Cannon

Cristian Perissutti

Progetto per il corso di
"Informatica e Sistemi in Tempo Reale"

Prof. Giorgio C. Buttazzo
Università di Pisa

Contents

1 Descrizione del progetto

1.1 Scopo

Lo scopo del progetto è simulare, in ambiente 3D, un cannone controllabile dall'utente il quale deve colpire dei bersagli che si muovono lentamente dalla parte opposta di un muro.

Tramite le frecce direzionali è possibile cambiare la posizione e l'angolazione del cannone. Alla pressione del tasto <SPACE > viene sparato il proiettile, il quale segue una traiettoria descritta dalle leggi che verranno spiegate in seguito.

Se il proiettile colpisce uno dei due bersagli il punteggio viene incrementato e quando si raggiungono i 10 punti si vince il gioco. Dopo ogni sparo la posizione e l'altezza del muro vengono modificate casualmente.

È inoltre possibile attivare l'effetto di un vento con direzione e velocità casuali premendo il tasto <V >. Alla pressione del tasto <T > è possibile attivare un mirino che indica dove atterrerà il proiettile con un certo errore.

L'interazione con l'utente avviene quindi mediante la tastiera e il mouse, ed in particolare attraverso i comandi:

- Tasto sinistro del mouse + drag: cambia orientazione della visuale;
- <W >, <A >, <S >, <D >, <Q >, <E >: spostamento della visuale;
- <SU >, <GIÚ >: cambia angolazione del cannone;
- <SX >, <DX >: trasla il cannone;
- <SPACE >: sparo di un proiettile;
- <V >: attiva/disattiva vento;
- <T >: attiva/disattiva mirino;
- <M >, <N >: aumenta/diminuisce potenza del cannone;
- <ENTER >: comincia una nuova partita (solo in caso di vittoria);
- <ESC >: esce dal programma in qualsiasi momento;

1.2 Modello fisico

Il moto del proiettile è definito tenendo conto della velocità iniziale (potenza del cannone) e delle forze di gravità e attrito viscoso dell'aria.

L'effetto dell'attrito viscoso è stato modellato utilizzando la drag equation piuttosto che la formula di Stokes per renderlo più verosimile al caso di un cannone reale (quindi con velocità del proiettile molto elevate).

La velocità è stata scomposta nelle componenti Vx, Vy, Vz; Le variazioni di velocità sono le seguenti:

- $Vx = Vx - \frac{Fvx*dt}{m} * sign(Vx)$
- $Vy = Vy - g * dt - \frac{Fvy*dt}{m} * sign(Vy)$
- $Vz = Vz - \frac{Fvz*dt}{m} * sign(Vz)$

Siano Cd il drag coefficient, rho la densità dell'aria, V la velocità relativa dell'aria e A l'area della sezione del proiettile, si ha che la forza del vento è data da:

$Fv = \frac{Cd*rho*A*V^2}{2}$

La posizione del proiettile lungo la direzione x viene quindi calcolata come

$posx = posx + Vx * dt$

Analogamente vengono calcolate posy e posz.

2 Ambiente di Sviluppo

Per lo sviluppo del progetto si è utilizzato un DELL Inspiron 3501, lavorando su una partizione della memoria su cui è stato installato Ubuntu 22.04.

Da terminale si è provveduto ad installare le librerie allegro (versione 4.4.3), e allegroGL (versione 0.4.3), mentre il compilatore gcc, la libreria pthread e OpenGL erano già presenti nel sistema.

3 Interfaccia grafica

L'interfaccia grafica, realizzata tramite le librerie allegro e allegroGL, è mostrata in Fig.1. Essa ha una risoluzione di 1280x720.

