Relazione Progetto Programmazione

Emanuele Pischetola

December 2022

1 Divisione dei ruoli

I ruoli sono stati suddivisi in base al carico di lavoro. La suddivisione è avvenuta nel seguente modo:

- **Schermo** (Emanuele Pischetola): si occupa di gestire la stampa sul terminale attraverso la librerie curses/ncurses.h.
- **Entità e Oggetti** (Emanuele Pischetola): si tratta delle classi che descrivono tutti gli elementi rappresentabili nel gioco. Sono state realizzate tramite l'uso di eriditarietà
- **Fisica di gioco** (Saad): in generale si occupa di prendere in input lo stato di gioco corrente e restituire il nuvo stato di gioco, in base a ciò che è avvenuto.
- **Mappa** (Diego Ammirabile): definisce la mappa corrente e di conseguenza si occupa di memorizzare le stanze visitate e il loro contenuto
- **Game Loop** (Diego Ammirabile):

1.1 Schermo

Lo schermo è implementato nei file Screen.hpp, GameInterface.hpp, GameMenu.hpp, GameControls.hpp. La classe Screen è generica, mentre le altre rappresentano una specifica finestra.

1.2 Entità e Oggetti

Le entità di gioco sono tutte figle della classe Core, che rappresenta l'elemento più generico possibile. Le classi direttamente figlie sono Entity ovvero gli elementi "vivi". ItemOnGround cioè gli oggetti, e Wall i muri. I file sono Core.hpp, ItemOnGround.hpp, Entity.hpp, Wall.hpp, Bullet.hpp, Player.hpp, Hostile.hpp

Mentre gli oggetti sono i figli della classe Item, che possono essere Consumabili come pozioni e chiavi, oppure dei potenziamenti alle statistiche come attacco e vita. Sono tutte implementate nel file Equipment.hpp

