Relazione di
“Cory in da house”

Rafal Pacchioli
Adrian Chrobak

1. Analisi del progetto

1.1 Requisiti del gioco

Il software in questione mira a essere un gioco dove il protagonista, Cory, si muove all'interno di stanze randomicamente generate, di dimensione prefissata. Cory nel corso della sua impresa dovrà affrontare diversi tipi di mostro, oltre che alla fame che deve soddisfare raccogliendo cibo.
Le stanze possono essere di diverso tipo, scelto casualmente e ci si sposta da una stanza all'altra tramite porte.
Le stanze possono contenere:
-Avversari da affrontare
    Che una volta incontrati avvieranno una fase di combattimento a turni
-Oggetti da raccogliere
    Che raccolti andranno a migliorare le sue statistiche, quali resistenza o forza d'attacco, oltre che placare il suo appetito.
-Oggetti inanimati
    Che saranno sulla mappa per dare fastidio, e con i quali Cory avrà minima possibilità di interazione.

La fase di combattimento tra Cory e i suoi nemici è svolta in un sistema automatico sempre a turni, in una finestra a parte, nella quale algoritmi di lotta svolgeranno i calcoli necessari per determinare chi avrà la meglio con quanti danni subiti.

Lo scopo del sistema vuole essere sopravvivere il più a lungo possibile alle difficoltà che il mondo lancia al giocatore, fino alla inevitabile morte del protagonista per malnutrizione o danni subiti.

1.2 Problemi

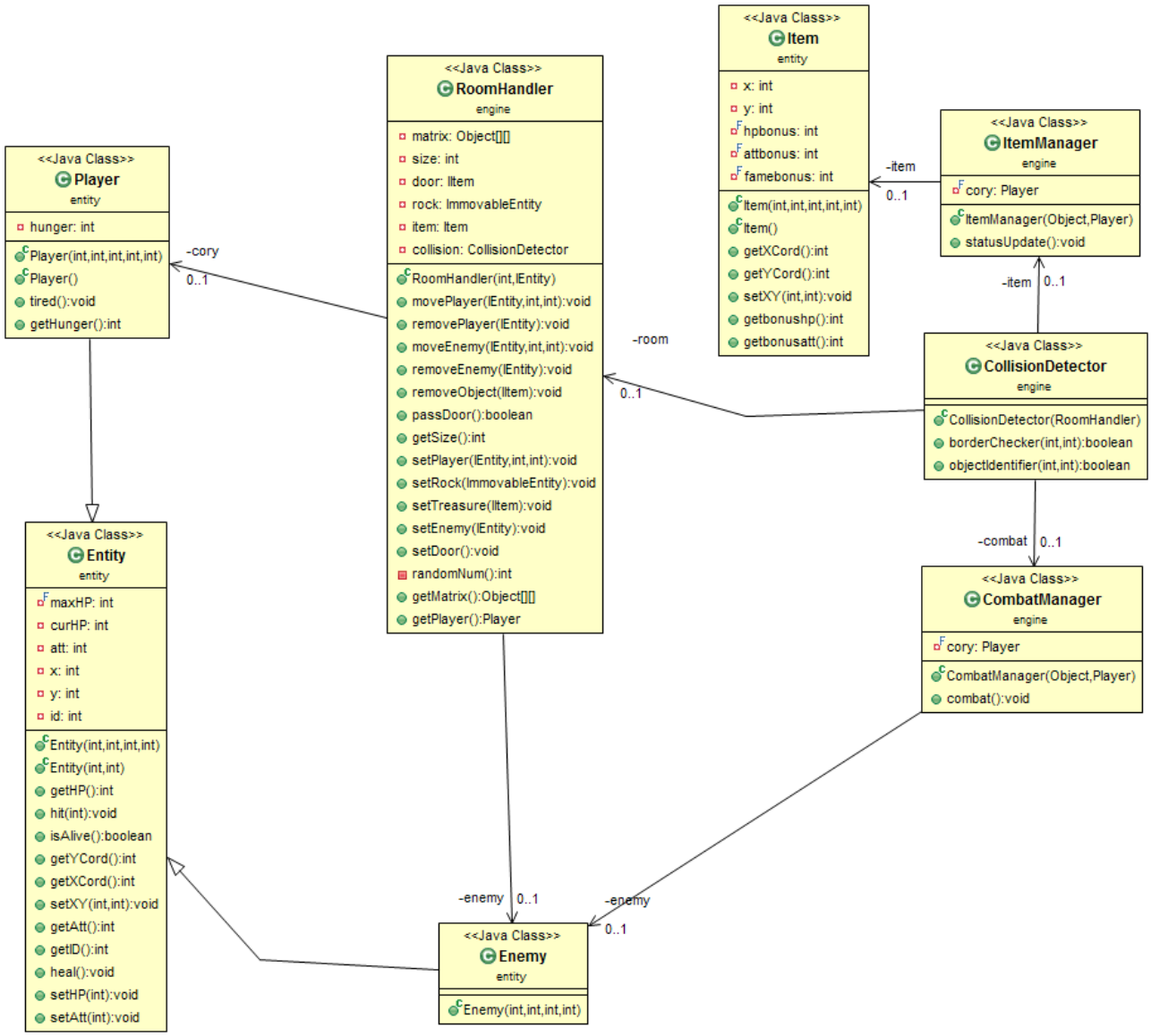
Il problema principale consiste nel fatto che i due sviluppatori sono non eccessivamente furbi e preparati in materia.
La difficoltà primaria nello sviluppo sta nel creare un ambiente interattivo nel quale il giocatore riesce a muoversi, e che a sua volta è “vivo” e popolato, il tutto generato sul momento con un certo criterio logico. Il tutto senza cadere in duplicazione di codice, con uso decente di ereditarietà.
Altro problema è il riuscire a gestire in modo funzionale il passaggio tra le varie stanze, e il gestire la schermata di combattimento.
Infine, separare in modo razionale la parte visiva dal modello riducendo al minimo le dipendenze, e permettendo una facile manutenzione dell'aspetto grafico, così come libertà nel migliorare la logica dei livelli inferiori.