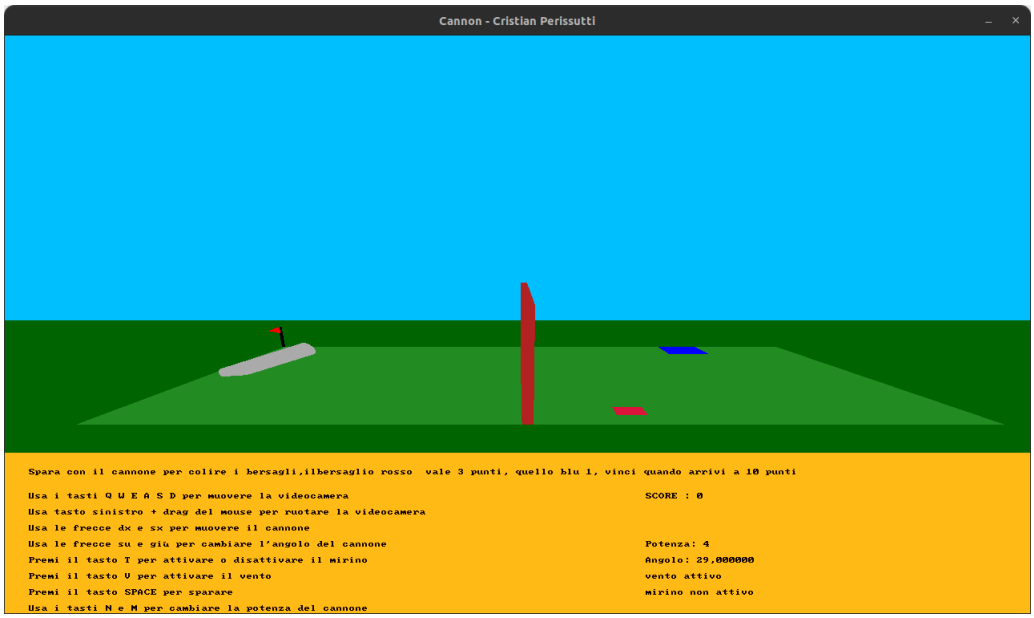

Figure 1: Interfaccia grafica del programma

3.1 Ambiente 3D

Nella parte superiore della finestra è mostrata una rappresentazione della porzione di ambiente 3D osservata.

In generale l'ambiente è formato da uno sfondo azzurro e da un piano di colore verde scuro (ground), mentre l'area di interesse (playground) è un rettangolo di colore verde chiaro.

Il parallelepipedo di colore rosso è il muro che viene rigenerato con posizione e altezza casuali dopo ogni sparo. Il muro divide il playground in due aree: area cannone e area target, indicate in Fig. 2.

Nell'area cannone sono presenti:

- Il cannone, di colore grigio chiaro;
- Il proiettile, di colore grigio scuro. Inizialmente è collocato allinterno del cannone, ed è quindi visibile solo durante lo sparo, Fig. 3;
- Una bandiera presente in un angolo. Questo elemento è formato da un'asta di colore nero e, se il vento è attivo, da un triangolo di colore rosso direzionato in maniera concorde al vento;

