UNIVERSITA' DEGLI STUDI DI BARI ALDO MORO

DIPARTIMENTO DI INFORMATICA

Corso di Laurea in Informatica

TESI DI LAUREA IN

Sviluppo di Videogiochi

A Mind Within You: integrazione di intelligenza artificiale generativa in un videogioco action-adventure a scorrimento 2D

Relatore

Prof. Pierpaolo Basile

Laureando

Ivan Digioia

ANNO ACCADEMICO 2024-2025

INDICE

1.INTRODUZIONE

Lo sviluppo dei videogiochi ha sempre dovuto confrontarsi con i limiti tecnologici imposti dal periodo storico di riferimento. Dalle prime esperienze interattive degli anni ‘70, caratterizzate da grafica minimale e capacità di calcolo estremamente limitate (come *Pong* e *Space Invader*), fino alle produzioni moderne che sfruttano avanzate tecnologie di intelligenza artificiale e grafica fotorealistica.

Nei primi decenni, la principale limitazione era la potenza di calcolo, che impediva di realizzare mondi complessi e interattivi. I videogiochi erano progettati con regole rigide e schemi ripetitivi, in cui gli NPC (personaggi non giocanti) seguivano comportamenti predefiniti e prevedibili. Con l’avvento delle console a 16 e 32 bit, le capacità grafiche e computazionali si ampliarono, consentendo la creazione di ambientazioni più dettagliate e sistemi di gioco più articolati[1]. Titoli come *The Legend of Zelda: A Link to the Past* introdussero mappe più vaste e interconnesse, mentre giochi come *Final Fantasy VI* iniziarono a sperimentare con trame complesse e meccaniche di gioco più profonde.

L’evoluzione successiva fu segnata dall’introduzione della grafica tridimensionale, resa possibile dall’aumento della potenza di calcolo delle console e dei PC. Durante gli anni ‘90, titoli come *Doom*[2] contribuirono a questa transizione, anche se utilizzavano una tecnica di rendering basata su **raycasting**[3], che simulava una prospettiva tridimensionale pur operando su una mappa bidimensionale. Con l’arrivo di giochi come *Super Mario 64* il vero 3D poligonale divenne lo standard del settore, trasformando radicalmente il level design e il gameplay.

L’avvento dell’intelligenza artificiale seppure nelle sue forme più grezze ha rappresentato una delle innovazioni più significative nell’industria videoludica, introducendo inizialmente degli script statici e delle routine predefinite. L’evoluzione dell’IA in quegli anni è stata graduale, in cui i sistemi a stati piano paino diventavano sempre più articolati e complessi, tanto che nei primi anni 2000 l'evoluzione delle IA è virata non più verso la complessità di tali sistemi ma verso tecniche che rendessero i nemici più dinamici e reattivi. In giochi come *F.E.A.R.*, gli NPC dimostravano una sorprendente capacità di adattamento, utilizzando tattiche di copertura, aggiramento e coordinazione tra compagni di squadra. Questo segnò un passo avanti rispetto ai nemici con schemi di attacco prevedibili, rendendo il gameplay più coinvolgente e sconvolgendo la critica videoludica.

Negli anni più recenti, le reti neurali e le tecniche di apprendimento automatico hanno ulteriormente ampliato le possibilità dell’IA nei videogiochi. Titoli come *The Last of Us Part II* mostrano NPC in grado di comunicare tra loro, reagire in modo contestuale agli eventi e cambiare strategia in base al comportamento del giocatore. Inoltre, gli strumenti basati su IA vengono sempre più utilizzati nello sviluppo per generare ambienti, animazioni e persino linee di dialogo procedurali. Un esempio è il *Generative Quest System* di *AI Dungeon*, che utilizza modelli di linguaggio avanzati per creare storie uniche e interattive basate sulle scelte del giocatore.

