Relazione progetto Informatica (2019/2020)

Nella cartella scaricata è già presente il file (CarGame2D.exe) compilato.
Se si desidera compilare il codice bisogna eseguire il comando g++ *.cpp verificando di avere visual c++11 installato.

Entity:
Classe madre di player, bonus e enemy. Viene utilizzata per raggruppare tutti i metodi che gestiscono le coordinate e la dimensione delle entità.

- Metodo isOccupied: verifica che la zona in cui verrà stampata l'entità sia libera.

Player
Gestisce il giocatore, si occupa di controllare le collisioni (collideBonus e collideEnemy), di muovere il player (move) e di eliminare la posizione precedente dal canvas (remove).
Ha come attributo un oggetto di tipo Sprite che gestisce la rappresentazione grafica del player (viene istanziato nel costruttore).

- Metodi collideBonus e collideEnemy: gestiscono le collisioni tra player e enemy/bonus, controllando le hitbox tramite il metodo Axis-Aligned Bounding Box (collisioni tra forme rettangolari).
- Metodo collideWalls: controlla che il player non esca dal campo di gioco.
- Metodi draw e remove: passano il canvas, altezza, larghezza e le coordinate della posizione del player al metodo draw dello sprite che si occupa del disegno effettivo.
- Metodo move: Muove il player verso una delle otto direzioni possibili. Al suo interno richiama il metodo getMovementInput della classe Input per sapere quale tasto è stato premuto. Il player si muove di due celle alla volta in confronto al resto delle entità che si muovono solo di una cella, in questo modo si avvantaggia il player nei primi livelli ma lo si penalizza nei livelli più avanzati in quanto risulta più difficile da controllare quando più la velocità aumenta.

Bonus
Simile a enemy. Nel costruttore della classe viene passato il bonus che può conferire allo score una volta "raccolto". Non ha metodi per verificare le collisioni perchè vengono controllate dalle altre classi.
Controlla però la collisione con il bordo inferiore.

Enemy
Nel costruttore della classe enemy viene passato anche il danno che causa. Gestisce le collisioni con entità (collideEntity), bonus (collideBonus), mura laterali (collideLateralWalls) e muro inferiore (collideBottomWall). Muove il nemico (move) e disegna (draw) o rimuove (remove) lo sprite.

- Metodi collide CollideEntity e collideBonus: richiedono come parametro il vettore che contiene tutti i nemici/bonus presenti a schermo e verificano che non stiano collidendo, se questo avviene allora il nemico viene spostato indietro di una cella in base alla sua direzione. CollideLateralWalls è simile.
- CollideBottomWall verifica che il nemico non stia per uscire dal canvas, se ciò stà per accadere allora restituisce true.
- Metodo move: muove il nemico verso sinistra o destra di una cella in maniera casuale mentre lo sposta sempre verso il basso di una cella.

Sprite
Si occupa di disegnare e cancellare le entità dal canvas. Ha come parametro una lista di stringhe che rappresentano l'entità disegnata.

- Metodo addLine: aggiunge una riga al vettore, si aspetta una stringa (la riga)
- Metodo draw: disegna lo sprite sul canvas
- Metodo remove: cancella lo sprite dal canvas (sostituisce i suoi caratteri con spazi bianchi)

Game

La classe game si occupa di gestire tutti i metodi che controllano il gioco come la gestione delle collisioni tra tutte le entità, la modifica del livello e il disegno sul canvas.