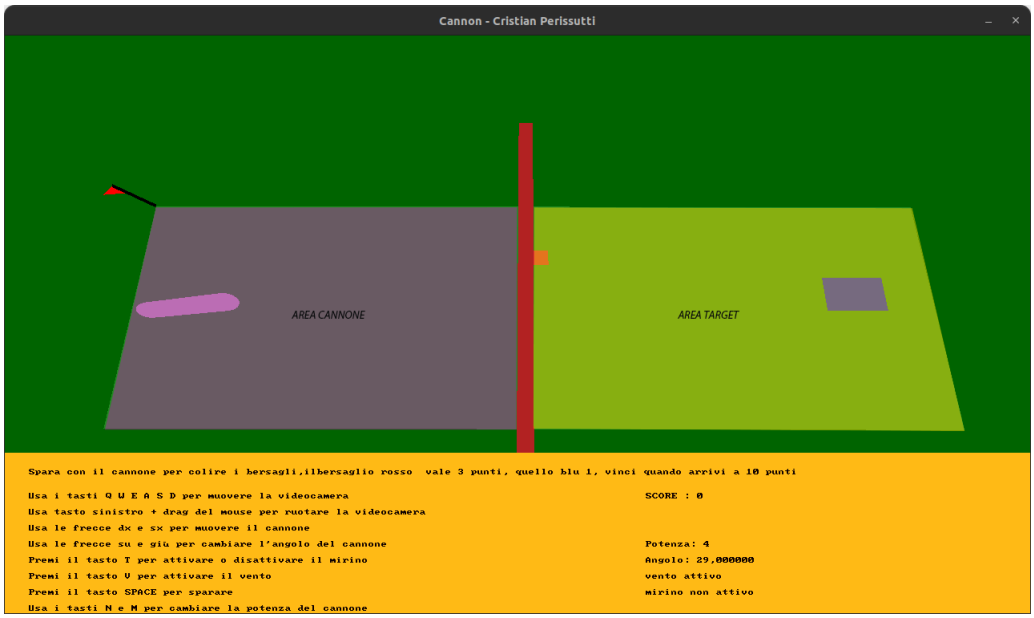

Figure 2: Divisione in area Cannone e area Target

Nell'area target sono presenti due target che si muovono lentamente a livello del terreno. Se un target sta per uscire dall'area target rimbalza sul bordo e continua il suo moto. Il target di colore blu è più grande e se colpito aumenta il punteggio di 1 punto, il target di colore rosso ha dimensioni inferiori e se colpito aumenta il punteggio di 3 punti.

Quando il mirino è attivo, una piccola sfera di colore arancione a livello del terreno indica il punto di atterraggio del proiettile. Tale punto è volutamente impreciso e non tiene conto di una eventuale collisione con il muro in modo da evitare di rendere troppo semplice colpire i bersagli.

3.2 Menù

Nella parte inferiore della finestra è presente un menù su cui sono indicate le istruzioni di base che illustrano le funzionalità dei vari tasti e alcune informazioni aggiuntive tra cui il punteggio, linclinazione del cannone e potenza del cannone. Per la realizzazione del menù sono state utilizzate esclusivamente le funzionalità della libreria allegro.

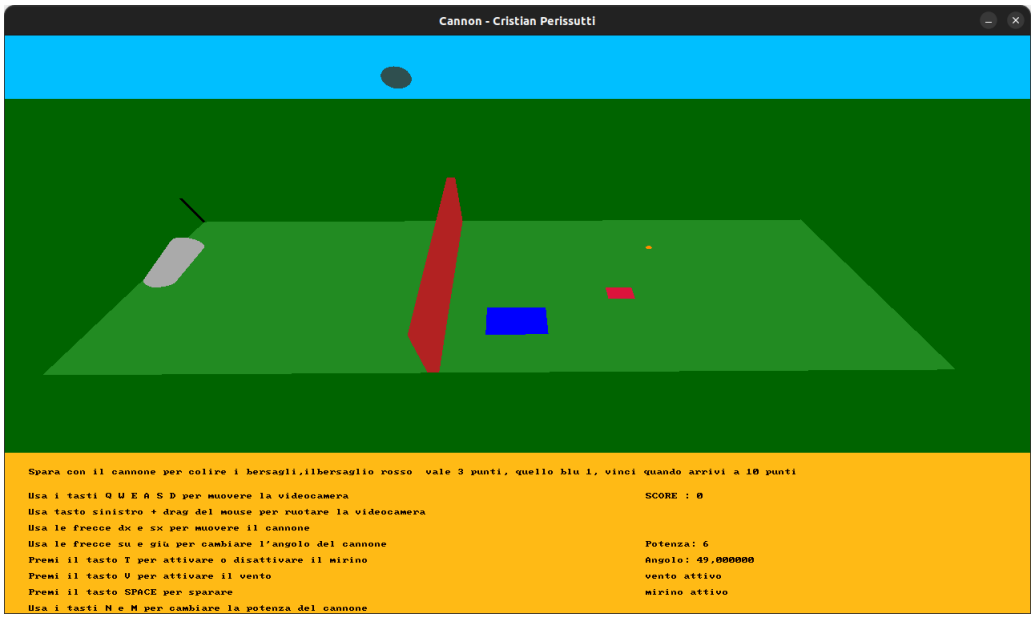

Figure 3: Visuale durante lo sparo, con mirino arancione attivo

3.3 Schermata di Vittoria

Al raggiungimento dei 10 punti è visualizzata una schermata di vittoria (Fig. 4) che occupa l'intera finestra. Le uniche azioni possibili a questo punto sono ricominciare la partita premendo il tasto <ENTER > o uscire dal programma premendo il tasto <ESC >.

