ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA
corso INFORMATICA (8009)

RELAZIONE PROGETTO DI PROGRAMMAZIONE
ATdot cpp
.s. 2022/2024

A cura di:
Gianluca Sperti 0001078941
Tiziano Gregori 0001089069
Novi Lorenzo 0001090067

ATdot cpp è un videogioco platform in grafica ASCII, sviluppato a più livelli con un punteggio e con il protagonista controllato tramite tastiera.
È possibile avanzare solo dopo aver sconfitto tutti i nemici nel livello e aver raccolto le monete.
Tramite i punti acquisiti è possibile acquistare potenziamenti nello shop.
È possibile salvare la partita e riprendere da essa in qualsiasi momento.
Bordi, scritte e difficoltà sono modificabili dal menù impostazioni.

Premesse:
L'intero progetto è stato sviluppato in ambiente Linux e come editor di testo è stato utilizzato Visual Studio Code. Le librerie standard utilizzate sono :

- iostream
- ncurses
- fstream
- stdlib.h
- ctime
- stdio.h
- cstring

Il progetto inoltre è costituito da altre diciassette librerie con relativo file .cpp.
Tali librerie sono state denominate : mercante.hpp , Platformgame.hpp , Protagonist.hpp, shop.hpp, Board.hpp , Heal.hpp, Bullet.hpp , Coin.hpp , Drawable.hpp Enemy.hpp , Game.hpp, HUD.hpp , matrice.hpp, Market_menu.hpp, menu_creator.hpp, menu_creator_options.hpp, selection_menu.hpp.
Spiegheremo più avanti lo scopo di tali librerie.

Svolgimento

Fase 1- Installazione libreria ncurses e suddivisione dei compiti.
Lavorando tutti in ambiente Linux l'installazione della libreria è risultata particolarmente semplice è bastato aprire il terminale e digitare il comando:
sudo apt-get install libncurses5-dev libncursesw5-dev
Dopo esserci assicurati che la libreria funzionasse correttamente siamo passati alla divisione del lavoro all'interno dei membri del gruppo.

 # Gianluca Sperti: ha implementato il protagonista, i nemici e i vari oggetti raccoglibili, armi HUD e la classe Platformgame che si occupa di aggiornare lo stato di gioco e gestire tutte le varie liste di oggetti.

#Tiziano Gregori: ha implementato tutti i menù e il menu-shop implementazione impostazioni grafiche e salvataggi,potenziamenti e gestione dello scorrimento dei livelli

#Lorenzo Novi: ha implementato le mappe e salvataggi

Fase 2-Implementazione codice
Non riuscendo ad incontrarci per programmare tutti insieme ogni componente (almeno inizialmente) ha svolto la propria parte di codice per conto suo. Di seguito ogni studente riporta la descrizione e il funzionamento del codice da lui implementato:

#Gianluca Sperti: La classe Drawable ha lo scopo di creare un generico oggetto che sarà poi stampato nella mappa, con attributi di coordinate e chtype rappresentante l'oggetto.
Tutti gli oggetti stampati su schermo sono sue sottoclassi: Enemy, Coin, Heal, Bullet e Protagonist.
La classe Protagonist contiene invece tutti gli attributi associati alle sue statistiche e metodi per incrementare grazie ai potenziamenti, contiene inoltre i metodi per far muovere il protagonista, farlo saltare, cadere (gravità).
La classe Enemy contiene un attributo che contraddistingue i due tipi di nemici, un brawler e uno shooter, che avranno stats assegnate in base alla difficoltà in gioco.
La classe Platformgame si occupa della creazione delle varie liste di nemici nei livelli, degli oggetti come Coin, Heal e Bullet e della modifica in base alle varie dinamiche di gioco.
Ogni lista di oggetti ha una propria push, pop e stampa.
Gli oggetti Enemy e Bullet qui hanno un metodo move che li fa muovere nella mappa in base alle caratteristiche di ogni oggetto.
Ad esempio gli oggetti Bullet andranno nella direzione in cui il protagonista ha sparato e i nemici si muoveranno in una direzione e la cambieranno in caso incontrassero un ostacolo.
È presente un metodo Falling che gestisce gli eventi in caso di caduta del protagonista, ad esempio raccogliere un oggetto in caduta, uccidere un nemico cadendoci sopra.
Ogni volta che si raccoglie un oggetto o si uccide un nemico verranno aggiornate le statistiche ed eliminati gli oggetti nelle liste.