2. Design

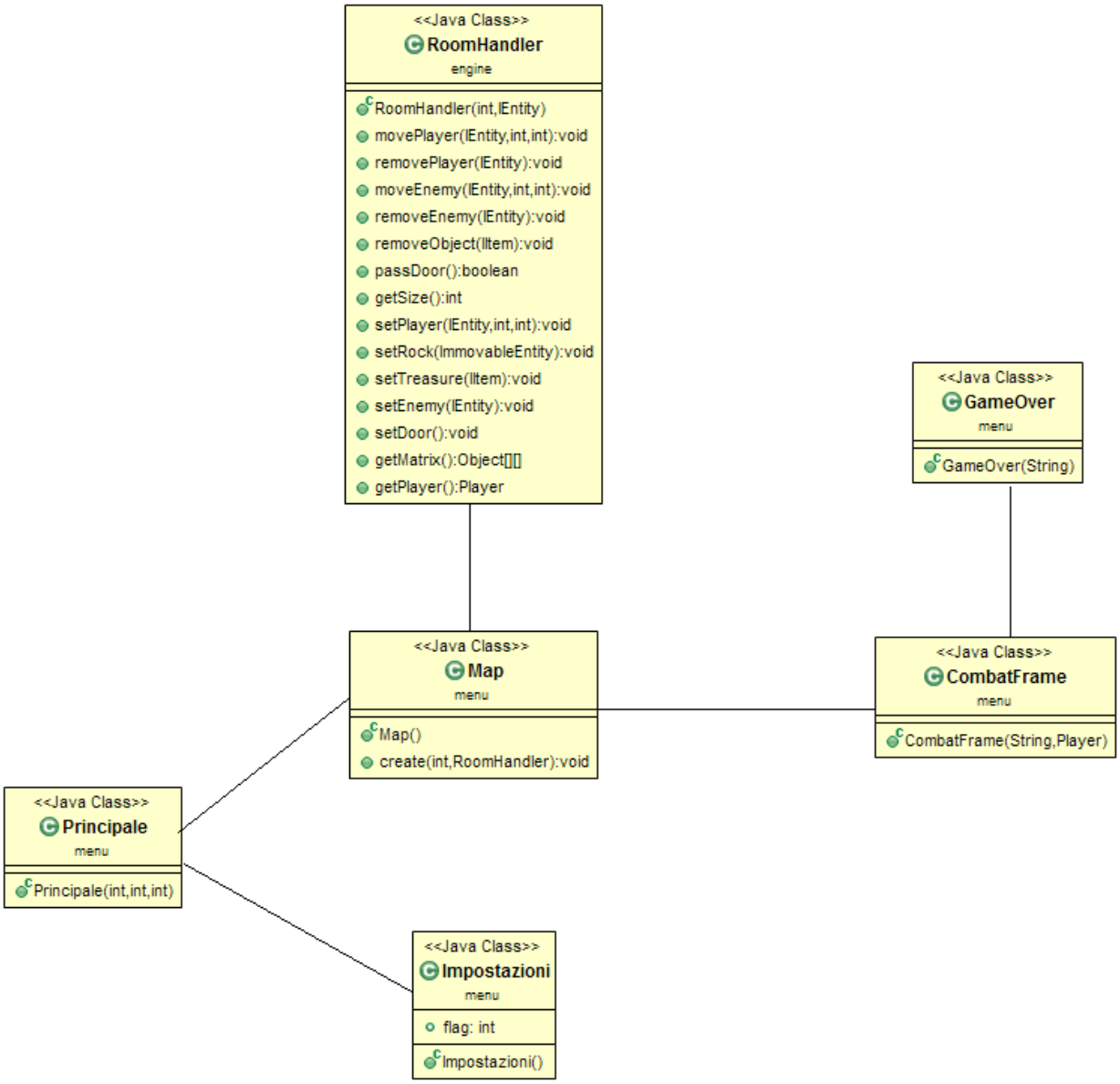
2.1 Architettura

Il modello è costituito da una classe e la sua relativa interfaccia, ossia RoomHandler, che si occupano di creare la stanza e di gestire le entità presenti al loro interno, quali nemici, oggetti, porte e giocatore. Tutte le entità “vive” derivano dalla stessa Interfaccia, e a loro volta dalla sua implementazione per sfruttare al meglio il riuso del codice. Stesso discorso vale per gli oggetti inanimati presenti nel mondo di gioco.
Parlando del modello più in dettaglio, si tratta di una matrice di oggetti, che passata alla view verrà costruita graficamente. In base agli oggetti contenuti all'interno della stanza le singole caselle vengono gestite.
La parte del controller è gestita dall'interno della stanza tramite getter e setter che permettono di muovere gli oggetti nella matrice, e che quindi può essere poi semplicemente usata per aggiornare la view.
Quindi, in linea puramente teorica, volendo si può sfruttare la classe RoomHandler con un tipo di interfaccia grafica diversa, e probabilmente migliore di quello da noi applicato.

Funzionamento a grosso modo:
La prima cosa che succede all'avvio è la creazione di un menu. Nelle impostazioni si può scegliere il grado di difficoltà, uscire o avviare un nuovo match. In quest'ultimo caso si apre una nuova finestra con una stanza caricata dentro.
La classe RoomHandler al momento della sua creazione si popola di oggetti. La prima cosa da fare è sistemare il giocatore in una posizione, poi la porta di uscita. Infine, oggetti inanimati quali rocce e tesori. Il tutto viene poi passato alla parte di view, che in base al contenuto della stanza modellerà l'interfaccia.
A ogni movimento del giocatore la stanza si aggiorna, intercettando con opportuni controlli collisioni varie.
In caso di collisione con un mostro si avvia la parte di combattimento in una finestra a parte. In caso di collisione con parti di mondo o rocce il movimento viene annullato. Il gioco così prosegue fino alla morte del personaggio, che una volta avvenuta chiuderà la finestra e farà tornare al menu.



UML che descrive il funzionamento del modello/controller



UML che narra della relazione fra le entità del menu e della mappa

3. Divisione dei compiti e metodologia di lavoro

Pacchioli si è occupato della parte di progettazione e creazione di un modello, delle entità presenti nel gioco, della stanza e delle relative interfacce e relazioni fra gli oggetti di gioco. Per dirla breve, il package “entities” ed “engine”.
