Fondamenti di Computer Graphics M (8 cfu)

Lab 02 - 2D Animazione Gameplay

Lorenzo Righi

2 marzo 2024

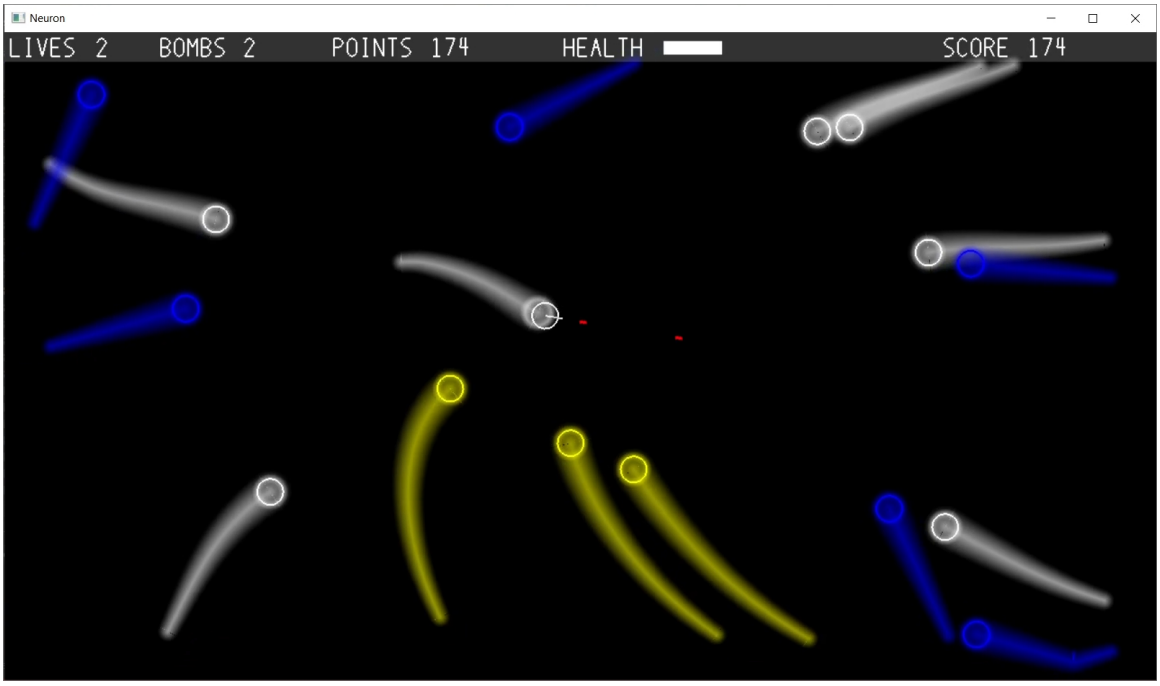


Figura 1: Demo 2D Neuron

Traccia

Si richiede di realizzare una demo di animazione digitale 2D interattiva simile al materiale fornito. Lo studente può utilizzare il codice fornito come materiale di ispirazione e consultare il sito glslsandbox che mostra diversi esempi 2D. Tuttavia non saranno accettati elaborati che si limiteranno ad apportare banali modifiche al materiale di base.

1 Introduzione

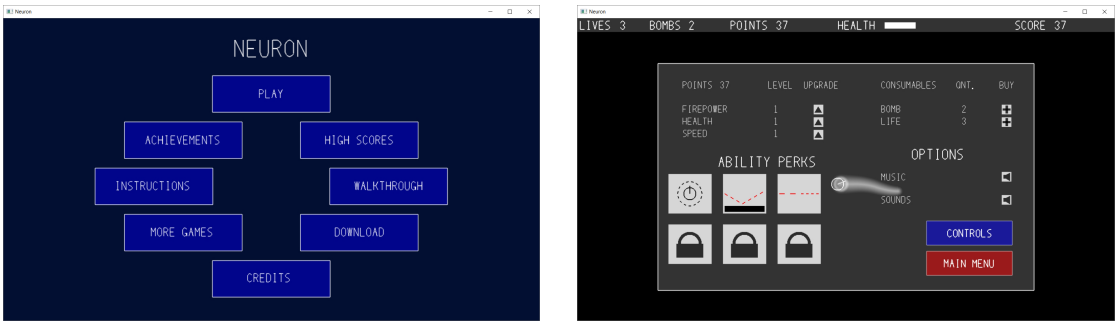
Neuron è un videogioco di tipo arena shooter, ispirato a un vecchio gioco in flash per browser.

