Progetto d’esame computer graphics: primo semestre

Simulatore di mira per videogiochi sparatutto: AimTrainer

Studenti: Marco Bertoldi Matricola: S4525875

Riccardo Vigna Matricola: S4512729

I videogiochi della categoria sparatutto (FPS), al momento, sono tra i più diffusi. Seppur differenti tra di loro (per ambientazione, strategia, meccaniche di gioco, grafica etc.), hanno un aspetto in comune, che consiste nel colpire un bersaglio, solitamente in movimento. Inoltre sono giocati quasi sempre in modalità multiplayer, quindi i giocatori si affrontano tra loro online. Ad alto livello, gli sparatutto sono principalmente giocati usando mouse e tastiera, perciò ci concentreremo sullo sviluppo in questo settore. Uno degli aspetti chiave per vincere uno scontro, è quello di avere la mira migliore del proprio avversario, ma come si può migliorare la propria mira?? La risposta è molto semplice, con l’allenamento. Un metodo ovvio è quello di giocare molto, così da affrontare molti nemici e allenarsi di conseguenza, ma all’interno di una partita vi sono numerosi tempi morti, in cui non si è in combattimento. Perciò per allenarsi e migliorare più velocemente, risulta più adatto un simulatore di mira.

IL SIMULATORE:

Il simulatore è composto da vari esercizi, ognuno indirizzato a migliorare uno specifico aspetto. Alcuni esempi di esercizi possono essere:

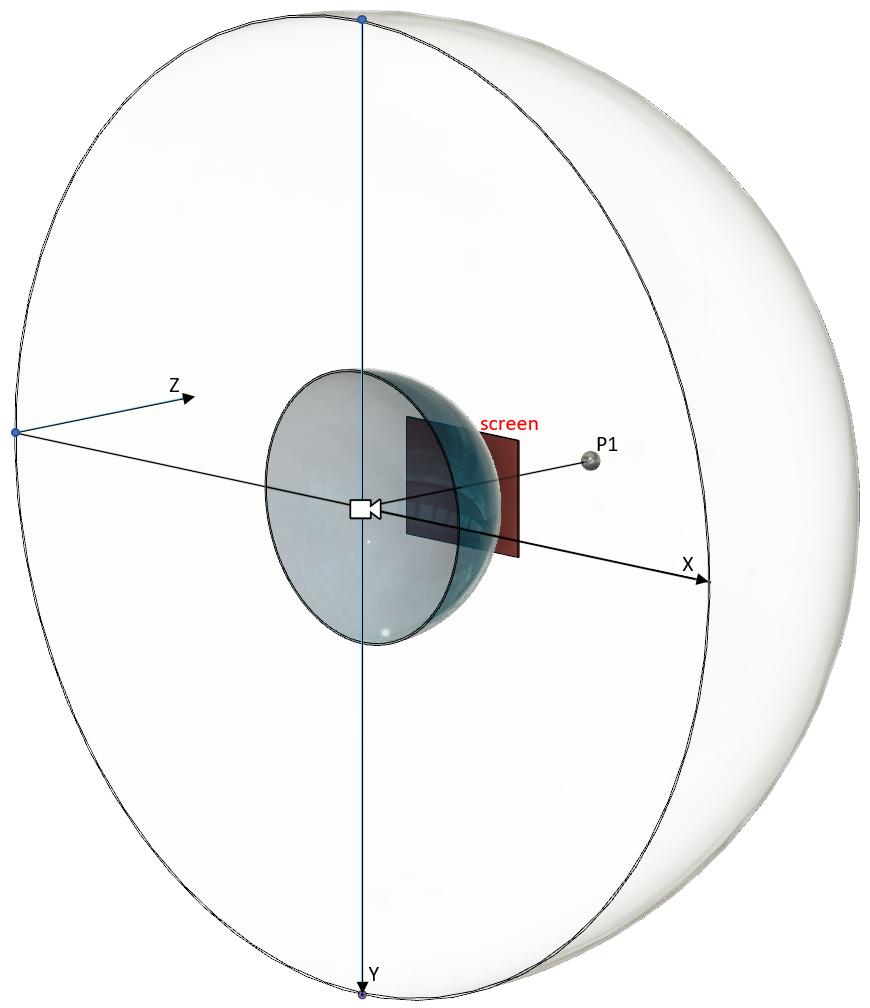
1. seguire un bersaglio in movimento, allenando la precisione nei piccoli spostamenti del mouse e i riflessi ai cambi di direzione.
2. colpire bersagli che appaiono e scompaiono in determinati punti dello schermo, così da migliorare la reattività alla apparizione di un bersaglio e la capacità di eseguire grandi spostamenti del mouse, rapidi e precisi.