L'intelligenza artificiale non si limita più alla gestione degli NPC: sempre più giochi utilizzano algoritmi generativi capaci di adattarsi per offrire la miglior esperienza possibile al giocatore, offrendo livelli di difficoltà dinamici, missioni personalizzate e persino mondi generati proceduralmente. Questo è evidente in giochi come *No Man's Sky*, dove l'universo esplorabile è creato da algoritmi di generazione procedurale, garantendo un'esperienza unica a ogni giocatore.

1.1 Obiettivi

Questa tesi di laurea si concentra sull'implementazione di un'interfaccia che permetta al giocatore di conversare con un'intelligenza artificiale già addestrata, ma arricchita con un background contestualizzato all'universo narrativo del gioco. L'obiettivo è analizzare come l'IA interagisca con il giocatore, verificando se sia in grado di mantenere la sua attenzione e di adattarsi dinamicamente alle sue risposte, creando un'esperienza unica e personale.

Nonostante l'intelligenza artificiale sia uno degli aspetti fondamentali dei videogiochi, la sua evoluzione è stata più lenta o comunque meno costante rispetto ad altri elementi come la grafica, il level design e la narrativa. Questa ricerca esplorerà i motivi di questa disparità, identificando le principali sfide tecniche e concettuali nell'implementazione di un'IA realmente reattiva e contestualizzata.

Nelle prime fasi analizzeremo una selezione di giochi che hanno innovato l'uso dell'IA nella gestione degli NPC del mondo di gioco creato attorno a loro. Studieremo in che modo titoli famosi come *Metal Gear Solid*, *F.E.A.R.* e *Alien: Isolation* abbiano sfruttato dei sistemi altamente avanzati per creare delle interazioni più credibili e di come questi siano riusciti a dimostrare quanto margine di progresso è ancora possibile ottenere innovando in questo campo.

Successivamente, ci concentreremo sulla progettazione e implementazione dell'IA nella demo di "*A Mind Within You"*, esaminando gli aspetti tecnici e metodologici del sistema. Verranno analizzate le strategie adottate per rendere la conversazione più immersiva e credibile, nonché le sfide incontrate durante lo sviluppo.

2. ANALISI E STATO DELL'ARTE

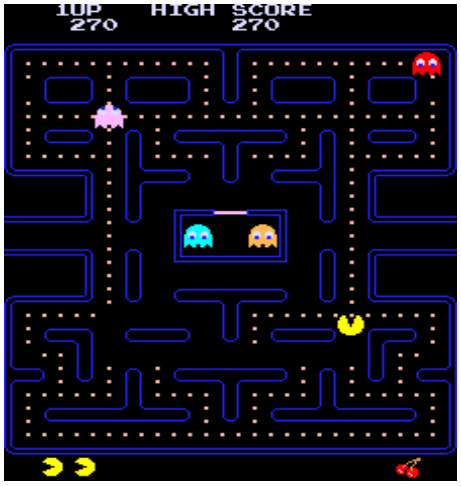
In questo capitolo analizzeremo come l'intelligenza artificiale è stata implementata e sviluppata nei videogiochi, evidenziando i titoli che si sono distinti per le loro innovazioni. Inizieremo con alcuni esempi di giochi che hanno introdotto soluzioni avanzate nell'IA, per poi approfondire i casi più significativi, ossia quei titoli che hanno ridefinito il modo in cui l'IA può influenzare il gameplay e l'esperienza del giocatore.

Attraverso questa analisi, sarà possibile comprendere come l'IA sia stata sfruttata nei videogiochi, quali siano stati i suoi sviluppi più rilevanti e come questa si è sviluppata negli anni. Questo ci permetterà di individuare le sue applicazioni più efficaci e di esplorare nuove possibilità per rendere l'interazione con il giocatore sempre più immersiva e dinamica.

