RELAZIONE SUL PROGETTO 2022/2023 DEL CORSO DI PROGRAMMAZIONE

Il progetto è stato strutturato su diverse classi per permettere un maggiore ordine e lettura del codice. Esse sono le seguenti:

- "Game": è la classe principale del progetto che gestisce la parte del gioco vera e propria come le interazioni tra la mappa e il giocatore, il movimento dei nemici, i soldi, l'interfaccia grafica e il livello Mercato. Qui viene gestito anche il salvataggio.
- "Maps": si occupa di tutto ciò che riguarda la mappa come la sua generazione, la generazione dei nemici e dei soldi e la generazione delle strutture.
- "Board": si occupa di tutta la parte grafica che fa uso di comandi e funzioni provenienti dalla libreria Ncurses.
- "Player": gestisce tutto ciò che concerne il giocatore, come il movimento, la parte grafica e lo sparo.
- "Enemy": gestisce i vari tipi di nemici con il loro movimento, lo sparo per alcuni e la parte grafica.
- "Bullets": gestisce i proiettili sparati dal giocatore e dai nemici.
- "Segments": gestisce la costruzione dello scheletro della mappa.
- "Powerup": gestisce i potenziamenti.

Tramite "Game", tutte queste classi vengono inserite nel main di "Project" che semplicemente carica il gioco e lo fa partire. Esso termina una volta che il giocatore finisce tutte le vite

<u>Classi secondarie.</u>

Nella classe **"Board"** viene creata la struttura su cui si appoggia tutto il gioco. Semplicemente viene creata una cornice dove possiamo visualizzare il gioco. Fondamentale per il movimento del giocatore da tastiera e quello dei nemici da computer, è stata la libreria Ncurses che permette tramite il comando WTIMEOUT di gestire il movimento di più oggetti allo stesso tempo.

Nella classe **"Bullets"** viene creata la struttura di un proiettile e gestito il suo movimento. Ogni proiettile è un elemento di una lista dinamica caratterizzato dalle coordinate che avrà poi nella mappa, dalla direzione del movimento e da un codice che serve per identificare univocamente ogni proiettile (utile per esempio nella cancellazione). Nel giocatore e nei nemici che sparano, avere i proiettili come lista dinamica permette di gestire al meglio la rimozione (per cancellarli dalla mappa) e l'inserimento (per farli sparare) di un singolo proiettile. In questo modo si ha anche un'ottimizzazione della memoria dato il numero di proiettili che vengono sparati. I tipi di proiettile sono sostanzialmente due: classici che infliggono un singolo danno ai nemici ed esplosivi che uccidono il nemico all'istante. Il proiettile non viene inserito nella mappa fisicamente ma verrà stampato solo graficamente. Questo sarà maggiormente chiaro durante descrizione della classe "Game".

Nella classe **"Powerup"** viene creata la struttura di un potenziamento generico. Essi sono caratterizzati da un nome, una descrizione, un prezzo, una variabile per la difficoltà e la quantità che ha un significato diverso a seconda del potenziamento preso in esamine. In tutto i potenziamenti sono dodici e si dividono in tre tipologie:

1. Armi
   1.1. Pistola: spara un proiettile per volta
   1.2. Fucile: spara due proiettili per volta
   1.3. Mitragliatrice: spara tre proiettili per volta
   1.4. Doppia pistola: spara due proiettili per volta in direzioni opposte
2. Bonus
   2.1. Vita: ti da una vita in più
   2.2. Scudo: ti protegge da un singolo danno contro un nemico
   2.3. Ricarica di cento proiettili
   2.4. Ricarica di un proiettile esplosivo
   2.5. Jumping: ti aumenta il salto
3. Attivabili
   3.1. Armatura: per un tempo limitato (più pezzi dell'armatura hai, maggiore è il tempo di durata) ti dà l'invincibilità da nemici e da trappole presenti nel gioco
   3.2. Volo: puoi volare per un tempo limitato
   3.3. Teletrasporto: ti teletrasporta di un determinato numero di blocchi avanti o indietro in base al numero di teletrasporti che si hanno (più teletrasporti hai, più ti teletrasporta lontano).

Il prezzo di ogni potenziamento è stato deciso in base alla potenza e all'utilità che può avere nel gioco. Per esempio, Volo e Doppia Pistola che facilitano di gran lunga il gioco, costano molto di più che un semplice Scudo. L'importanza della variabile difficoltà dei singoli potenziamenti verrà poi spiegata nella classe "Maps".

Nella classe **"Segments"** viene creata la struttura di un singolo segmento ovvero una porzione di mappa. Esso si presenta come una matrice 25x25 di caratteri o semplicemente un vettore di stringhe. Vedremo in seguito che i segmenti sono la struttura portante della mappa, dato che sono le matrici che contengono pezzi di mappa. Essi, infatti, costituiranno i mattoncini per la creazione della mappa. Il significato dei segmenti sarà più chiaro nella descrizione della classe "Maps".

