SQUARE INVADERS
Player e Bullets

CORSO DI GAME PROGRAMMING
1° ANNO

Docente **Davide Caio**

ACCADEMIA ITALIANA VIDEOGIOCHI

GFX TOOLS

Classe statica che contiene le utility per le primitive di disegno.

- Clean () → pulisce lo schermo
- PutPixel (int x, int y, byte r, byte g, byte b) → inserisce un pixel in posizione x, y
- DrawHorizontalLine (int x, int y, int width, byte r, byte g, byte b) → disegna una linea orizzontale di lunghezza width.
- DrawVerticalLine (int x, int y, int height, byte r, byte g, byte b) → disegna una linea verticale di altezza height.
- DrawRect (int x, int y, int width, int height, byte r, byte g, byte b) → disegna un rettangolo di dimensioni width * height

VECTOR 2

Struct che definisce un vettore di due dimensioni.

Campi:

- X → componente x
- Y → componente Y

Metodi:

- Costruttore Vector2 (float x, float y) → inizializza le due componenti del vettore.
- Sub (Vector2 vec) → sottrae il vettore vec al vettore corrente.
- GetLenght () → ritorna la lunghezza del vettore.

COLOR

Struct che definisce colore (RGB).

Campi:

- R → componente red
- G → componente green
- B → componente blue

Metodi:

- Costruttore Color (byte r, byte g, byte b) → inizializza il colore con le componenti passate come parametro.

RECT

Classe che definisce un Rettangolo

Campi:

- position → posizione del rettangolo
- width → larghezza rettangolo
- height → altezza rettangolo
- color → colore del rettangolo

Metodi:

- Costruttore Rect (float xPos, float yPos, int w, int h, Color col) → inizializza il rettangolo.
- Translate(float x, float y) → trasla il rettangolo.
- Draw() → disegna il rettangolo.

RANDOM GENERATOR

Classe statica che nasconde la classe Random di System

Campi:

- random → istanza singleton della classe Random

Metodi:

- GetRandom (int min, int max) → ritorna un numero random tra min e max dell'istanza Random random.

OBJECT POOLING

Creare un nuovo oggetto oppure distruggerlo (manualmente oppure perdendo ogni reference alla sua istanza) è un'operazione costosa perché allocare e deallocare memoria durante l'esecuzione del software.

Nel Game Development è buona norma evitare di allocare e deallocare memoria durante il gameplay fin dove è possibile. Questo vuol dire che la creazione e distruzione di oggetti dovrebbe avvenire solamente durante il cambio di scena (il caricamento di un livello, mappa, menu).

Una tecnica di ottimizzazione volta ad evitare allocazione e deallocazione di memoria durante il gameplay è l'**object pooling**.

L'object pooling consiste nella creazione di un insieme di oggetti (la pool), solitamente raggruppati in una collezione fissa come un array, durante la creazione del livello oppure all'avvio del gioco (come nel caso del nostro Square Invaders). All'interno questi oggetti avranno un campo che definisce la sua attivazione e in base a questa attivazione l'oggetto effettuerà delle operazioni all'interno del game loop.

Quando durante il gameplay avremo bisogno di un oggetto di quel tipo, ci basterà andare nella pool e richiedere il primo oggetto che ha lo stato attivazione a false per poterlo attivare e utilizzarlo a nostro piacimento.

Una volta che l'oggetto non deve più essere utilizzato, basterà cambiare il valore del campo attivazione per escluderlo dal game loop e riporlo all'interno della pool pronto per essere riattivato nel momento in cui da gameplay avremo bisogno di un nuovo oggetto di quel tipo.

BULLET

Campi:

- width: larghezza del proiettile (totale).
- height: altezza del proiettile (totale).
- velocity: velocità del proiettile.
- color: colore del proiettile.
- Position: posizione del proiettile.
- IsAlive: flag che ti dice se in questo momento è spawnato.

Metodi:

- Costruttore Bullet (int w, int h, Color col) → inizializza larghezza altezza e colore proiettile.
- Update () → muove il proiettile basandosi sulla sua velocità.
- Draw () → disegna il proiettile.
- Shoot (Vector2 startPos, Vector2 startVelocity) → fa partire il proiettile in una determinata posizione e con una determinata velocità.

PLAYER

Classe che definisce il giocatore

Campi:

- position → posizione della navicella
- width → larghezza del proiettile.
- height → altezza del proiettile.
- velocity → velocità del proiettile.
- baseRect → base della navicella
- cannonRect → cannone della navicella
- speed → velocità della navicella
- maxSpeed → velocità massima della navicella
- distToSide → distanza addizionale (oltre la width/2) per cui la navicella si ferma ai bordi della finestra
- bullets → pool di proiettili. OBJECT POOLING
- counter → timer per poter sparere di nuovo
- shootDelay → tempo di ricarica del cannone
- color → colore della navicella.

Metodi:

- Costruttore Player(Vector2 pos, int width, int height, Color col) → inizializza la navicella del giocatore.
- GetFreeBullet () → prende il primo proiettile dormiente dalla pool.
- Shoot () → spara un proiettile.
- Input () → controlla l'input del giocatore per muovere la navicella e/o sparare
- Update () → muove la navicella e tutti i proiettili attivi in quel momento.
- Draw () → disegna la navicella e tutti i proiettili attivi in quel momento.
- GetPosition () → ritorna la posizione della navicella.

GAME

Classe statica che definisce la logica di gioco (creazione window, player e nemici e gestione del tutto) e si occupa di gestire il game loop

Campi:

- window → la finestra dove viene renderizzato il gioco.
- player → il giocatore
- totalTime → il tempo di gioco

Metodi:

- Costruttore Game() → crea la finestra e inizializza il giocatore.
- GetPlayer () → ritorna l'istanza corrente del giocatore.
- Play () → Gestisce il game loop.

PROGRAM

Nella classe program, all’interno del metodo statico Main, semplicemente basterà chiamare il metodo statico Play della classe Game.