Figure 4: Schermata di vittoria

4 Strutture e costanti globali

Nel file cannon.h incluso nel progetto è definita la struttura thread_param, che contiene i parametri utilizzati nei vari thread, tra cui il periodo, la priorità, la deadline relativa e quella assoluta. Altre strutture definite sono cams che contiene i parametri di posizione e orientazione della visuale e tar che contiene i parametri di posizione, velocità e dimensione dei target. Il file contiene, tra le altre cose, anche le definizioni di molte costanti globali usate nel programma, che possono essere modificate con semplicità e costituiscono soprattutto parametri relativi alla fisica del problema (ad esempio massa del proiettile e costanti fisiche) e valori limite delle variabili.

5 Flusso di programma e thread

All'avvio del programma vengono inizializzati allegro e allegroGL e installati i gestori di tastiera, mouse e timer. Inoltre vengono inizializzate le variabili globali.

Successivamente tramite la funzione rset() vengono inizializzate alcune variabili a valori casuali. A questo punto vengono definiti i parametri e creati i thread che gestiscono il moto dei due target e il thread che gestisce gli input.

Infine vengono definiti i parametri e viene creato il thread che esegue il rendering dell'ambiente 3D e il programma principale va in attesa fino a che quest'ultimo thread non termina. Una volta che ciò accade, anche il programma principale termina.

I thread sono descritti più nel dettaglio nei paragrafi seguenti. Il programma adopera la politica di scheduling di default.

5.1 rendering

Il thread di rendering si occupa di creare l'interfaccia grafica descritta precedentemente e di aggiornarla per mostrare a schermo i cambiamenti di posizione dei vari componenti in relazione anche allo spostamento della visuale.

In particolare, è un thread periodico di periodo 20ms e priorità 25.

La sua funzione rendering() inizialmente imposta i parametri del modo grafico e crea la finestra. Poi entra in un ciclo while in cui vengono chiamate le altre funzioni per costruire all'interno dell'ambiente le strutture 3D necessarie.

Nello specifico:

- create_ground() crea un rettangolo di colore verde scuro (ground) e un rettangolo di dimensioni ridotte e colore verde chiaro (playground);
- create_wall() crea il muro, ovvero un parallelepipedo rosso (formato da 5 rettangoli) in una data posizione e con una data altezza;
- create_target() crea i bersagli, ovvero due quadrati ad altezza del playground. La posizione, il colore e le dimensioni sono descritti nel vettore strutture di tipo tar, tar_vec[];
- create_cannon(), tramite la funzione cylinder() crea un cilindro di raggio di base e altezza costanti, colore grigio chiaro e posizione e inclinazioni specificate tra gli argomenti.
- create_projectile() tramite la funzione sphere() crea una sfera di raggio definito tra le costanti e colore grigio scuro centrata nelle coordinate specificate negli argomenti;
- create_wvane() crea unasta verticale nera e, se il vento attivo, un triangolo di colore rosso direzionato nella stessa direzione del vento e di lunghezza proporzionale alla velocità del vento;
- se il mirino è attivo, trajectory() calcola il punto di atterraggio del proiettile e in tale punto crea tramite la funzione sphere una piccola sfera arancione (chiamata "ghost" all'interno del programma). Tale posizione viene resa imprecisa aggiungendo alle coordinate del punto di atterraggio un rumore casuale;

Successivamente alla chiamata di queste funzioni, il programma entra in modalità allegro per definire strutture 2d. tramite le funzioni di allegro disegna il menù nella parte inferiore della finestra. Una volta fatto ciò, il programma esce dalla modalità allegro.

Se è stata raggiunta la condizione di vittoria (win = 1), al posto delle funzioni sopra elencate viene solamente disegnata sull'intera finestra la schermata di vittoria in maniera analoga a quanto accade per il menù.