2 Gameplay

Il giocatore controlla una navicella che può muoversi e sparare nello spazio di gioco. I nemici vengono generati ad ondate lungo i bordi della finestra, e il giocatore per proseguire deve distruggerli cercando di non farsi colpire. In base allo stage e all'ondata corrente il numero e il tipo dei nemici sarà diverso, ovviamente seguendo una logica di difficoltà incrementale. Quando un nemico viene distrutto rilascia una o più ricompense, queste seguono in automatico il giocatore e una volta raccolte conferiscono dei punti che possono essere spesi in una sezione dedicata del menu di pausa. L'obiettivo del gioco è distruggere i nemici delle ondate accumulando quanti più punti possibili, e termina o completando l'ultimo stage o quando la barra della salute raggiunge lo zero in mancanza di vite extra.

2.1 Menu

Il gioco prevede due menu, quello principale ed esterno mostrato in Fig. 2a che permette di iniziare una nuova partita o di accedere a sezioni come i crediti o i comandi di gioco, e quello interno di pausa mostrato in Fig. 2b, a cui è possibile accedere durante la partita premendo 'p'.



(a) Main Menu

(b) Pause Menu

Figura 2: Menu di gioco

Nel menu di pausa è possibile consumare i punti per aumentare la potenza di fuoco (Fig. 3c), la salute massima o la velocità della navicella.

(a) Firepower 1 (b) Firepower 2 (c) Firepower 5

Figura 3: Potenza di fuoco

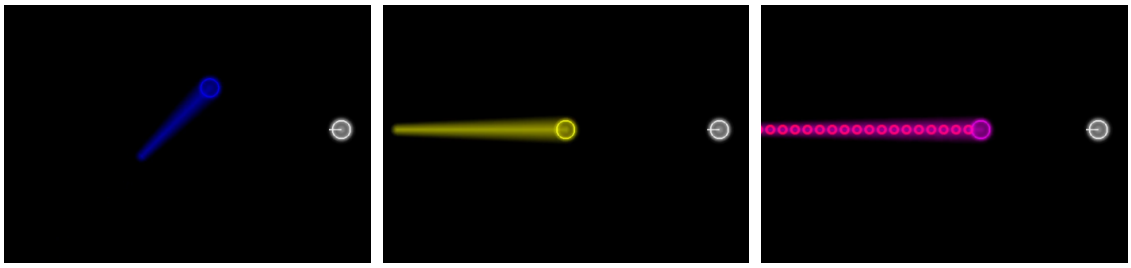
E' inoltre possibile acquistare vite extra e bombe, che possono essere usate per distruggere tutti i nemici presenti, e attivare una o più abilità tra le 6 disponibili:

- `Reduced Size`, riduce le dimensioni del giocatore aiutandolo a destreggiarsi tra i nemici
- `Bouncing Bullets`, permette ai proiettili di rimbalzare due volte sulle pareti della finestra prima di svanire, aumentando le probabilità di colpire un nemico
- `Double Time`, riduce il cooldown di fuoco dei proiettili, raddoppiandone la cadenza di tiro
- `Penetrating Bullets`, conferisce ai proiettili la capacità di sopravvivere all'impatto con i primi due nemici che penetrano
- `Explosive Bullets`, i proiettili possono esplodere all'impatto distruggendo tutti i nemici in una piccola area circolare, utile negli stage che prevedono l'accumulo di molti nemici
- `Time Alter`, altera il flusso temporale di gioco per un breve periodo, consentendo al giocatore di destreggiarsi meglio tra i nemici

