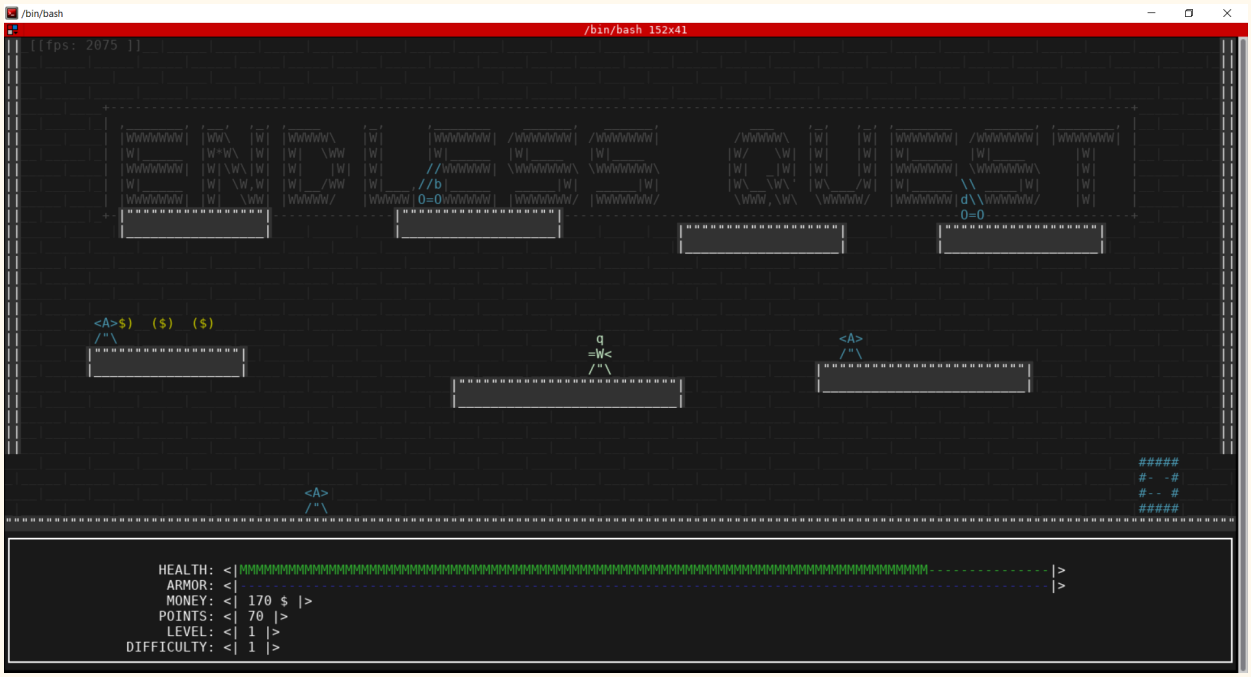
Corso di Programmazione
Prof. Cosimo Laneve

ENDLESS QUEST
ASCII PLATFORMER

Marco Reina, Lorenzo Guerrini



INTRODUZIONE

Il gioco è scritto in C++ con l'utilizzo della libreria ncurses per gestire la grafica.
Sono state utilizzate anche altre librerie: chrono.h ci permette di calcolare precisi lassi di tempo, time.h e stdlib.h vengono usati per generare numeri randomici.
Il gioco è uno sparatutto in cui il player si trova ad affrontare una serie di nemici differenti su più livelli random. La struttura del livello è composta da una serie di piattaforme disposte su più altezze e uno o più nemici. Ai lati sono presenti due porte, attraverso quella di destra si può avanzare di livello se i nemici del livello corrente sono stati completamente eliminati. Il punteggio aumenta per ogni nemico eliminato o moneta collezionata. È possibile accedere anche al menù con cui si può aprire il mercato o cambiare livello, tornando indietro.

Struttura del codice

Alla base del funzionamento del gioco c'è un ciclo while che gira continuamente. Ogni ciclo di gioco si può suddividere in due fasi: nella prima si legge dalla tastiera in input, aggiornando di conseguenza tutte le classi, mentre nella seconda fase viene stampato tutto in output.
La durata di questo ciclo è salvata nella variabile **`deltaTime`**. Sapere quanto tempo è passato dall'ultimo 'frame' ci permette di utilizzare le leggi della fisica per calcolare, per esempio, la posizione di un proiettore che si muove a una determinata velocità. Inoltre si possono gestire animazioni come quella delle monete e il danno subito da ogni entità.
I livelli sono gestiti tramite una lista la cui testa punta al livello più alto raggiunto.