Chrobak si è occupato della creazione del menu di gioco, della parte di combattimento, e della mappa in base al modello passatogli da sotto, ossia il rimanente.
Dettagli quali arte, e piccole finezze in giro sono stati affrontati a turno da entrambi, per quanto Pacchioli abbia premuto di più sulla pulizia del codice, e Chrobak sul suo funzionamento.
La parte di integrazione del codice è avvenuta tramite dei “semplici” passaggi di classi all'interno delle classi dedicate alla view, quindi ci si può ritenere soddisfatti della riduzione minima delle dipendenze fra le due parti di lavoro.
Non senza difficoltà, poiché la definizione di “essenziale” di entrambi è stata più volte in contrasto, principalmente nel dover estrarre i singoli oggetti dalla classe stanza.
Per quanto la parte di modellazione sia stata eseguita dal primo, le difficoltà dovute alla ignoranza di entrambi sul linguaggio sono state affrontate dal secondo, quindi si potrebbe definire il carico di lavoro distribuito in modo equo.

L'utilizzo di bitBucket e Mercurial è stato essenziale nello svolgimento dell'opera, poiché ha permesso di avere in tempo reale modifiche essenziali in alcuni momenti, permettendo inoltre di lavorare in modo veloce sia da casa che in laboratorio a Cesena.
Ci si rende anche conto del quanto a parere nostro il DVCS non sia stato usato pienamente, dato che ci siamo limitati al solo uso di push, pull e sporadici merge, ma anche a questo livello è stato di grande aiuto.

4. Commenti finali

4.1 Commento sul lavoro e futuro

Ci rendiamo benissimo conto del fatto che il progetto è stato affrontato con superficialità, in non troppo tempo e con delle scarsissime conoscenze di fondo con una progettazione, a tutti gli effetti, scarsa.
Ciò che è nato è un prodotto in qualche modo funzionante ma con una lunga serie di difetti dovuti principalmente alla carenza di tempo.
In caso si volesse portare avanti il progetto, decisamente bisognerebbe riorganizzare leggermente la relazione tra controller e modello, così come trovare librerie migliori per la grafica e impegnarsi nella creazione di arte.

4.2 Istruzioni di gioco

Avviato il gioco, il giocatore si muove usando i tasti WASD.
Tasto E per aprire l'interfaccia del personaggio
Tasto Q per uscire dalla mappa

Siccome ci sono stati grandi problemi con la grafica, momentaneamente la grafica è rappresentata come segue:
Cory: Tondo blu
Nemici: Esagono rosso
Rocce: Triangolo giallo
Potenziamenti: Quadrati verdi