Dagli allenamenti si possono ottenere delle statistiche sulla base dei punteggi fatti, osservando eventuali miglioramenti o lacune in determinati esercizi, così da programmare gli allenamenti successivi. Il simulatore può anche essere usato come riscaldamento prima di giocare a un qualsiasi sparatutto, dato che, una volta raggiunto alti livelli, le capacità ottenute tendono a perdersi rapidamente, se non mantenute giorno dopo giorno. Chiaramente tutto questo allenamento è indicato a quei giocatori che, per passione o per lavoro, giocano gli sparatutto ad alto livello o a livello competitivo/torneistico. Ciononostante, è da ricordare l’importanza di tenere allenati i riflessi e, più in generale, il cervello. Difatti studi hanno evidenziato come videogiocatori abituali di FPS e giochi di azione mostrino maggiore percezione dello spazio, migliori riflessi e capacità di tracking (Castel, Pratt, & Drummond, 2005; Dye, Green, & Bavelier, 2009; Feng, Spence, & Pratt, 2007; Green & Bavelier, 2006a, 2006b; Strobach et al., 2012). Miglioramenti delle abilità di gioco si sono riscontrati anche sul breve termine: in seguito ad allenamenti di cinque ore i soggetti hanno mostrato maggiori abilità in una task di tipo tracking (Chen, Chen, & Li,

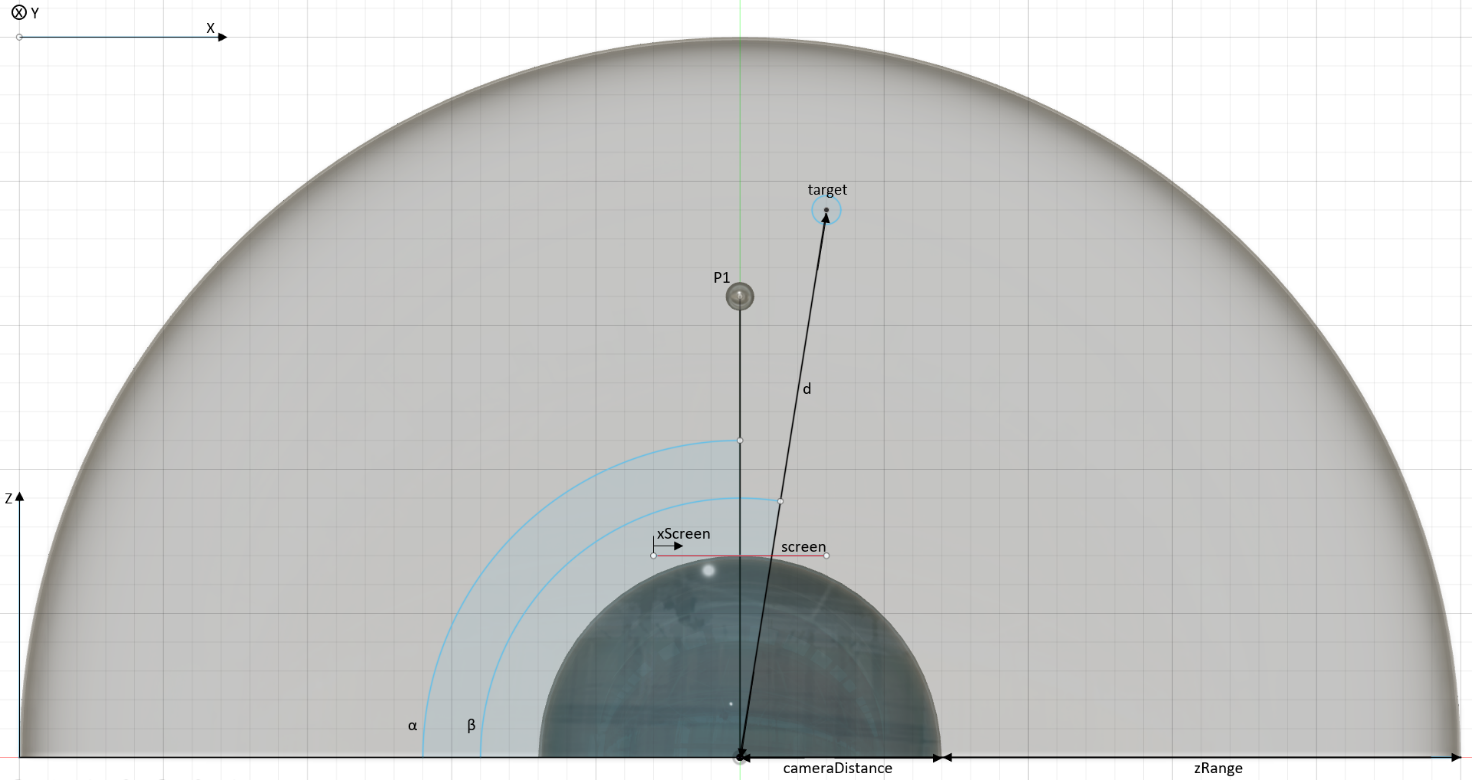
2015). Di recente, tramite prove di laboratorio, si è evidenziato che soggetti allenati sono capaci di movimenti più rapidi, senza intaccare la precisione degli stessi, grazie ad una migliore gestione del meccanismo di retroazione visuo-motorio (Michael A. Rupp, Daniel S. McConnell, Janan A. Smither, 2019). Ad esempio, si potrebbe pensare di usare il simulatore come terapia di riabilitazione di alcune facoltà neurologiche che coinvolgono gli occhi o il cervello, danneggiate o perse in seguito ad un danno o un intervento chirurgico. L’implementazione base del gioco prevede l’inserimento di vari bersagli, che si comportano in modo diverso a seconda dell’esercizio. Vi è poi un puntatore che indica dove si sta mirando. Il resto è composto da grafiche di vario tipo e di un menu per scegliere l’esercizio. La piattaforma di sviluppo sarà monogame, utilizzando una grafica 2D.

