Brown

Gianlorenzo Urbano, Massimo Pignatti

February 2023

1 Introduzione

Brown è un gioco top-down sviluppato in c++ con la libreria ncurses. L'esecuzione avviene in un terminale, con la possibilità di visualizzare il gioco sia in modalità ascii che a colori.

2 Ripartizione del lavoro

Il lavoro è stato diviso come segue:

- Motore di gioco, generazione del mondo di gioco: Gianlorenzo Urbano;
- Logica di gioco: Massimo Pignatti.

3 Motore di gioco

3.1 ECS

Una delle componenti fondamentali di ogni game engine è sicuramente la gestione delle entità. La scelta è ricaduta su un ECS, ovvero un Entity Component System: più efficiente e versatile rispetto ad una semplice implementazione tramite lista di classi entità. A livello del game engine, un' entità non è nient'altro che un `uint32_t`, ovvero un `unsigned int`, mentre i componenti (e.g. la posizione, lo sprite, ecc) non sono attributi di una classe ma strutture indipendenti.

Per accedere ai dati si utilizzano `component_array` (uno per ogni categoria di componente): si utilizza l'id dell'entità come indice dell'array per scrivere e leggere informazioni riguardanti la singola entità; la classe `component_manager` si occupa di gestire l'array.

Un *sistema* invece è una classe che si occupa di gestire solo ed esclusivamente entità con la stessa `signature` del sistema stesso: questo permette, ad esempio, di costruire un sistema di rendering che stampa a schermo solo le entità che hanno un componente transform e una sprite. Infine, la classe `brain` si occupa di gestire il tutto all'unisono, gestendo i vari sistemi.

3.1.1 Native script

Uno dei componenti implementati è il native script, che permette di implementare logica di gioco più complessa, rendendo disponibile anche la comunicazione tra entità attraverso eventi.

3.1.2 Altri componenti

Altri componenti implementati sono:

- `Transform`: indica la posizione e la direzione attuale di un'entità;
- `Sprite`: la collezione di caratteri associata ad un'entità, ovvero quello che viene stampato a schermo;
- `Animator`: permette di cambiare dinamicamente la sprite in modo tale da creare varie animazioni;
- `UI`: testo che viene stampato a schermo; utile per visualizzare informazioni relative alle entità singole come il player.

3.2 Game stack

La classe virtuale `state` è la base di ogni stato di gioco. Fornisce l'interfaccia delle funzioni e delle variabili che devono essere necessariamente implementate da qualsiasi stato di gioco. Questo è necessario perché la classe `engine`, il motore di gioco in sè, ha uno stack di stati, dal quale ad ogni ciclo di gioco vengono eseguite alcune funzioni, come `update()`, `draw()` ed `handle_events()`.

3.3 Eventi

Il motore di gioco offre anche la possibilità di utilizzare un sistema di eventi implementato tramite `event listeners`, per evitare fenomeni come l'event polling. Dichiarato l'evento (che altro non è che un id, come per le entità), una classe può iscrivergli un suo metodo attraverso un `brain`, presente in ogni stato.

4 Game

4.1 Menù

Il menù iniziale del gioco fornisce al player una porta d'ingresso a un mondo di azione. Quando il giocatore entra nella porta, viene pushato un nuovo stato di gioco, in cui deve affrontare vari nemici e trovare la chiave per accedere alla stanza del boss. Allo scopo di aiutare il player a completare le missioni, sono state inserite pozioni e oggetti che aumentano le statistiche del personaggio.

4.2 Partita

Durante la partita, il player deve uccidere tutti i nemici, trovare la chiave per aprire la stanza del boss e ucciderlo. L'obiettivo del giocatore è quello di avanzare il più possibile nel gioco.

4.3 Player

Il player è gestito dalla classe `player_controller`. Gli spostamenti sono gestiti con i tasti WASD e premendo il tasto T è possibile sparare. L'inventario è gestito con il tasto I e gli oggetti possono essere raccolti premendo il tasto E. Ogni personaggio inizia la partita con 100 hp ma questi possono essere aumentati durante il gioco grazie a diversi potenziamenti.

4.4 Score

Il player inizia la partita con 0 di score e può ottenerne uccidendo i nemici presenti in ogni stanza.

4.5 Nemici

I nemici vengono gestiti dalla classe `ranged_enemy`. Questi nemici hanno la possibilità di sparare proiettili e di colpire i nemici. Possono muoversi in direzione del player e sparare solo se sono nel range del player. Le statistiche dei nemici, compreso il boss, vengono calcolate in base al livello del mondo raggiunto dal player. In questo modo si rende sempre più difficile avanzare e ottenere uno score più alto.

4.6 Artefatti e pozioni

Gli artifatti e le pozioni sono sottoclassi della classe "Item". Una volta raccolta possono essere utilizzate (pozioni) oppure tenute nell'inventario per un potenziamento passivo (artefatti). Ci sono vari artefatti:

- `Artefatto vita`: aumenta la vita del player;
- `Artefatto armatura`: aumenta leggermente l'armatura del player, riducendo il danno che potrebbe subire;
- `Artefatto danno`: aumenta il danno che il player infligge ai nemici.

4.7 Inventario

Gli oggetti possono essere aggiunti o rimossi dall'inventario con le funzioni `add_item` e `remove_item`. Viene renderizzato da `inventory_renderer`. L'inventario viene gestito come un array e si può utilizzare l'inventario aprendolo (tasto I) e premendo le freccette UP e DOWN per selezionare l'oggetto che si vuole. Si possono utilizzare gli oggetti(tasto E) o buttare via (tasto R). L'inventario ha

una capacità limitata a 10 tipi di oggetti (nota bene, oggetti con statistiche diverse occupano slot diversi) e quindi un'ulteriore sfida è cercare di raccogliere la giusta combinazione di artefatti e pozioni per sopravvivere più a lungo. Il player quindi si potrebbe trovare a scegliere tra aumentare il proprio danno a discapito, ad esempio, della salute. Una volta rimosso un artefatto dall'inventario i suoi effetti scompaiono.

4.8 Stanze

Le stanze sono generate a partire da tiles. I tiles sono dei pezzi di mappa di grandezza 5x5 che uniti tra di loro formano una vera e propria mappa. Tutte le stanze all'interno di un mondo sono effettivamente stati diversi, ognuno con le proprie entità ed eventi. Entrando in una nuova stanza quest'ultima viene aggiunta sul `game stack`, in modo tale da avere disponibile l'effetivo percorso che il player ha fatto all'interno del mondo.

4.9 Generazione del mondo

Per prima cosa, il gioco determina quante stanze deve avere il mondo, in base al livello raggiunto dal player: si parte da 7-8 stanze che aumentano di 2 o 3 ogni volta. Successivamente, vengono generate le stanze speciali, ovvero la stanza del boss e la stanza della chiave.

4.10 Boss

Il boss è il nemico finale di ogni livello, la quale sconfitta porta il player ad avanzare al mondo successivo. Ha un attacco speciale: lancia tre proiettili in una delle 4 direzioni addosso al player. Questi proiettili sono difficilmente schivabili, quindi bisogna prepararsi bene alla battaglia. Una volta morto e una volta uccisi tutti i nemici del piano, è possibile visitare il nuovo piano. Verrà pushato un nuovo stato di gioco corrispondente a questo piano.