Progetto

esame di Programmazione 2024/2025

Introduzione

Il progetto è stato svolto in coppia dagli studenti di informatica Esposito Francesco (terzo anno) e Belbusti Luca (primo anno), il lavoro in questione prevedeva la realizzazione del celebre gioco Snake tramite il linguaggio di programmazione object oriented C++ e la libreria grafica ncurses. Le specifiche da rispettare richiedevano la presenza di una classifica con punteggi ordinati in ordine descrescente accessibile tramite un menù di gioco, il suddetto menu permette anche di iniziare una nuova partita.

Il gioco non si sviluppa su mappe diverse ma su livelli, ciascuno con una velocità di movimento differente e proporzionale alla difficoltà del livello stesso. Un timer determina la fine della partita, a quel punto il giocatore si vedrà assegnato un punteggio che tiene conto del numero di mele mangiate e anche del livello di difficoltà scelto, premiando con un moltiplicatore i players che avranno optato per un livello più arduo.

Ci sono altre vie che possono portare il giocatore al game over, ad esempio se il serpente collide con se stesso o se il giocatore decide di uscire volontariamente dal gioco premendo un tasto a questo designato ( q nel nostro caso ).

La principale differenza dal gioco originale è l’assenza della feature che per ogni mela mangiata dal serpente aggiunge al suo corpo un segmento, rendendo quindi più complicato muoversi col procedere della partita.

Divisione compiti

*Belbusti Luca:*

* Sistema di movimento
* Reticolo di gioco e grafica livelli
* Spawn e dinamiche della mela
* Timer
* Menu

*Esposito Francesco:*

* Salvataggio e lettura su e da file
* Punteggio ( Bonus, classifica, highscore )
* Sistema di livelli e difficoltà
* Menu

Implementazione delle specifiche

La parte di salvataggio dei progressi viene svolta salvando su due file testuali diversi la classifica e il punteggio più alto ottenuto fino a quel momento, i file sono rispettivamente classifica.txt e punteggio.txt . Da questi file sono letti quindi i punteggi qualora il giocatore decidesse di leggere la classifica o giocando desideri visualizzare qual è il punteggio più alto mai registrato; la classifica viene ordinata tramite un bubble sort e permette all’utente di vedere al termine di quale partita è stato registrato un certo punteggio ed anche il livello giocato fino ad un massimo di 100 partite. Se il codice non rileva punteggi salvati informa il giocatore tramite testo.

Quando il giocatore termina di consultare la classifica può tornare al menu principale premendo un qualsiasi tasto.

La funzione showClassifica() nel file menu.cpp si occupa di leggere i punteggi da file e ordinarli prima di renderli disponibili per la lettura da parte dell’utente dal menu di pausa.

La funzione Classifica() invece situata nel file SnakeGame.cpp salva i dati nel file classifica.txt

È da menzionare anche la funzione ScoreMultiplier che moltiplica il numero di mele mangiate per il numero corrispondente al livello giocato.

CLASSE SNAKE

La classe Snake gestisce il movimento del serpente, eventuali cambi di direzione, le collisioni con sé stesso, e il disegno dello stesso sulla griglia di gioco. Per fare ciò tra i campi della classe troviamo la variabile “length” che tiene traccia della lunghezza predefinita del serpente; “tail” e “head” che sono strutture del tipo “coordinates” in modo tale da tenere sempre traccia delle coordinate di testa e coda del serpente; “position” ovvero un array con le coordinate delle parti del serpente(all’indice 0 dell’array è contenuta la posizione della coda e, a scalare, in ultima posizione quella della testa); e infine “direction” che contiene una direzione tra quelle dell enum DIRECTION (U,D,L,R).

Il costruttore della classe crea una matrice della grandezza del campo da gioco per mappare il serpente e successivamente imposta a true le caselle in cui si trova inizialmente (in posizione centrata allo schermo) e salva le posizioni di testa e coda che serviranno per il movimento.

Il metodo Move restituisce un booleano per controllare che quando il serpente si muove, non si scontri con sé stesso (restituisce false in questo caso, true altrimenti): all’interno del metodo viene creata una nuova testa in base alla direzione in cui si muove il serpente, con la funzione CheckSelfCollision() si controlla che la nuova testa non faccia già parte del corpo, in caso contrario viene eliminata la coda del serpente mettendo a false quella casella, vengono fatti scalare di uno tutti i componenti dell’array position in modo da avere le coordinate della nuova coda all indice 0 e viene inserita in ultima posizione dell’array la posizione della testa. In questo modo il movimento del serpente viene realizzato togliendo un pezzo dalla coda e aggiungendolo in testa.

Andando avanti il metodo ChangeDirection(int key) riceve come input l’input dell’utente e cambia la direzione vietando i cambi sullo stesso asse.

Infine, oltre ai metodi get, troviamo il metodo Draw() che semplicemente passa l’intera matrice e disegna con il carattere “0” le casella segnate con true, in cui quindi è presente il corpo del serpente, e con il carattere “X” la testa del serpente.

CLASSE APPLE

La classe Apple disegna in punti casuali le mele che il serpente deve mangiare e, quando ciò avviene, rigenera un’altra mela in un altro punto casuale.

Tra i metodi è da notare NewApplePosition() che viene chiamato dal ciclo di gioco quando il serpente mangia una mela: viene quindi generata un'altra mela in posizione casuale facendo però attenzione che la coordinata non sia occupata dal corpo del serpente.

CLASSE BOARD

La classe Board si occupa di generare il campo di gioco e di stampare a schermo il timer, lo score attuale, l’highestscore e il livello in cui ci si trova.

***SnakeGame::run()***

***Il ciclo di gioco (run) gestisce l’esecuzione continua del gioco Snake, aggiornando lo stato del serpente, l’input del giocatore e la visualizzazione a ogni iterazione.***

***All'avvio, vengono inizializzati:***

* ***lo stato di gioco (gameon),***
* ***il punteggio,***
* ***la posizione iniziale del serpente,***
* ***e una posizione casuale per la mela.***

***Il ciclo principale (while (gameon)) esegue costantemente tre attività:***

1. ***Gestione dell'input utente (non bloccante):  
   L'input da tastiera viene letto con getch() in modalità asincrona. Se il giocatore preme una freccia direzionale, il codice verifica che non sia una direzione opposta rispetto a quella corrente, per evitare che il serpente si giri su sé stesso. Se valida, la direzione viene salvata come pendingDirection. Sono gestiti anche i comandi di pausa (spazio) e uscita (q).***
2. ***Aggiornamento dello stato del gioco:  
   Il serpente viene mosso solo ogni TICKS\_PER\_MOVE cicli per mantenere una velocità costante. Quando scatta un "tick di movimento":***
   * ***la direzione corrente viene aggiornata,***
   * ***il serpente si muove in quella direzione,***
   * ***viene verificata la collisione con la mela,***
   * ***e controllato il tempo residuo.***
3. ***Rendering e visualizzazione:  
   A ogni aggiornamento valido, vengono ridisegnati lo sfondo, i bordi, il serpente, la mela, il punteggio corrente e il record.***

***Infine, viene usato napms(10) per regolare il tempo tra ogni tick (ogni iterazione del ciclo), garantendo fluidità e stabilità nella velocità di gioco.***