14 settembre 2020

Relazione progetto

Leonardo Giacomini, Mario Miccichè, Andrea Schinoppi, Niccolò Sghinolfi

---

Menù

Al momento dell'avvio dell'applicativo, in console compare il menù principale che presenta le tre opzioni "START GAME", "INFO" ed "EXIT". Per navigare all'interno del menù, l'utente dovrà utilizzare i tasti 'S' e 'W' e premere invio per confermare.

La classe `Menù` contiene 5 funzioni:

Il costruttore prende in input un intero, inizializzato di default a 9, che va a modificare in blu il colore del testo selezionato all'interno del menù. Inoltre inizializza i campi `tasto` e `sel` della classe rispettivamente a 0 e 1 .

La funzione `void select(int mat[20][30])` è un loop che viene interrotto nel momento in cui il giocatore preme invio, rendendo `tasto` == 13.
Stampa a video, sfruttando la matrice presa in input, il menù principale e permette all'utente di scegliere una delle tre opzioni citate in precedenza facendo aumentare e diminuire `sel` grazie a `_getch()`, presente nella libreria <conio.h>, colorando di blu la voce in quel momento selezionata e di bianco (codice 15) le altre.

La funzione `void info(int mat[20][30])`, chiamata da select() nel momento in cui viene selezionata la voce "INFO", stampa a video le istruzioni per giocare.

La funzione `void exit()`, chiamata da select() nel momento in cui viene selezionata la voce "EXIT", stampa a video una ASCII art riportante la scritta "GOOD BYE" e termina il programma.

La funzione `void load_game(int mat[20][30])`, chiamata da select() nel momento in cui viene selezionata la voce "START GAME", stampa a video una ASCII ART riportante la scritta "LOADING", una barra di caricamento e un'auto che si muove lungo lo schermo. Quando la barra di caricamento raggiunge il 100%, viene chiamata la funzione `in_game()` che fa iniziare il gioco vero e proprio.

La funzione `void pausa(int mat[20] [30])`, viene chiamata dal giocatore durante la partita facendo apparire a schermo un piccolo menù di pausa deve poter scegliere se continuare la partita o ritornare al menù principale del gioco, nel mentre il gioco rimane fermo al punto in cui viene premuto il tasto "P".

La funzione `void game_over(int max, int mat[20] [30])`, viene invocata nel momento in cui il punteggio della partita o il punteggio del primo livello scende sotto lo zero, la funzione fa terminare la partita, mostra a schermo il punteggio massimo e fa apparire un menù che permette di iniziare una nuova partita o di ritornare al menù principale.

Strada e macchina

Non appena il gioco ha inizio vengono stampati a schermo la macchina e i bordi della strada. L'auto è formata da una 'X' e da quattro 'ø' che fungono rispettivamente da carrozzeria e da ruote, mentre la strada è delimitata da parentesi quadre aperte e chiuse. Da questo momento in poi l'utente può muovere la macchina con i tasti 'A' e 'D' per schivare gli ostacoli che scendono dall'alto.

Le due classi `Strada` e `Macchina` hanno due costruttori molto simili perché entrambi inizializzano le matrici di interi che prendono in input assegnando a tutti gli elementi il valore 0.

La classe `Strada`, oltre al costruttore, contiene 2 funzioni:

La funzione `void draw_map(int mat_strada[20][30])` stampa a video diversi caratteri a seconda del valore assegnato alla matrice nella posizione considerata.
La scelta di assegnare ad ogni elemento sullo schermo un "ID" si è rivelata comoda per implementare diversi tipi di oggetti poiché è stato possibile gestirli con l'ausilio di una sola funzione.

La funzione `void reset_map(int mat_strada[20][30])` pone tutti gli elementi della matrice a 0 tranne quelli posti agli estremi laterali a cui sono invece assegnati rispettivamente i valori 8 e 9 che rappresentano i bordi della strada.

La classe `Macchina` contiene 6 funzioni:

Il costruttore, oltre ad inizializzare la matrice, assegna i valori 15 e 18 alle posizioni sull'asse verticale e orizzontale della macchina per far si che all'avvio del gioco si trovi al centro della strada.

La funzione `void set_car(int mat_m[20][30])` assegna agli elementi della matrice nella posizione in cui si trova la macchina i valori 1 e 2, rappresentanti la carrozzeria e le quattro ruote.

La funzione `int hit_box(int mat_m[20][30])` verifica se l'auto si è scontrata con un ostacolo e ritorna un valore diverso in base alla posizione in cui è avvenuta la collisione: 1 se davanti, 2 se al centro, 3 se dietro e 4 se colpisce il bordo della strada. Questi valori saranno utilizzati da un'altra funzione per la gestione degli ostacoli.

La funzione `void move()` permette di ricevere l'input da tastiera e di muovere la macchina grazie alla funzione `GetAsyncKeyState()`, presente nella libreria <windows.h>.