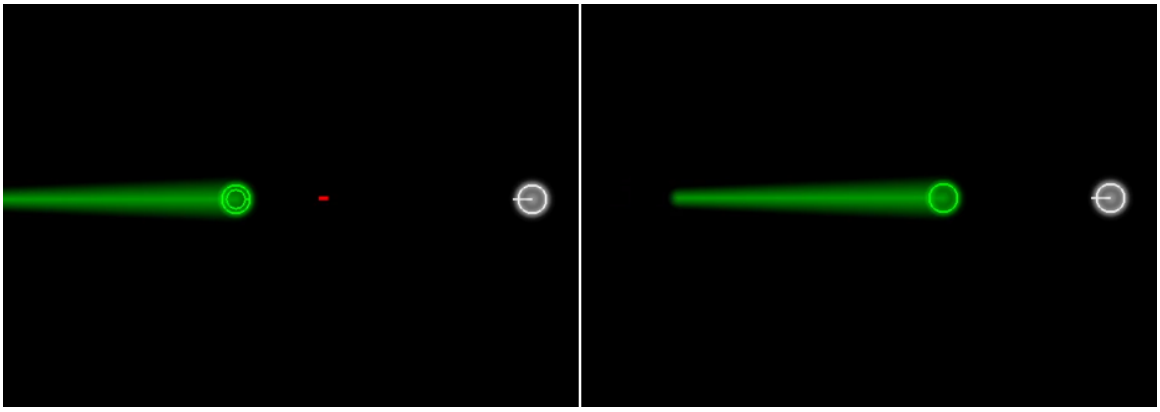
2.2 Nemici

Il gioco prevede 7 tipi di nemici:

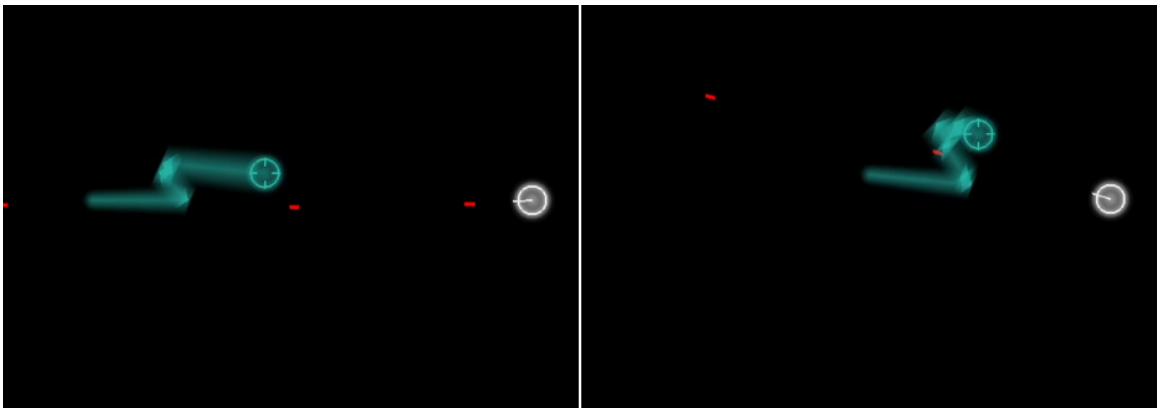
- `Blu`, si muovono linearmente e rimbalzano contro le pareti
- `Gialli`, si muovono seguendo il giocatore
- `Magenta`, si muovono seguendo il giocatore, hanno velocità incrementata



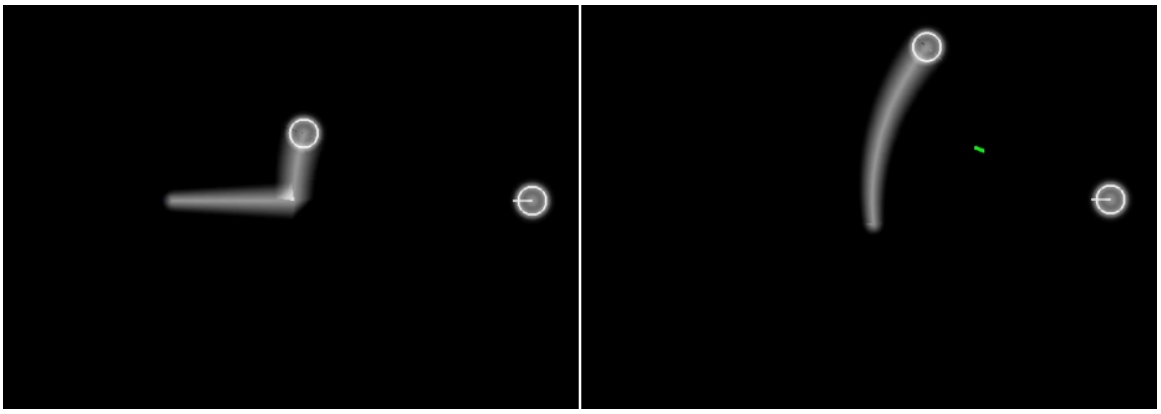
- `Verdi`, si muovono seguendo il giocatore, hanno due vite