AMBIENTE 3D:

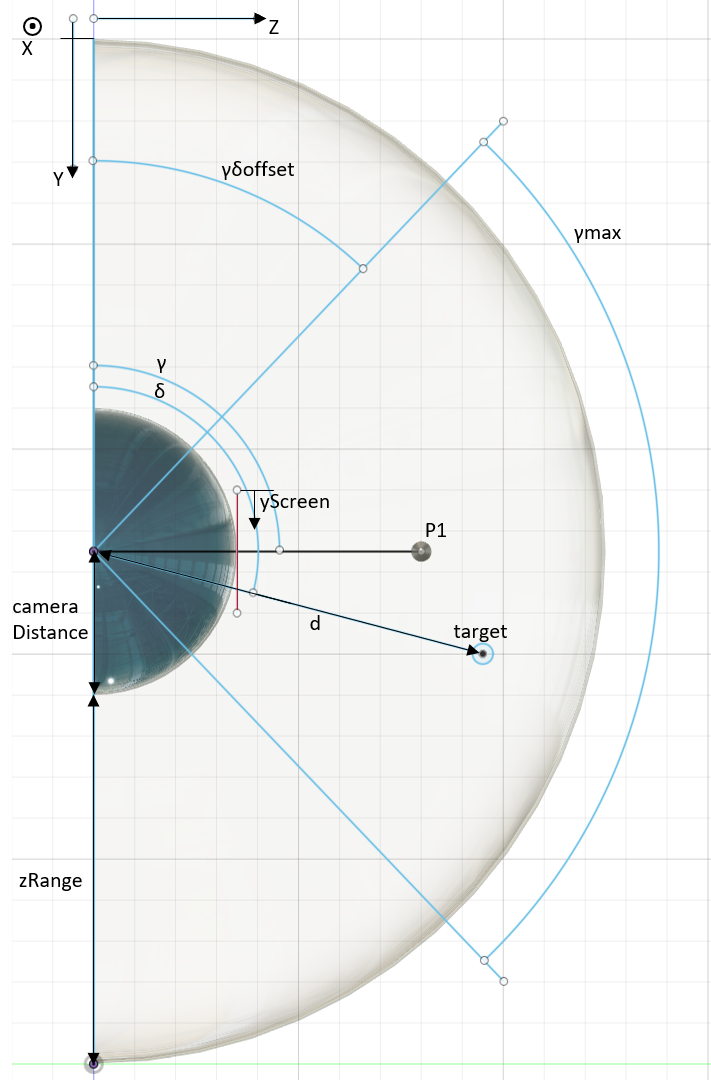
Tramite prove pratiche durante lo sviluppo si è notato come la grafica 2D risulti estremamente limitante dato il fine ultimo del gioco. Giochi sparatutto moderni sono sviluppati in 3D e l’assenza della terza dimensione si percepisce nella schermata e si ripercuote in una mancanza di immersione all’interno della scena. In un simulatore a due dimensioni il giocatore viene proiettato in un ambiente piatto, percepito differente da quello in cui si svolge il gioco per il quale ci si sta allenando, vanificando l’utilità dell’esercizio. Per dare un’illusione di profondità si è pensato allora ad un ambiente tridimensionale all’interno del quale si può muovere il bersaglio, schematizzato di seguito:



La camera è posta al centro delle due semisfere e il bersaglio può muoversi nello spazio libero tra le due semisfere. Segue una vista dall’alto sul piano xz:



E una vista laterale sul piano yz:



Con “screen” si fa riferimento al rettangolo avente dimensioni pari allo schermo del pc, sulle quali verrà proiettata la posizione del target per la stampa a schermo. Il rettangolo screen giace sul piano tangente la semisfera di raggio cameraDistance. L’angolo compreso tra l’asse X e la congiungente tra il centro dello schermo e la posizione della camera è definito come α, scelto come appartenente all’intervallo [0,π]. Analogamente si definisce l’angolo γ come l’angolo compreso tra l’asse delle Y e la congiungente tra centro dello schermo e posizione della camera. Si è scelto di “mappare” lo sfondo sulla semicirconferenza di raggio cameraDistance, segue dunque la considerazione seguente:

L’angolo α viene calcolato come l’ascissa del rettangolo viewSource, libero di muoversi nei limiti dello sfondo, diviso la distanza dal centro della camera, cameraDistance. Essendo viewSource.X libero di muoversi tra 0 e background.Width-screenDim.X compresi e avendo vincolato α all’intervallo [0,π] deve valere

Risulta vincolato il valore di cameraDistance, che viene calcolato come:

Lo stesso ragionamento viene fatto per l’angolo γ, ma con cameraDistance fissato. Si calcola dunque il massimo valore di tale angolo in relazione all’altezza dello sfondo:

Volendo centrare il range di γ attorno al centro della scena, si definisce un offset comune per gli angoli γ e δ:

L’angolo γ viene infine calcolato come:

A causa delle dimensioni dello sfondo, con larghezza ed altezza diverse, γ risulta appartenente all’intervallo [γδoffset, γδoffset + γmax].

Si definisce d la distanza tra la posizione della camera, indicata successivamente come cameraOrig, e il centro del target, realizzato tramite una sfera di raggio radius:

L’angolo β, compreso tra l’asse delle X e la congiungente tra cameraOrig e il centro del target, appartiene all’intervallo [0,π] e viene calcolato come segue:

Analogamente l’angolo δ, compreso tra l’asse delle Y e la congiungente tra cameraOrig e il centro del target, appartiene all’intervallo [γδoffset, γδoffset + γmax] e viene calcolato come segue:

Per dare l’impressione di un movimento lungo l’asse delle Z si realizza uno scalamento dello sprite che rappresenta il target, realizzato tramite una sfera di raggio radius, in base alla distanza d.

Immagine che contiene antenna, armadietto, screenshot

Descrizione generata automaticamente

Lo schermo su cui viene proiettata l’immagine si è ipotizzato essere a distanza fissa cameraDistance. A tale distanza, il raggio del target viene percepito come minore a causa di d e pari al valore apparentRadius, come esemplificato in figura. In particolare si ha:

e segue dunque il fattore di scala per lo sprite:

Il valore scale si nota essere compreso nell’intervallo (0,1], in quanto il target è libero di muoversi tra le due semisfere e non si troverà dunque ad una distanza inferiore a cameraDistance. La gestione dei vincoli spaziali viene discussa in seguito.