Le funzioni `int get_x()` e `void edit_x(int x)` servono rispettivamente ad ottenere il valore corrente di x, ovvero la posizione dell'auto sull'asse orizzontale, e a modificarne manualmente il valore.

Livelli e ostacoli

Ogni livello viene gestito tramite una lista bidirezionale ove ogni nodo corrisponde ad un livello al cui interno si trova un puntatore ad un'altra lista che gestisce gli ostacoli. Ogni lista viene gestita con appositi puntatori. Ogni nodo della lista livello viene generato nel momento in cui viene raggiunto il punteggio necessario per finire il livello corrente, nel caso in cui si torni al livello precedente la generazione degli ostacoli riprende dalla coda della lista ostacoli del livello corrente.

La classe `lista_obst` contiene 10 funzioni:

`Liv_obst`: genera il primo nodo ostacolo manualmente. Gli ostacoli successivi sono poi generati proceduralmente ed inseriti in nella rispettiva lista.

`Pos_x`: genera casualmente la coordinata dell'ostacolo.

`Rand_id`: genera casualmente l'id dell'ostacolo. Ad ogni ostacolo è stato assegnato un id, ovvero un codice identificativo.

`New_ostacolo`: aggiunge ostacoli in coda alla lista ostacoli. Genera il primo ostacolo del livello separatamente, poi ne genera altri 19, per un totale di 20 ostacoli.

`Clear_liv`: scorre la lista ostacoli e cancella i nodi degli ostacoli il cui id viene cambiato a 0. Ad un ostacolo viene cambiato il proprio id solamente se colpito dal giocatore.

`Next_liv`: se il livello successivo è già stato creato, sposta il puntatore del livello corrente al nodo del livello successivo, altrimenti viene prima generato un nodo in coda alla lista livelli dalla funzione new_liv.

`New_liv`: aggiunge nodo livello alla lista dei livelli. Il back è identificato dal liv precedente. Inizializza i nodi per gli ostacoli e il conteggio livello. Controlla qual è l'ostacolo che ha fatto aumentare di livello. Prede quel puntatore e lo sposta all'inizio del livello e in game si spostano da soli e non devo spostare la lista obst nel livello successivo. Poi crea i 20 ostacoli successivi.

`Liv_down`: sposta il puntatore del livello corrente al nodo precedente della lista livelli tramite il puntatore back. Inoltre sposta il puntatore della lista ostacoli alla fine della lista ostacoli del livello precedente.

Le funzioni `get_liv` e `draw_liv` rispettivamente ritornano e stampano il numero del livello.

La classe `Ostacoli` si occupa di stampare a schermo l'ostacolo opportuno. Contiene tre funzioni:

Costruttore: Inizializza tutti gli elementi della matrice a 0.

`Draw_obst`: carica il valore id dell'ostacolo nella matrice, dandogli la relativa forma.

`Draw_part_obst`: è la funzione corrispondente alla stampa dell'ostacolo riga per riga.

Punteggio

La gestione del punteggio è affidata alla classe `Score`, quando il giocatore si trova al livello 1 e il suo punteggio scende sotto allo 0, il gioco termina mostrando il punteggio massimo raggiunto e la scritta "GAME OVER".

La classe `Score` è formata da 9 funzioni:

Costruttore: inizializza i valori ad un valore preimpostato.

`Score_up`: aumenta punteggio relativo al livello, il punteggio totale attuale e punteggio massimo raggiunto durante l'intera partita.

`Score_down`: diminuisce i valori del punteggio del livello e del punteggio totale. Il punteggio massimo non vene modificato.

`Molt_up`: aumenta o diminuisce il moltiplicatore del punteggio se viene superato o meno un determinato livello.

Le funzioni `get_score_liv`, `get_score_tot` e `get_score_max` ritornano tutte il relativo punteggio, indicato nel nome della funzione.

`Reset_score_liv`: la funzione fa un reset del punteggio relativo al livello corrente quando questo viene superato oppure si ritorna al livello precedente.

`Draw_score`: stampa tutti i punteggi a schermo.

Game

La Classe game si occupa di mettere insieme tutte le parti esposte precedentemente per dare la forma finale al gioco.

Contiene solo la funzione `in_game` che chiama le altre funzioni sopra descritte, inizializza alcune variabili e al suo interno vengono dichiarati gli oggetti che compongono il gioco vero e proprio. Si tratta di un loop che termina solo nel momento in cui il giocatore perde la partita e, in particolare, si occupa di:

1) Definire la posizione della macchina e degli ostacoli
2) Generare e gestire gli ostacoli
3) Verificare le eventuali collisioni
4) Gestire il punteggio
5) Stampare a video sia gli oggetti di gioco che i punteggi
6) Gestire la difficoltà
7) Permettere al giocatore di mettere in pausa
8) Mostrare il menù di game over