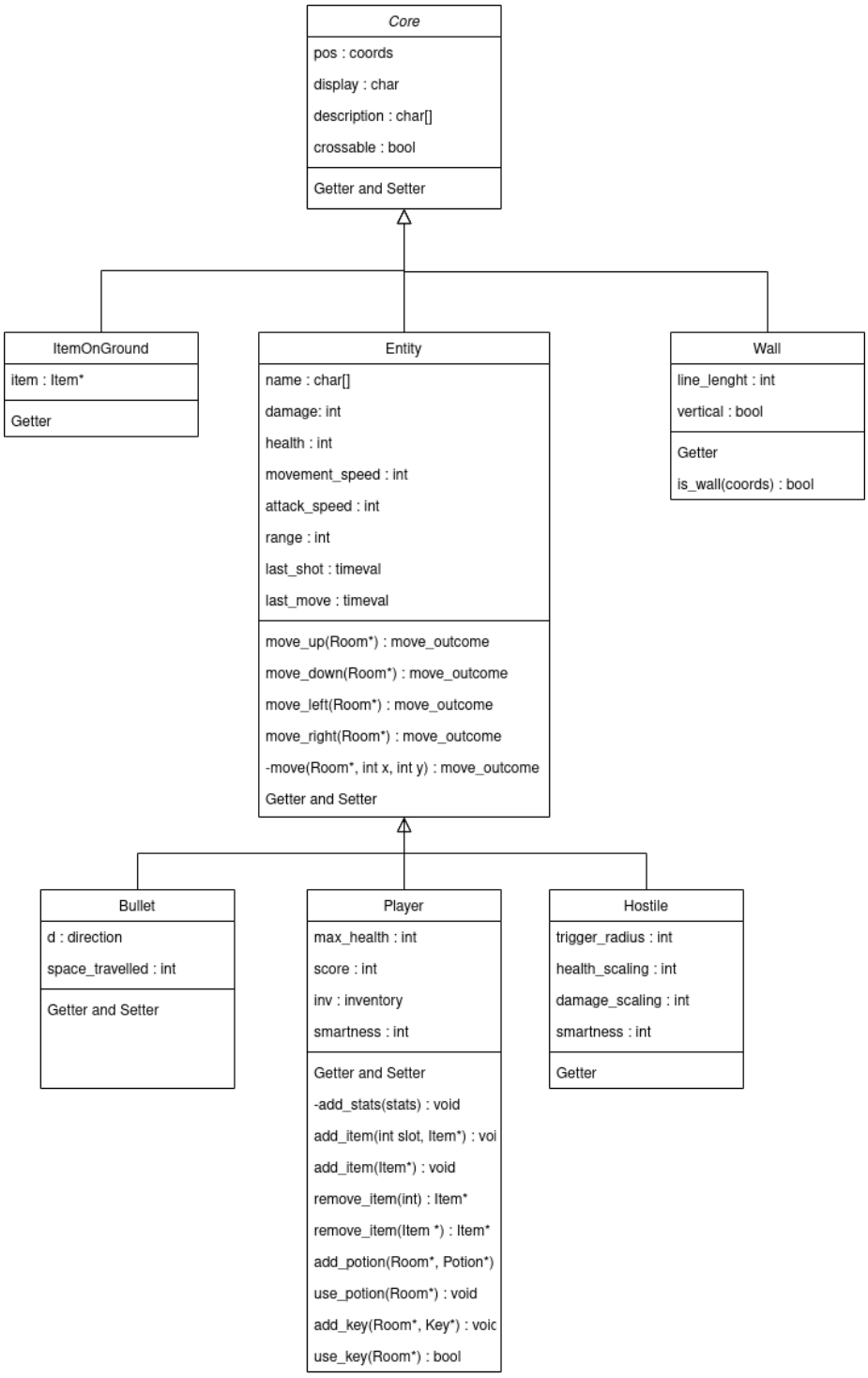

Figure 1: UML delle entità di gioco

1.3 Fisica di gioco

I metodi della fisica di gioco sono raccolti nei file physics.h/physics.cpp,

1.4 Mappa

La mappa e i relativi metodi sono descritti nel file Map.h, ed è stata implementata tramite una struttura. Mentre la stanza è una classe implementata in Room.hpp. Inoltre per realizzare la mappa sono state implementate due strutture dati: la lista e la coda. Sono nei file List.hpp e Queue.hpp

1.5 Altro

Altri file che non ho menzionato sono constants.h che contiene tutti i valori costanti e alcune strutture, time_handle.h che ha alcune funzioni utili per calcolare il tempo in millisecondi, questi file sono di responsabilità di E. Pischetola. Mentre gli eventi, nei file RoomEvent.hpp e Events.hpp, sono di responsabilità di D. Ammirabile.

2 Scelte implementative

Abbiamo pensato di implementare il progetto divendendolo in 3 componenti fondamentali: la vista, il modello e il controller.

2.1 Modello

Il modello è il componente che si occupa di tenere i dati del gioco. Quindi è rappresentato dalla mappa e dalle classi Room, Core, Item e le relative classi figlie. Esso infatti viene aggiornato dal controller ogni ciclo del game loop per rispecchiare la situazione di gioco.

Esempio: quandi il giocatore preme w per muoversi verso l'alto le coordinate dell'oggetto player vengono cambiate.

2.2 Vista

La vista è il componente che si occupa esclusivamente della visualizzazione grafica sul terminale, ed è rappresentato dalle classi Screen, GameMenu, GameInterface, GameOptions. Il compito di queste classi è infatti leggere la versione più aggiornata del modello, quindi l'oggetto di tipo Room relativo alla stanza corrente, e stampare sul terminale la corrispondente rappresentazione grafica.

Esempio: la vista vede che le coordinate del player sono cambiate, quindi cancella il carattere del player nella posizione precedente, e poi stampa il carattere del player nelle nuove coordinate.

2.3 Controller

Il controller si occupa di modificare il modello, è rappresentato da physics.h. I suoi metodi vengono invocati dalle altre classi.

2.4 Eventi

Per gestire la comunicazione tra i componenti abbiamo implementato gli eventi. Ogni volta che il controller cambia qualcosa nel modello aggiunge il relativo evento in una coda. La coda è legata alla stanza corrente, infatti è un paramentro della classe Room. Quando la stampa della vista viene invocata, essa legge la coda degli eventi ed esegue la routine associata ad ogni evento che trova. Gli eventi si possono trovare nel file Events.hpp