Tramite le funzioni glflush() e allegro_gl_flip() il buffer di visualizzazione viene scambiato (double buffering). Infine tramite la funzione adjcamera() viene aggiornata la visuale e le strutture venono visualizzate nella finestra.

La condizione di uscita dal ciclo è la pressione del tasto <ESC >. Quando ciò accade, il thread termina e l'esecuzione torna al programma principale.

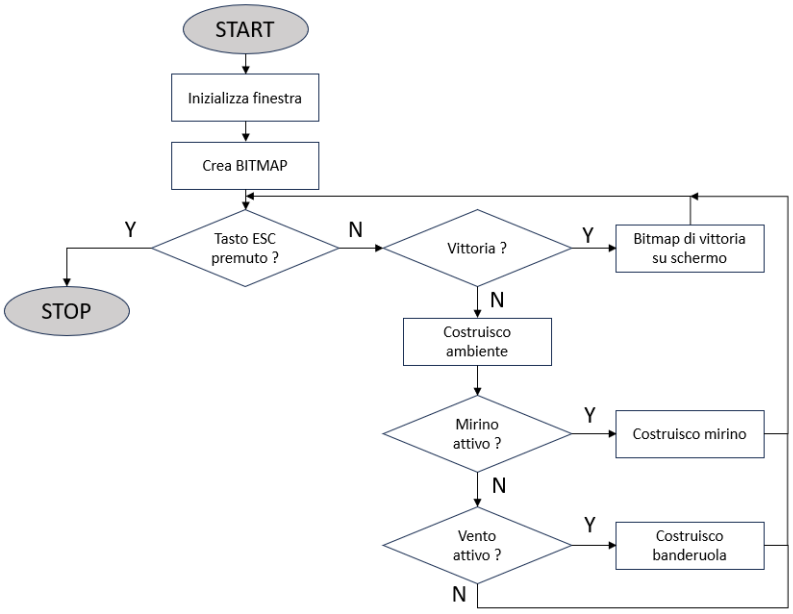

Figure 5: Diagramma di flusso del thread di rendering

5.2 Movimento dei target

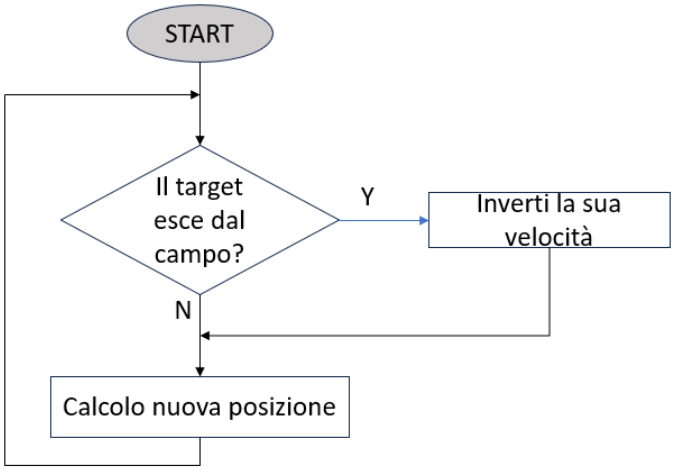

Figure 6: Diagramma di flusso del thread di move_target

I thread move_target si occupano di modificare la posizione dei target all'interno dellarea target.

Entrambi i thread hanno un periodo di 20ms e una priorità di 10, tale periodo permette di avere un movimento fluido degli elementi.

Vengono creati due thread utilizzando la stessa funzione move_target.

In questa funzione è presente un ciclo while nel quale prima viene controllato se il moto del target lo porterebbe fuori dallarea target. In tal caso la velocità in direzione di uscita viene cambiata in segno per simulare un rimbalzo.

In seguito viene aggiornata la posizione del target.

