Corso di Programmazione

Progetto assegnato nel corso dell'anno accademico 2019/2020.

Partecipanti

- Colamonaco Stefano (codice di matricola: 0000915083)
- Di Pasquale Alessio (codice di matricola: 0000916060)

Consegna

Il tema del progetto è la creazione di un gioco di corse automobilistiche, sviluppato su più livelli generati dinamicamente e senza traguardi. Il giocatore è rappresentato da un automobile che deve muoversi lungo la strada cercando di raccogliere i bonus forniti, evitando altre auto ed eventuali ostacoli per ottenere il punteggio più alto possibile.

Il tutto deve essere creato in C++ utilizzando un approccio OOP e senza l'ausilio di librerie grafiche.

Relazione

Il progetto è stato completato tra il mese di agosto ed i primi giorni di settembre utilizzando il servizio di hosting per servizi software GitHub, in modo da favorire il lavoro di gruppo a distanza dovuto all'emergenza Covid-19. La maggior parte delle sessioni di programmazione si sono svolte con entrambi i partecipanti presenti con l'ausilio di Skype.

Il tutto si compone di 14 files, tra .cpp, .hpp e un makefile per agevolare la compilazione. Le classi che compongono il progetto sono:

- GameManager: è la classe principale, che si occupa del funzionamento vero e proprio del gioco.
- LevelManager: Si occupa della gestione del livello corrente, del passaggio al precedente o al successivo, e della creazione dei nuovi livelli raggiunti.

- Map: Questa classe serve ad avere un collegamento diretto tra il livello e gli oggetti, in quanto ad ogni livello è associata una mappa costituita dai bonus/malus incontrati e le relative coordinate.
- Collectible: E' la classe padre di tutti gli oggetti 'raccoglibili' che possono essere incontrati. E' stata utile a definire attributi e metodi comuni a tutti i bonus/malus.
- Collectibles: All'interno di questi files non è definita un'unica classe, bensì tre, tutte derivanti dalla classe padre Collectible e che si occupano di instanziare i diversi tipi di oggetti incontrabili.

Ogni classe possiede i propri files per l'intestazione e la definizione di attributi e metodi, inoltre sono stati inclusi due header aggiuntivi: includes.hpp e Constants.hpp, rispettivamente per includere le librerie necessarie e le variabili a valore costante.

Per avviare il programma è sufficiente digitare da linea di comando 'make run' nella cartella contenente il makefile, che si occuperà di compilare i files necessari e di avviare l'eseguibile 'CrazyRun' generato.

Il menù principale si compone di tre scelte: iniziare una nuova partita, accedere alla schermata contenente i crediti agli autori e terminare il programma. Scegliendo la prima opzione si passa alla schermata di scelta della skin (l'implementazione corrente prevede tre skin differenti). Una volta effettuata la scelta della skin ha inizio la nuova partita.

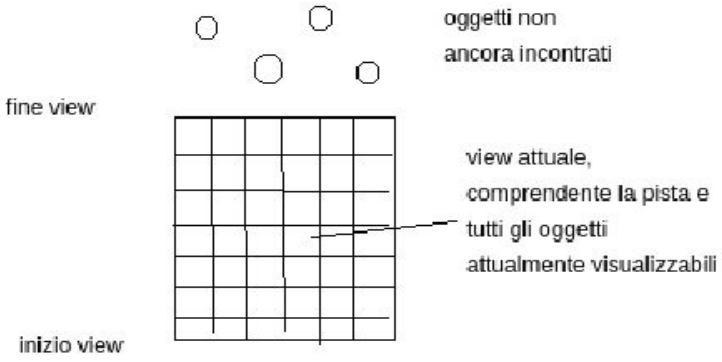
Il metodo 'prepare', contiene l'inizializzazione delle impostazioni relative alla libreria 'ncurses', utilizzata per garantire all'utente di poter inserire molteplici comandi e senza l'ausilio del tasto invio attraverso la funzione getch(). Inoltre per evitare problemi legati all'ottimizzazione, alcune funzioni di questa libreria quali printw() e addch() sono state utilizzate per stampare la griglia di gioco.

Una volta terminati i preparativi, viene chiamato il metodo start, ovvero quello che contiene i loop che definiscono il gioco. Nel loop più esterno troviamo le inizializzazioni che permettono di passare da un livello all'altro, ad esempio reperire la mappa associata al livello corrente e la quantità di oggetti che sono presenti in ogni view (una view è l'intervallo entro cui vengono visualizzati nuovi oggetti sulla pista. La view in un determinato momento mostra tutti gli oggetti con distanza dall'inizio del livello compresa tra la distanza della view rispetto all'inizio e i successivi MAPHEIGHT passi avanti).

Nel loop più interno invece ha sede la gestione del singolo livello. In particolare vengono eseguite ripetutamente le seguenti operazioni:

- incremento del punteggio di 5 punti
- aggiornamento del best score
- incremento della posizione della view
- controllo di eventuali collisioni del player con oggetti o bordi
- controllo dedicato ad un eventuale incremento o decremento di livello
- controllo riguardante la generazione di eventuali oggetti nell'intervallo view successivo
- viene stampata la view corrente compresa di player, oggetti, bordi, punteggio e numero del livello corrente
- viene attivato un delay generato dinamicamente rispetto al livello
- controllo di eventuali comandi da parte dell'utente.

La parte più interessante del progetto, e forse anche la più complicata a livello logico è quella che riguarda la generazione dei bonus/malus all'interno dei livelli.
Possiamo immaginare la view come una matrice in continuo scorrimento che funziona in questo modo:



Per limitare al massimo lo spazio occupato dalle mappe, gli oggetti prima di essere visualizzati non sono contenuti in una matrice (che altrimenti lascerebbe molti spazi inutilizzati), bensì in liste ordinate che vengono scansionate ed ampliate quando necessario. Ogni oggetto della classe Map contiene infatti tre liste, una per tipologia di raccoglibili: rampe, ostacoli e automobili. Quando arriva il momento per un oggetto di essere visualizzato, questo viene posizionato in due matrici:
una composta solo da caratteri e utilizzata per la stampa della view corrente.
Ed un'altra composta da puntatori a Collectible, per favorire la ricerca delle informazioni di un determinato oggetto quando questo compare in una collisione con il player.

La generazione di nuovi oggetti è gestita per distanza rispetto alla view corrente. Infatti quando stiamo visualizzando una view, in realtà è già stata generata quella successiva.

Questo tipo di approccio ha facilitato molto il passaggio da un livello al successivo o precedente, in quanto basta ripartire dai primi elementi delle liste associate alla mappa del livello corrente, fino al momento di generare nuovi oggetti. Tale momento è salvato in un attributo della classe map chiamato 'lastConsideredZone', esso non è altro che un intero contenente la massima distanza entro cui sono già stati generati oggetti.

Al termine del progetto sono stati eseguiti molteplici test a livelli piuttosto elevati senza riscontrare problemi. Mentre nei livelli più bassi, potrebbe verificarsi un leggero ritardo nella visualizzazione della parte inferiore della mappa, dovuto al fatto che la funzione getch() della libreria ncurses esegue il refresh dello schermo automaticamente.