<u>Classi principali:</u>

Nella classe **"Maps"** vengono gestite numerose cose, quindi andiamo con ordine. Come accennato nella classe "Segments", una singola mappa è una lista dinamica di segmenti che accedono a dei file di testo che contengono parti (realizzate graficamente tramite dei caratteri) di mappa. Una mappa standard è costituita da dei segmenti di inizio e di fine che impediscono al giocatore di uscire dalla mappa, a dei segmenti vuoti che ne permettono una migliore visualizzazione, dai segmenti che costituiscono il mercato e infine dai segmenti che costituiscono la mappa in generale. Questi ultimi sono cinque più il numero della difficoltà e vengono scelti randomicamente da un numero estremamente alto di segmenti con pattern fissi (circa più di cento). Questa randomicità e questo ampio ventaglio di segmenti presenti, permettono praticamente un'infinita diversità nella creazione di nuove mappe. In linea generale i segmenti di mappa presentano tre strutture: le piattaforme, i muri e le spine. Tutte queste sono considerate strutture solide dal giocatore e dai nemici e le spine tolgono vita al giocatore. Successivamente vengono create le monete. Il numero di monete generate è fisso a cinque per cercare di essere ben bilanciato con i prezzi del mercato. Per esempio, così facendo

il giocatore non può “permettersi” potenziamenti se non ha giocato almeno due livelli e preso tutte le monete. Infine, vengono generati i nemici. Parleremo nel dettaglio dei nemici nella classe “Enemy”, qui ci limiteremo a presentarne la generazione. Ogni volta che il giocatore raggiunge il mercato e compra dei potenziamenti, viene aggiornata la difficoltà del gioco in base ai potenziamenti comprati. I nemici hanno una differente difficoltà basata sostanzialmente sul tipo di movimento che fanno e sul numero di vite che hanno. La generazione tiene conto di questa difficoltà, la quale viene usata per calcolare il numero massimo di ogni tipo di nemico che può essere generato. Per esempio, fino a che la difficoltà è sotto il 3 vengono generati i primi nemici che sono più semplici da sconfiggere. Fino ad una difficoltà minore di 5 non vengono creati nemici che sparano e così via. Ovviamente c’è anche qui una componente di randomicità per rendere il gioco nuovo ogni volta ma sempre con una difficoltà crescente. Si è scelto di non permettere una diminuzione della difficoltà; nel caso non si dovesse comprare nulla, la difficoltà rimarrebbe inalterata. Nel gioco i nemici sono gestiti da liste dinamiche che permettono una facile gestione nella generazione (con l’inserimento nella lista) e nella morte (con la rimozione dalla lista) di un nemico. Sia nella generazione dei soldi che dei nemici viene tenuto conto della mappa. Nessuno di essi può nascere in qualche posto irraggiungibile dal giocatore o in posti che presentano qualche struttura della mappa.
Un’ultima cosa di cui si occupa “Maps” è la gestione dell’interazione tra giocatore e mappa. Come già accennato sopra il giocatore considera solido ogni elemento della mappa che presenta un simbolo tra “#”, “|” , ”-“ , ”^”, che rappresentano le strutture fisiche. Di questa interazione ne gestisce “Maps” con delle funzioni che vengono richiamate in “Game” durante il movimento del giocatore. Tutte le strutture sono fisse e non rimovibili.

Prima di introdurre la classe “Game” bisogna descrivere brevemente le classi “Player” e “Enemy”.

Nella classe **“Player”** viene introdotto il giocatore. Esso, tramite input da tastiera, può fare varie cose. I movimenti che può fare sono andare avanti e indietro e saltare in entrambe le direzioni. Il salto è stato implementato come un movimento continuo per facilitare la gestione della collisione con gli oggetti della mappa e di dare la possibilità di sparare durante il salto. Si è deciso di assegnare al giocatore anche i potenziamenti in modo tale che fosse più facile gestire l’acquisto nel livello Mercato. Semplicemente il giocatore presenta già tutti i potenziamenti ma con quantità fissata a 0. Una volta comprato qualcosa questa variabile aumenta o se si usa qualcosa diminuisce. Sempre qui sono gestite le tempistiche di attivazione e durata dell’armatura, del teletrasporto e del volo. Tutti questi potenziamenti sono gestiti come input da tastiera. Il giocatore, come già accennato prima, può anche sparare. Per ogni nemico ucciso, esso riceverà dei punti che potrà collezionare durante il gioco. Questo fattore punteggio aumenta da un lato la voglia di giocare perché ci si può porre degli obbiettivi come record di punteggio e simili, dall’altro incentiva il fermarsi ad uccidere i nemici e non proseguire cercando solo di evitarli. I proiettili e l’arma sono dei potenziamenti che il giocatore può acquistare. Così facendo, il giocatore per i primi livelli dovrà sperimentare il gioco senza poter uccidere i nemici (se presenti), per affinare il movimento, il salto e le insidie presenti nel gioco.