5.3 Gestione degli input

Il thread di input si occupa di controllare se sono premuti tasti della tastiera o del mouse e agire di conseguenza. Il thread è periodico di periodo 30 ms e priorità 20. La sua funzione input_ex() entra in un ciclo while e svolge le seguenti operazioni:

- salva la posizione del cursore nelle variabili prevx e prevy. alla pressione del tasto sinistro del mouse calcola la variazione tra la posizione attuale del mouse e quella salvata nelle variabili precedenti. Gli angoli di rotazione della visuale vengono calcolati in maniera proporzionale a questo spostamento;
- se il tasto <Q> (<E>) è premuto la visuale trasla verso l'alto (basso);
- se il tasto <W> (<S>) è premuto la visuale trasla in avanti (indietro);
- se il tasto <A> (<D>) è premuto la visuale trasla verso sinistra (destra);
- la pressione del tasto <T> attiva (o se già attivo disattiva) il mirino. Per evitare che una pressione prolungata del tasto <T> attivi/disattivi più volte il mirino, un ciclo while vuoto blocca il thread finchè il tasto non viene rilasciato;
- la pressione del tasto <V> attiva il vento e calcola un valore casuale di velocità del vento nelle due direzioni x e z. Se il vento è già attivo la pressione di <V> disattiva il vento ponendo a 0 le sue velocità lungo

x e z. Per evitare che una pressione troppo prolungata del tasto <V> attivi/disattivi più volte il vento, un ciclo while vuoto blocca il thread finchè il tasto non viene rilasciato;

- la pressione del tasto <M> aumenta la potenza del cannone. Questo avviene solo se non c'è uno sparo in corso e se non si è raggiunto il limite massimo di potenza. Per evitare che una pressione troppo prolungata del tasto <M> aumenti troppo rapidamente la potenza, un ciclo while vuoto blocca il thread finchè il tasto non viene rilasciato;

- la pressione del tasto <N> diminuisce la potenza del cannone. Questo avviene solo se non c'è uno sparo in corso e se non si è raggiunto il limite minimo di potenza. Per evitare che una pressione troppo prolungata del tasto <N> diminuisca troppo rapidamente la potenza, un ciclo while vuoto blocca il thread finchè il tasto non viene rilasciato;

- se il tasto <UP> (<DOWN>) è premuto, l'angolo del cannone rispetto allasse x aumenta (diminuisce), questo avviene fino a che non si è raggiunto il valore massimo (minimo);

- se il tasto <SX> (<DX>) è premuto, il cannone si sposta lungo il lato dell'area cannone opposta al muro, questo avviene fino a che non si è raggiunta la posizione limite;

- se il tasto <SPACE> viene premuto, il programma viene attivato lo sparo (shoot = 1) vengono impostati i parametri del thread cannon e viene creato tale thread. Per evitare che una pressione troppo prolungata del tasto <SPACE> spari più proiettili in una volta, un ciclo while vuoto blocca il thread finchè il tasto non viene rilasciato.

I segnali di input appena elencati vengono accettati solo se non c'è uno sparo in corso (shoot = 0) in modo da evitare problemi nell'aggiornamento delle variabili globali in caso di più spari consecutivi.

Dopo aver scansionato la tastiera per i tasti premuti, nel caso non sia stato eseguito uno sparo, il proiettile viene posizionato all'interno del cannone.

Se il programma è in stato di vittoria (win = 1) le considerazioni appena fatte non sono più valide e viene controllato solo se il tasto <ENTER> viene premuto. In tal caso il punteggio viene azzerato e si esce dalla condizione di vittoria. In questo modo si inizia una nuova partita.

5.4 Sparo del proiettile

Il thread cannon si occupa di gestire la traiettoria percorsa dal proiettile una volta sparato.

Il thread è periodico di periodo 20 ms con priorità 15 e viene attivato dal thread input. La sua funzione cannon() entra in un ciclo while attivo fino a che il proiettile non ha raggiunto il suolo.

All'interno del ciclo viene eseguito un controllo:se il proiettile è uscito dal playground o ha colpito il muro, lo sparo termina, altrimenti posizione e velocità del proiettile vengono aggiornate come indicato nel modello fisico.