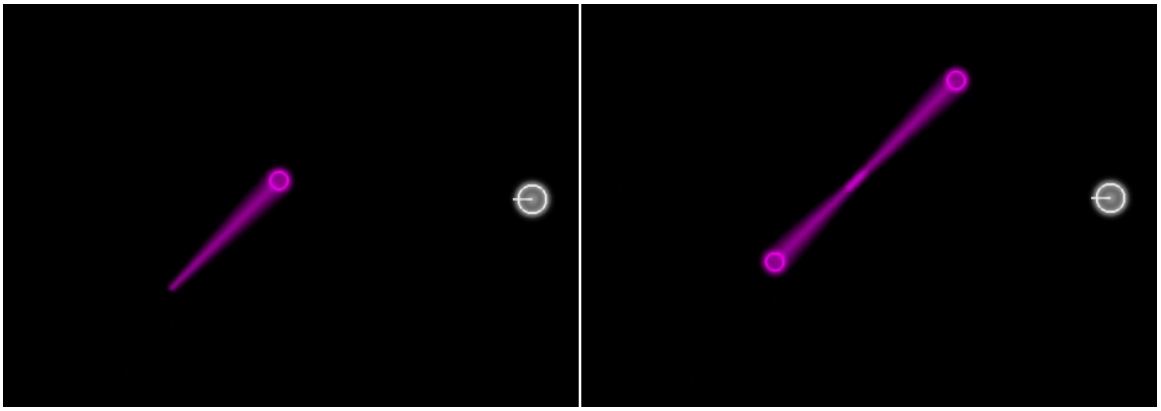
- `Ciano`, si muovono seguendo il giocatore, cercano di evitare i proiettili sparati



- `Bianchi`, sotto a una distanza di soglia si muovono seguendo il giocatore, sopra ad essa si muovono ruotandovi attorno. Periodicamente sparano un proiettile in direzione del giocatore



- `Viola`, si muovono linearmente e rimbalzano contro le pareti. Periodicamente generano in direzione opposta alla propria un figlio con le medesime proprietà



2.3 Comandi

I comandi previsti sono i seguenti:

- `CLICK DESTRO`, all'interno dei menu: selezione dell'opzione in hover; all'esterno (partita in corso): se il mouse è dentro la finestra di gioco spara un proiettile in direzione del cursore a partire dalla navicella
- `W`, muove la navicella verso l'alto
- `A`, muove la navicella verso sinistra
- `S`, muove la navicella verso il basso
- `D`, muove la navicella verso destra

- `P`, apre/chiude il menu di pausa
- `R`, mostra/nasconde i segmenti dei lati dei triangoli per il rendering delle scie delle navicelle
- `SPACEBAR`, se il contatore delle bombe è ¿ 0, ne consuma una e tutti i nemici presenti vengono distrutti in un area circolare con raggio incrementale a partire dalla posizione del giocatore (Fig. 5)
- `SHIFT`, se la perk "Time alter" è selezionata, ne attiva/disattiva l'effetto
- `ESCAPE`, se si è nell'interfaccia del menu principale, termina l'esecuzione