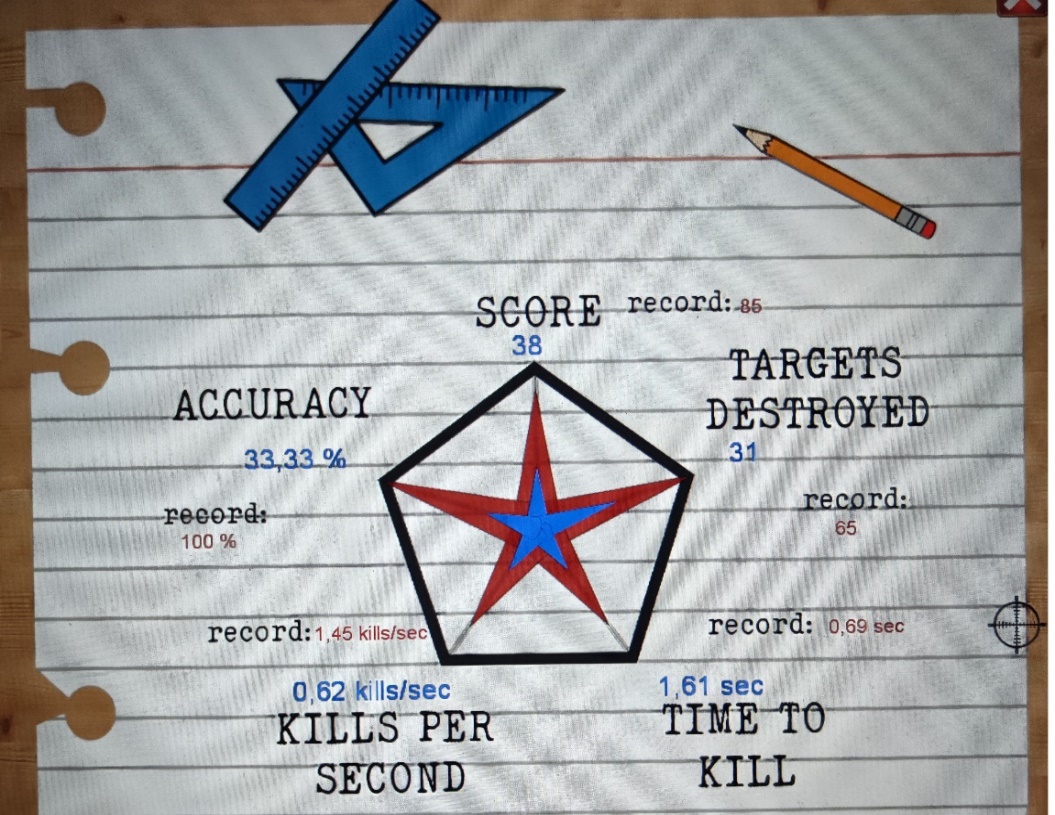
Nel modello 3D esposto precedentemente è stato indicato in rosso un rettangolo che identifica lo schermo del dispositivo su cui l’immagine viene proiettata. La proiezione sullo schermo del target dipende dalla posizione dello schermo, indicata da α e γ, e dalla posizione del target, indicata tramite β e δ. Le coordinate utilizzate per la stampa a schermo dello sprite del target, opportunamente scalato in base alla distanza, si ricavano come segue:

Le coordinate vengono memorizzate in una variabile Point chiamata positionOnScreen. Il punto middleScreen indica le coordinate che identificano il centro dello schermo (facendo riferimento al sistema di coordinate bidimensionale dello schermo).

Lo spostamento del target all’interno dello spazio tridimensionale deve essere vincolato in modo tale da risultare sempre raggiungibile dal puntatore, nonostante i vincoli sullo spostamento dello stesso dettati dalle dimensioni dello sfondo. Lo schermo è libero di muoversi entro i limiti degli angoli α e γ, dunque il target deve essere vincolato a muoversi all’interno dello spazio coperto da questi angoli. Ulteriore vincolo è costituito dal valore di d, che non deve mai essere inferiore a cameraDistance o superiore a cameraDistance+zRange. Devono essere dunque verificate le seguenti condizioni:

Nel caso in cui una delle precedenti venisse violata, per semplicità di soluzione si è scelto di far convergere il target in direzione del punto P1, posto al centro della scena. Lo spostamento verso P1 avviene con continuità o a scatti, in base alla modalità di gioco. Non è garantito che da qualsiasi punto nella scena il tragitto verso P1 non intersechi la semisfera interna di raggio cameraDistance. Si è scelto di ignorare questo difetto in quanto si ritiene che non comprometta l’azione di gioco, favorendo una maggiore continuità e fluidità dello stesso.

RISULTATI:

Terminato il gioco appaiono le statistiche, nella schermata seguente:

Vengono mostrate sia le statistiche della partita appena terminata (in blu), sia il migliore risultato ottenuto (in rosso). In questo modo si può fare il confronto, verificando miglioramenti o peggioramenti nella propria prestazione.