intersezione tra gli oggetti:
1) http://gamedev.stackexchange.com/questions/27299/android-java-rectangle-collision-detection-not-working

Capitolo 4
Commenti finali
4.1 Conclusioni e lavori futuri

Il team si è trovato molto bene nel lavoro di gruppo, infatti i membri sono riusciti a collaborare e a interagire facilmente.
Secondo noi, il lavoro è riuscito discretamente. Ci siamo accorti che lavorando in gruppo si ha modo di imparare molto grazie alle conoscenze dell'altro.

Un punto di forza molto importante del progetto è la stabilità del codice e la sua composizione; dopo numerosi test, ha risposto sempre in maniera corretta. Inoltre, la maggior parte del progetto è totalmente riutilizzabile per altri giochi e permette l'aggiunta, in maniera molto rapida, di nuove funzionalità senza stravolgere nulla.

Un punto di debolezza, invece, è il non totale distaccamento delle parti dell'MVC, che risulta progettato non perfettamente.

Il gioco verrà sicuramente nei prossimi mesi ampliato con nuove modalità, poteri, nemici e livelli per arrivare a creare un gioco più completo.

Appendice A
Guida utente

Il gioco risulta molto facile da utilizzare, i tasti principali infatti sono:
1) Frecce direzionali o lettere per muoversi (settabili in Options);
2) Tasto E per lanciare i kunai.