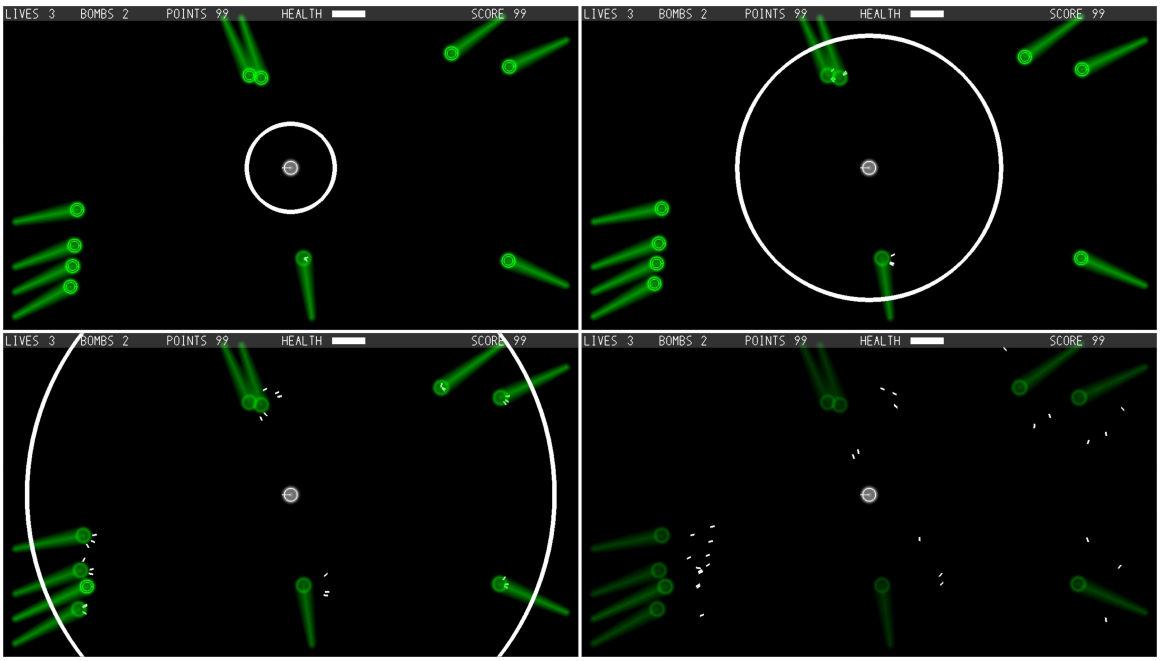


Figura 5: Sequenza Bomba

3 Implementazione

Il gioco ha una composizione molto semplice, prevede un main che chiama tutte le funzioni init necessarie ed inizializza ogni parametro di gioco, dopodiché avvia il Game Loop. Le strutture dati principali sono quella della navicella, contenente posizione, angolo e punti per la draw; quella dei nemici che è un'estensione della navicella con l'aggiunta di informazioni sul tipo ed il comportamento, una struttura per i proiettili ed una con i dati della sessione di gioco quali stage, ondate, salute, punteggio e variabili di stato.

3.1 Funzioni principali

Le funzioni principali che compongono la struttura del codice sono le seguenti:

- `init()`, funzione che chiama tutti i sotto-metodi di inizializzazione come initStages(), che popola la struttura dati per la gestione dei nemici; initGame(), che inizializza la struttura con i parametri di gioco; initMainMenu(), initPauseMenu(), initStatsBar() e initGameOver() si occupano invece della creazione delle interfacce grafiche da mostrare a schermo
- `drawScene()`, funzione che si occupa del disegno effettivo su schermo dei punti contenuti nelle strutture dati adibite. Con opportune variabili di controllo contenute nella struttura Game discrimina quale sia l'interfaccia di top level, ed evidenzia elementi in caso di un eventuale hovering con il mouse
- `update()`, funzione che implementa il Game Loop

3.2 Game Loop

Il Game Loop è implementato dalla update, la quale chiama le seguenti sotto-funzioni di aggiornamento per ogni frame:

- `checkStage()`, controlla stage ed ondata corrente e si occupa della generazione dei nemici in accordo con la lista pre-caricata tramite initStage(), inoltre verifica lo stato della partita ed innesca la sequenza di Game Over qualora questa sia terminata.

```
1    void checkStage() {
2        if (game.currentStage == stages.size()
3            gameOver();
4        else
5            //logica generazione nemici
6        ...
7    }
```