Classi principali

Nel gioco ci sono varie classi ognuna con un preciso scopo. Alcune cose che possono avere in comune sono i metodi **update** e **print**. I metodi chiamati update o simili servono per aggiornare lo stato del gioco, cambiando posizione, vita, e altri campi della classe, in relazione a deltaTime e all'ambiente circostante. I metodi chiamati in modo simile a print invece si occupano solo di stampare in terminale lo stato aggiornato del gioco. È importante che l'implementazione sia distaccata il più possibile dalle visuali.

1. ### Classe level
   La classe Level gestisce tutte le altre classi. Un'istanza di essa rappresenta una stanza del gioco. E questa istanza contiene puntatori a ogni oggetto presente, organizzati in diverse **liste**. Gli oggetti come le piattaforme vengono generati casualmente, mentre i nemici vengono implementati in base al numero del livello e alla difficoltà fornita.
   Questa classe permette alle altre di relazionarsi tra loro. fornendo l'utilissimo metodo **`check()`** che scorrendo tutte le liste calcola se in una determinata direzione è presente un altro oggetto

2. Classe entity

La classe entity è padre di tutti i nemici e del player. Implementa tutti i metodi base che permettono di muoversi sotto l'effetto della gravità. Inoltre ha vari campi come i punti vita che sono in comune con tutte le entità, e fornisce un'interfaccia basilare per leggerli e modificarli, come il metodo `getHealth()` o il metodo `hurt()`.

3. Classe bulletManager

La classe bulletManager gestisce tutti i proiettili presenti in un unico livello. Contiene una lista di proiettili che viene aggiornata continuamente in base alle esigenze di proiettili diversi. Alcuni che subiscono la gravità, ognuno con un proprio danno e visuali diverse. Viene anche utilizzata (impropriamente) per creare esplosioni, grazie alla creazione di proiettili con danno nullo, che sono quindi solo visivi.

4. Classe player

La classe player è quella controllata dall'utente. Ha varie funzionalità come quella di cambiare arma, e ha un armatura che devia una percentuale dei danni subiti. Entrambi sono power up permanenti che possono essere comprati al mercato.

5. Classe shooter

La classe shooter è l'unica che utilizza la gravità dei proiettili. Utilizzando le leggi fisiche del moto parabolico viene calcolata la velocità iniziale dei proiettili in modo tale che con la loro traiettoria colpiscano precisamente il player

6. Classe kuba

La classe kuba è la sola che non spara proiettili, ma fa danno con contatto diretto, come un toro che carica su un bersaglio. Per rendere il gioco più interessante però questa entità non si muove verso il player ma si sposta senza una meta precisa, e si gira ogni volta che incontra un ostacolo.

7. **Classe platform**

   Nella classe platform viene implementata l'unità di base dell'ambiente di un livello. Le piattaforme sono usate per la realizzazione della base e delle pareti del livello oltre che come elementi su cui giocatore e nemici possono saltare.

8. **Classe menu**

   La classe menu implementa l'interfaccia da cui è possibile cambiare livello e accedere al mercato. Tramite il mercato è possibile acquistare potenziamenti come armatura e vita, ma anche cambiare arma utilizzata, ciò è possibile perché la classe menu è dotata di un puntatore al player tramite il quale può direttamente implementare le modifiche.

Implementazione di hitBox

In questo gioco abbiamo deciso di implementare delle sorta di sprite, invece di avere un'unica posizione, entity e piattaforme hanno una hitBox che rappresenta un rettangolo sullo schermo.
Effettivamente la hitBox non è altro che una coppia di coordinate A e B. Che indicano rispettivamente l'angolo in alto a sinistra, e l'angolo in basso a destra del rettangolo. Attraverso questa astrazione è possibile trattare ogni oggetto come se occupasse non una sola casella, ma un intero rettangolo di caselle. Rendendo il gioco molto più distintivo senza complicare eccessivamente il codice.