Se il proiettile ha raggiunto il suolo, lo sparo termina e viene controllato se ha colpito un bersaglio, in tal caso il punteggio viene aggiornato. In seguito viene controllato se il punteggio ha raggiunto il massimo e in tal caso il programma entra in stato di vittoria (win = 1).

Quando uno sparo termina viene chiamata la funzione rset() che si occupa di assegnare casualmente una nuova altezza e posizione del muro e nuove posizioni iniziali e direzioni di velocità casuali per i due target. Inoltre, se il vento è attivo viene calcolata una nuova direzione e velocità casuali del vento.

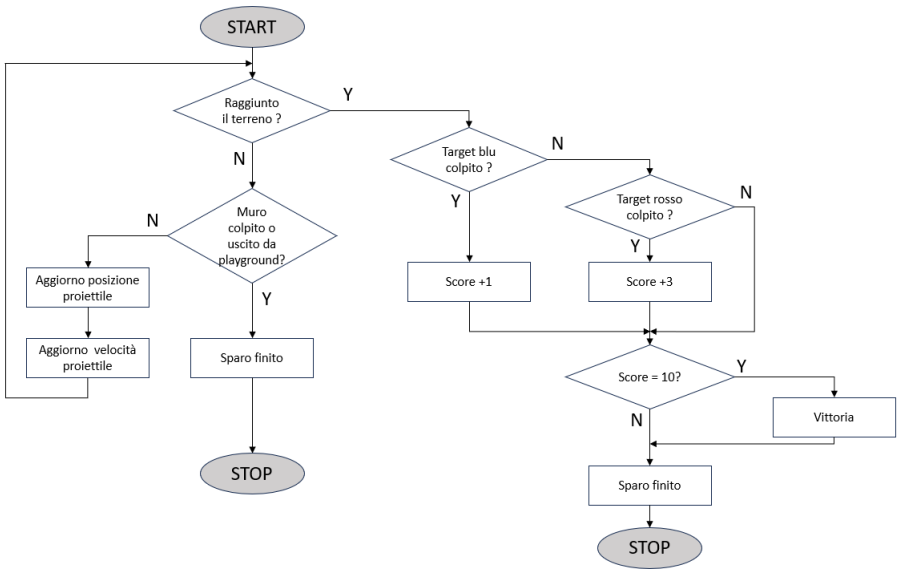

Figure 7: Diagramma di flusso del thread di sparo

6 Altri dettagli del progetto

6.1 Funzioni di gestione del tempo

Per la gestione del tempo e lo scheduling dei thread sono state usate alcune funzioni ausiliarie che lavorano su variabili di tipo timespec (definita in time.h) e thread_param:

- time_copy: copia una variabile di tipo timespec in un'altra;
- time_add_ms: aggiunge ad una variabile di tipo timespec una quantità in ms;
- time_cmp: confronta due variabili tempo;
- set_period: legge il tempo corrente e calcola il prossimo tempo di attivazione e la deadline assoluta del task;
- wait_for_period: sospende il task chiamante fino alla successiva attivazione, calcolando i nuovi tempi di attivazione e deadline;
- deadline_miss: incrementa il campo dl_miss della struttura thread_param in caso di deadline miss.

6.2 Meccanismi di Protezione

I diversi thread hanno una forte interazione tra di loro in quanto avviene una continua comunicazione mediante le variabili globali tra le funzioni che aggiornano le posizioni degli oggetti e la funzioni che esegue il rendering di tali elementi. Inoltre più funzioni possono modificare le stesse variabili globali. Si è reso quindi necessario implementale un meccanismo di protezione. Tale meccanismo è stato realizzato mediante un mutex che si blocca quando una funzione deve modificare o leggere una variabile globale.

Il programma prevede inoltre altri meccanismi di protezione come i controlli sulle funzioni allegro_init e install_allegro_gl. In caso di errori in questa fase il programma viene interrotto e restituisce un messaggio di errore. Inoltre viene eseguito un controllo sulla funzione set_gfx_mode per verificare la corretta inizializzazione dell'interfaccia grafica. In caso di errori viene visualizzato un messaggio.