- `updateNemici()`, aggiorna gli spostamenti dei nemici in accordo al comportamento previsto dalla loro tipologia
- `updateProiettili()`, aggiorna lo spostamento di ogni oggetto proiettile; per semplicità proiettili del giocatore, ricompense e proiettili nemici sono gestiti tutti attraverso la stessa struttura dati, la cui unica differenza grafica è il colore, ed il loro comportamento è discriminato da un intero "tipo" che assume valori 0, 1 e 2

```
1    void updateProiettili() {
2        for(int i = 0; i < proiettili.size(); i++) {
3            if(proiettili.at(i).type == 0)
4                //comportamento proiettili del giocatore, movimento lineare
5            else if (proiettili.at(i).type == 1)
6                //comportamento ricompense, movimento con componente angolare
7            else
8                //comportamento proiettili dei nemici, movimento lineare
9        }
10   }
```

- `checkCollisioni()`, si occupa di controllare, per ogni nemico, se è avvenuta una collisione con un proiettile o con il giocatore, nel primo caso il nemico viene flaggato e al termine della checkCollisioni si cicla nuovamente su tutti i nemici per rimuovere quelli segnati; nel secondo al giocatore viene tolta una quantità di salute dal totale in base al tipo del nemico, anche in questo caso flaggato per la successiva distruzione: se la salute del giocatore scende sotto lo zero questo perde una vita o, in mancanza di vite extra, la partita. Nella checkCollisioni viene anche controllato quali dei proiettili di tipo 1 (ricompense) hanno urtato il giocatore: questi vengono considerati raccolti, dunque vengono rimossi e il punteggio incrementato parallelamente.

```
1    void checkCollisioni() {
2        for(int i = 0; i < nemici.size(); i++) {
3            if(dist_N_G < raggiN[nemici.at(i).type] + raggioG)
4                //giocatore colpito, decremento vita e flaggo nemico
5            else {
6                for(int j = 0; j < proiettili.size(); j++) {
7                    if(proiettili.at(j).type == 0)
8                        if (dist_N_P < raggiN[nemici.at(i).type])
9                            //flaggo nemico e rimuovo il proiettile
10                   }
11               }
12       }
13       for(int i = 0; i < proiettili.size(); i++) {
14           if(proiettili.at(i).type == 1)
15               if(dist_G_P < raggioG)
16                   //ricompensa raccolta, incremento punteggio e rimuovo il proiettile
17       }
18       ...
19   }
```

4 Analisi degli elementi grafici

Non sono stati realizzati oggetti di grande complessità, pertanto gli elementi più interessanti risultano essere l'hovering sui pulsanti dei menu, l'animazione del Game Over e le scie dei nemici

4.1 Hover

Quando ci si trova nell'interfaccia del menu principale o del menu di pausa i pulsanti che possono essere premuti vengono evidenziati non appena viene rilevato l'hovering del cursore del mouse su di essi, questo effetto è ottenuto modificando il valore della trasparenza dei vertici che costituiscono i disegni