Nella classe **“Enemy”** vengono creati i nemici. I tipi di nemici sono in tutto dieci e si differenziano per il movimento che fanno, le vite che hanno e se sparano o no. Essendo tutti i

nemici caratterizzati da elementi comuni come il movimento e l'interfaccia grafica, si è fatto uso dell'ereditarietà. Tutti i nemici discendono dal nemico 0 e i nemici che sparano discendono tutti dal nemico 6. In questo modo si è evitato il dover ricreare tutto da zero ogni volta. La difficoltà dei nemici è crescente. Per esempio, i primi nemici hanno un movimento fisso come può essere andare avanti e indietro o giù e su, poi si passa ai nemici con un movimento più complicato tipo randomico o addirittura un movimento intelligente che permette al nemico di seguire il giocatore ovunque vada. La seconda metà dei nemici invece presentano la possibilità di sparare. Hanno tutti una tipologia di sparo diversa, chi spara solo se vede il giocatore avvicinarsi, chi spara invece sempre in una direzione o chi spara in più direzioni. Si noti che le vite dei nemici hanno un ruolo nella difficoltà del gioco, essa è proporzionata alla forza del nemico. Più i nemici sono facili, più anche la loro vita è bassa, mentre i nemici che sparano presentano un numero di vite più alto per aumentare maggiormente la difficoltà nell'ucciderli. Questa diversità nei nemici rende sicuramente il gioco più divertente ma anche più complicato per l'imprevedibilità dei movimenti. Per evitare che il gioco si facesse troppo complicato abbiamo deciso di implementare i nemici con dei simboli riconoscibili. Ogni nemico che non spara ha un numero che lo rappresenta. Più aumenta il numero, più il nemico diventa forte. Per quelli che sparano, si ha un numero che corrisponde al numero del nemico di cui hanno ereditato il movimento mentre per coloro che hanno un movimento nuovo vale la stessa regola di quelli che non sparano.

Passiamo infine alla classe **"Game"** che è la classe che riunisce tutte le classi descritte in precedenza. Qui viene gestito il cuore del gioco. Tutto ruota attorno a due funzioni principali: il costruttore e "UpdateState()". Partiamo dalla gestione delle mappe. Anche qui si è fatto uso di una gestione dinamica delle mappe. Esse sono implementate come lista dinamica in modo tale che fosse facile scorrere le varie mappe (ricordiamo che il giocatore può fare avanti e indietro tra le varie mappe) e generarne delle nuove all'occasione. Per ottenere lo scorrimento della mappa, la questione cruciale è il fatto che non è il giocatore a muoversi avanti e indietro ma la visualizzazione della mappa. Come se ci fosse una telecamera che in base agli input del giocatore si spostasse avanti e indietro. Quindi il movimento del giocatore effettivo è fatto solo in verticale. Questo tipo di movimento ci ha portato a lavorare sue due livelli: il livello del giocatore e il livello della mappa. Anche per assicurarci un salvataggio delle mappe precedenti e di tutti gli oggetti precedenti, i nemici e le monete sono parte della mappa e si muovono con essa (per i nemici c'è anche il loro movimento) mentre il giocatore è su un altro livello di lavoro. Come già accennato la generazione di una nuova mappa dipende dalla difficoltà che dipende a sua volta dai potenziamenti comprati. Più potenziamenti si comprano, più la difficoltà del gioco aumenta in modo proporzionale. Per non complicare troppo il gioco, si è deciso che alcuni potenziamenti hanno un peso minore nella difficoltà del gioco. Per esempio se si comprano più di una volta i proiettili, la difficoltà non aumenta, o anche se si compra una vita in più, la difficoltà non aumenta. Per impedire un'invincibilità troppo alta del giocatore si è posto un massimo di volte per comperare alcuni potenziamenti, ad esempio l'armatura o il jumping.

Per quanto riguarda il livello Mercato si è deciso di implementarlo come parte di ogni mappa (vedi "Maps") in modo tale che fosse facile gestire il passare da una mappa all'altra, ricordando che il giocatore può andare avanti e indietro. Ovviamente il giocatore non può passare due volte dallo stesso mercato, anche perché sennò il gioco sarebbe diventato troppo facile. Esso

presenta tre potenziamenti generati randomicamente da tutti i potenziamenti presenti nel gioco. Uno di ogni tipo diverso. Così facendo il giocatore non ha sempre accesso a tutti i potenziamenti ma solamente a quelli usciti. Per rendere il gioco un po' più complicato, il giocatore non può comprare più volte un potenziamento. Per facilitare la conoscenza dei potenziamenti al giocatore, nel mercato è stata inserita una parte di interfaccia grafica che spiega al giocatore che tipo di potenziamento ha di fronte.

Per la parte del salvataggio si è deciso di utilizzare l'accesso a file di testo in modo tale che in quel file fossero salvate tutte le informazioni che necessitavamo di salvare, come per esempio vite, soldi, punti e "Powerup" del giocatore. Per non appesantire l'esecuzione del codice, abbiamo deciso di effettuare il salvataggio solo all'uscita dal mercato o nel momento in cui si perde una vita. Questo comporta che se durante un livello utilizziamo un consumabile e poi chiudiamo il terminale, alla riapertura di esso ci ritroveremo all'inizio della mappa con il consumabile ancora nel nostro inventario.

Infine, abbiamo anche aggiunto una piccola interfaccia grafica che facilita l'utente a visualizzare la propria vita, i propri soldi e i "Powerup" comprati.