Il metodo UpdateState viene richiamato ogni tic:

- verifica che il protagonista abbia abbastanza vita per proseguire, se sia in una posizione di cambio mappa

- aggiorna lo stato di gioco, fa muovere nemici, proiettili, gestisce salto e gravità

Il metodo ProcessInput invece riceve ed elabora tutti gli input da tastiera.

Per quanto riguarda le Armi è stata implementata la classe Bullet, usata sia da dai nemici shooters che dal protagonista (gli oggetti bullet pushati dal protagonista avranno un attributo type diverso, in modo che i nemici non si facciano danno da soli attaccandosi).
Il danno del bullet varia in base alla difficoltà (per i nemici) e in base ai potenziamenti (per il protagonista).
I Bullet verranno poppati nel caso incontrassero un ostacolo o raggiungessero il range massimo di sparo.

#Tiziano Gregori: la classe “Menu_creator” ha lo scopo di creare un generico menù, a fin di ciò, al suo interno sono contenuti metodi per la stampa, calcolo della posizione delle scritte, modifiche estetiche e di navigazione del menù in questione. La classe “Option_menu” eredita da quella precedente, si occupa di creare un menu di opzioni: versione più specifica del menù sopracitato con maggiori differenze dal punto di vista estetico e tecnico. La tipologia di metodi contenuti è la stessa. La classe “Market_menu” è una variante, non troppo dissimile, del menù di opzioni perciò eredita dalla sua classe.
Infine tutti i tipi di menù sono controllati dalla classe “Menu_selector”: un gestore di liste che si occupa dell’interazione tra un menù e l'altro, al suo interno sono contenuti metodi per passare da menù in menù, tornare indietro e metodi get per ottenere eventuali modifiche dell’utente.
La classe “Game” si occupa dello scorrimento ed interazione tra livelli: si tratta di un gestore di liste che permette all’utente di andare avanti ed indietro fra livelli, aprire i diversi menù disponibili in gioco, salvare la partita attuale, caricare salvataggi, salvare le statistiche del personaggio fra un livello e mettere in comunicazione le diverse classi usate per costruire il gioco.

#Lorenzo Novi:La libreria che si occupa della creazione della mappa di gioco si chiama matrice. Nel codice definisco una classe GameMap che gestisce una mappa di gioco rappresentata da una matrice bidimensionale di interi. La mappa viene inizializzata con una certa dimensione specificata dall'utente. La funzione createMap() viene utilizzata per generare una matrice di interi compresi tra 0-1 che vengono generati in maniera random grazie alla libreria ctime , poi ordinati in modo tale da creare delle sequenze di 1 (che verranno tradotte come piattaforme) che consentano al player di raggiungere qualsiasi punto della mappa. Inoltre vengono aggiunti dei 2 in maniera random che stanno a simboleggiare le trappole. La funzione La funzione translateMap è responsabile della traduzione dei valori della matrice di gioco in caratteri da visualizzare nell'interfaccia utente, mentre la funzione printMap() viene utilizzata per stampare la mappa su una finestra ncurses.
La libreria shop si occupa invece della creazione della mappa shop funziona esattamente come la libreria delle mappe solo che gli 1 e gli 0 della matrice numerica vengono assegnati in maniera fissa.
Le funzioni per i salvataggi /caricamenti sono implementate all'interno della classe Game e funzionano tramite scrittura e lettura di due file .txt uno per le mappe e uno per le statistiche del player.

Fase 3-Unione codice
Durante la fase di composizione delle varie parti grazie a varie chiamate tramite discord e collaborazione da parte di tutti i componenti , siamo riusciti a unire i vari pezzi di codice. Dovendo quindi adattare le proprie parti di codice con quelle degli altri, ognuno poi ha potuto modificare parti di codice degli altri.
Mail d’istituto:

- tiziano.gregori@studio.unibo.it
- gianluca.sperti@studio.unibo.it
- lorenzo.novi@studio.unibo.it