- Metodo checkCollision: controlla innanzitutto le collisioni tra player e enemy in questo modo posso ridurre il numero di cicli nel controllo delle collisioni tra enemy e enemy. Ripeto la stessa cosa anche con bonus.
- Metodo initCanvas: crea il canvas iniziale e disegna i bordi del gioco. Viene chiamato solo una volta nel costruttore della classe.
- Metodo move: muove prima il player e poi tutto il resto. Quando muove un'entità verifica anche che non superino il bordo inferiore del campo da gioco, se ciò succede cancella quell'entità (nel caso del player blocca solo il movimento).
- Metodo draw: disegna tutte le entità nel canvas alle nuove posizioni, iniziando da player poi enemy e infine bonus. Come ultima cosa chiama la funzione drawBuffer della classe ConsoleDrawing per disegnare a schermo il canvas.
- Metodi addEnemy e addBonus: spawnano un nemico/bonus nuovo controllando che l'area di spawn sia libera (tramite il metodo isOccupied della classe Entity).
- Metodo checkLevel: controlla se lo score sia maggiore della variabile "levelUpTarget" se è così allora aumenta il livello, i nemici e i bonus, riduce la variabile speed (più lenta è più veloce è il gioco), aumenta il danno dei nemici e il valore conferito dai bonus. Se score è inferiore alla variabile prevLevel allora il giocatore retrocede di un livello.
- Metodo resetCanvas: richiama i metodi delete delle entità, in questo modo non è necessario riscrivere tutto il canvas ma solo le celle dove erano presenti le entità.
- Metodo showStats: crea un cavas che mostra il livello e lo score della partita. Restituisce la combinazione del canvas di gioco e quello delle statistiche.
- Metodo gameOver: stampa la schermata gameOver e chiede di premere il tasto invio o barra spaziatrice per giocare di nuovo.

ConsoleDrawing
La classe ConsoleDrawing viene utilizzata per disegnare il gioco sullo schermo.
E' pensata per avere il minimo numero di chiamate a std::cout, che risultava troppo lento per la corretta funzionalità del gioco. Per ottenere il minimo numero di chiamate utilizza un buffer che si ricorda lo stato del gioco all'istante precedente a quello in cui ci si trova. In questo modo viene modificato a schermo esclusivamente ciò che cambia tra lo stato attuale e il buffer.

Al suo interno viene inclusa la libreria windows.h, utile nelle funzioni "ShowConsoleCursor" e "setCursorPosition".

Funzionamento dei metodi della classe:

- "setCursorPosition": wrapper per la funzione "SetConsoleCursorPosition" della libreria "windows.h", creata per facilitarne la chiamata. Ha due parametri che identificano le coordinate all'interno della console.
- "DrawAtStart": disegna l'intero vettore nella console per la prima volta e poi copia il vettore nel buffer, in modo tale che successivamente sia possibile confrontare il vettore del momento in cui ci si trova con quello iniziale (utile durante la prima chiamata delle funzione "DrawBuffer").
- "DrawBuffer": disegna il vettore nella console in tutti i momenti a parte il primo. Controlla che il vettore che gli viene passato come parametro sia della stessa dimensione del buffer (senza questo controllo in caso di mancata consistenza della dimensione il programma crasha). Poi controlla se sono avvenute modifiche al vettore rispetto al buffer e stampa solo le modifiche. Infine copia il vettore nel buffer per mantenerlo aggiornato.
- DrawGameOver: quando viene chiamata stampa a schermo la scritta GameOver.
- "ShowConsoleCursor": rende invisibile la posizione del cursore nella console.

Input
La classe Input viene utilizzata per prendere l'input dalla tastiera. Utilizza il metodo "GetAsyncKeyState" della libreria "windows.h" per rilevare la pressione dei tasti.

Ha due metodi, uno (getMovementInput) per rilevare la pressione dei tasti utilizzati per il movimento e l'altro (getMenuInput) per i tasti utilizzati per il menu.
Per controllare la macchina durante il gioco è possibile utilizzare sia le frecce direzionali presenti nella tastiera, sia la più moderna configurazione WASD, in cui W per andare verso l'alto, A per andare a sinistra, S per andare in basso e infine D per andare a destra.
Fino a due tasti possono essere usati contemporaneamente, in modo tale da arrivare a restituire 8 direzioni totali.

LevelInterface
Gestisce la rappresentazione scritta del livello e dello score.

- Metodo setHeight: imposta l'altezza del canvas uguale a quella del parametro che richiede in input.
- Metodo getCanvas: restituisce il canvas.
- Metodo drawCanva: scrive nel canvas il livello e lo score che richiede come parametro.
- Metodo initCanvas: disegna i bordi del canvas

Suddivisione ruoli
Mattia Babbini: Game, Entity e Sprite.
Alessandro Libralesso: Input e ConsoleDrawing
Elena Lippolis: LevelInterface, i metodi showStats e gameOver della classe Game
Luca Paparo: Player, Bonus ed Enemy