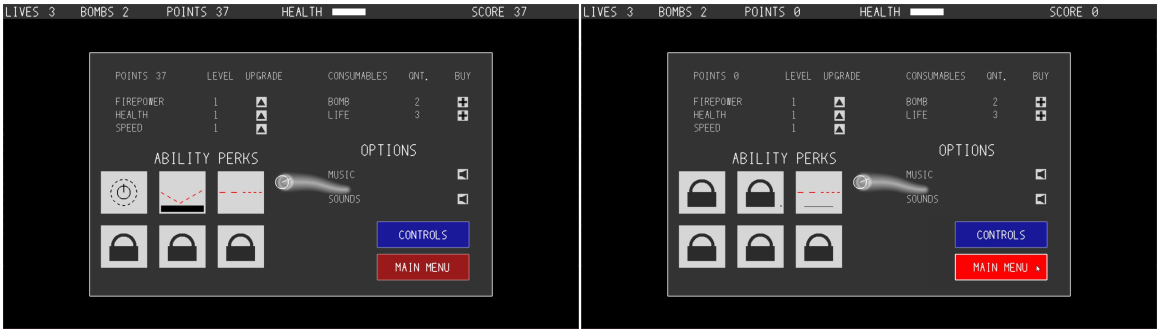


Figura 6: Hovering

4.2 Game Over sequence

Quando il giocatore esaurisce le proprie vite o quando completa il gioco, viene mostrata a schermo la scritta Game Over seguita dal punteggio e dai bottoni per iniziare una nuova partita o per tornare al menu principale. L'animazione della scritta è ottenuta aggiornando con un'opportuna combinazione di valori le matrici di traslazione e scala delle trasformazioni applicate alle strutture dati contenenti i vertici delle scritte, e alla modifica in parallelo della trasparenza del loro colore.

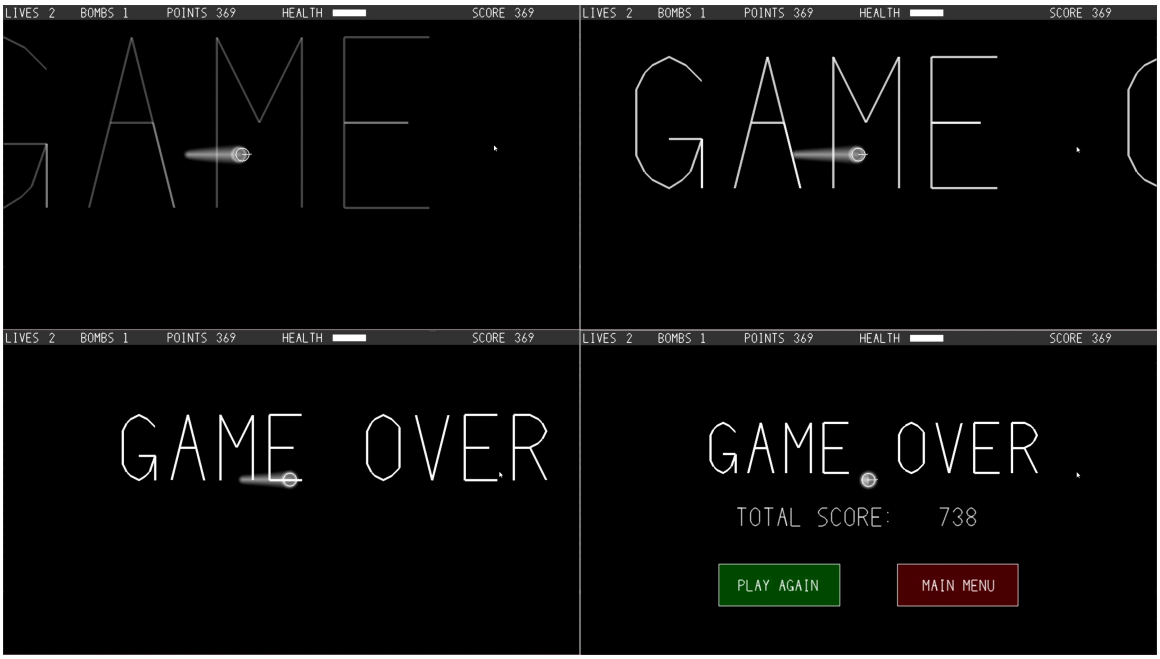


Figura 7: Game Over

4.3 Scia delle navicelle

il corpo principale del giocatore e dei nemici è costituito da un semplice cerchio, con piccole variazioni in base al tipo (il giocatore è provvisto di un cannone, i nemici di tipo 3 hanno un secondo cerchio interno, quelli di tipo 5 hanno segmenti aggiuntivi che ricordano un mirino...), l'unica parte leggermente più complessa riguarda la scia che si lasciano dietro durante gli spostamenti, questa è infatti costituita dai seguenti componenti:

- `Due corone circolari`, una è esterna al cerchio del corpo principale ed una è interna a questo, entrambe hanno rispettivamente i vertici esterni ed interni con trasparenza massima al fine di creare un effetto sfumato, e quella esterna ha il raggio interno che coincide con il raggio esterno di quella interna
- `Un semicerchio`, rappresenta l'inizio della scia, è centrato nello stesso punto del nemico, ha lo stesso raggio del corpo ed è rivolto nella direzione in cui la navicella si sta muovendo
- `Una serie di triangoli`, questi costruiscono la scia, non hanno una forma ben definita perché la direzione dei nemici può variare e questa viene costruita sulle posizioni della navicella nei frame precedenti. Idealmente se il nemico mantiene la stessa direzione questa parte della scia assume la forma di un trapezio isoscele, con i vertici sui lati aventi trasparenza massima per l'effetto sfumato

- `Un secondo semicerchio`, chiude la coda della scia con un semicerchio più piccolo rispetto al primo, rivolto in direzione opposta

In Fig. 8 sono mostrati i triangoli utilizzati per il rendering della coda delle navicelle (non vengono mostrati quelli delle due corone circolari attorno al corpo poiché non sufficientemente grandi da renderne osservabile la composizione).

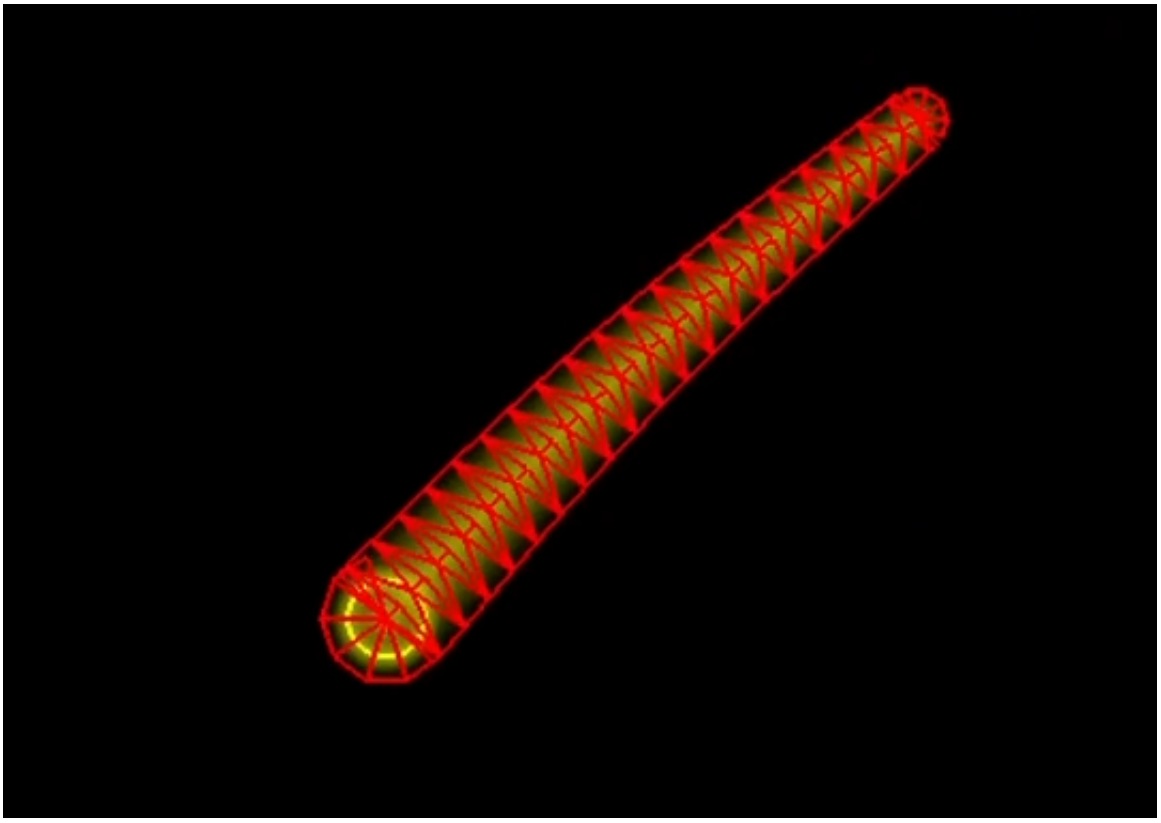


Figura 8: Trail Rendering Triangles