2.1 Dalle origini alle ultime innovazioni

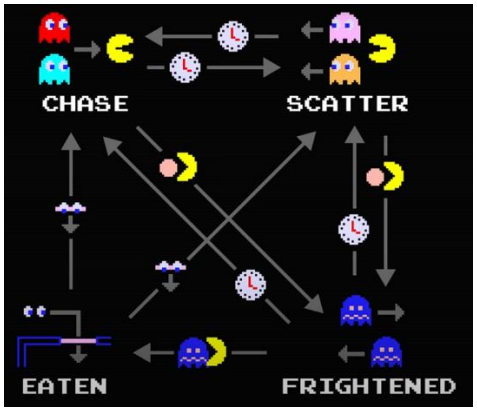
Nei primi videogiochi, l'IA era basata su semplici routine di comportamento, spesso costruite con regole predefinite e pattern statici. Titoli iconici come *Pac-Man* (1980) utilizzavano IA basate su schemi ripetitivi per determinare i movimenti dei fantasmi, mentre giochi come *Doom* (1993) introducevano nemici con reazioni predeterminate agli input del giocatore.

2.1.1 Esempi di videogiochi che evolvono l'intelligenza artificiale



Schermata di gioco di Pac-Man (1980)

Prima di *Pac-Man*[4] (1980) ideato in giappone e prodotto dalla Namco in formato Arcade, alcuni videogiochi avevano già introdotto forme rudimentali di intelligenza artificiale, come *Pong* (1972) con l'IA della racchetta avversaria, *Computer Space* (1971) con il comportamento nemico, *Space Invaders* (1978) con il progressivo aumento di velocità degli alieni e *Adventure* (1979) con i movimenti autonomi dei draghi.



Fasi dei fantasmi in Pac-Man

L'elemento innovativo di Pac-Man era il comportamento distinto di ciascun fantasma. **Blinky (rosso)** inseguiva direttamente Pac-Man, mentre **Pinky (rosa)** cercava di anticipare i suoi movimenti spostandosi quattro celle avanti rispetto alla sua direzione per anticiparlo e catturarlo da sola o con altri fantasmi. **Inky (ciano)** adotta come strategia quella di bloccare il tunnel più vicino a Pac-Man intercettandolo dalla parte opposta ed è il fantasma con lo schema di movimento più complicato, ed esegue molti dei suoi movimenti in sincronia con Blinky. Infine, **Clyde (arancione)** è il più lento dei quattro e non insegue mai Pac-Man a meno che non sia già inseguito da un altro fantasma.

Queste dinamiche resero Pac-Man uno dei primi giochi a introdurre un'intelligenza artificiale più articolata, in cui i nemici non si muovevano più in modo casuale ma rispondevano strategicamente alle azioni del giocatore, aumentando il livello di sfida e rigiocabilità.

Pochi giochi nella successiva decade hanno spinto l'innovazione nell'IA dopo Pac-Man e bisognerà aspettare fino al 1993 con l'uscita di Doom e il suo sistema a fasi ma sarà Metal Gear Solid nel 1998 a segnare il futuro delle IA degli anni successivi.

Copertina di Metal Gear Solid (1998)

Metal Gear Solid[5], pubblicato nel 1998 da Konami, diede vita al genere stealth, introducendo meccaniche di comportamento avanzate per i nemici che hanno reso il gioco più realistico e coinvolgente, grazie anche alle capacità hardware della PlayStation.

L'IA dei nemici in *Metal Gear Solid* si basa su una combinazione di percezione visiva e uditiva, creando un sistema di allerta a più fasi.

Nella **fase di pattugliamento (Idle)**, i nemici seguono percorsi predefiniti, sorvegliando determinate aree e mantenendo un comportamento prevedibile. Tuttavia, se il giocatore si avvicina troppo, fa rumore (ad esempio camminando su superfici metalliche o sparando) o lascia tracce visibili, come impronte nella neve, le guardie entrano nella **fase di sospetto (Caution)**. In questo stato, i nemici interrompono il loro percorso per investigare, guardandosi attorno o seguendo le tracce lasciate dal giocatore.

Schermata di gioco di Metal Gear Solid

Se, durante questa fase, il nemico conferma la presenza del giocatore, si passa alla **fase di allerta (Alert)**. In questo stato, le guardie attivano un allarme, chiamando rinforzi. In questo momento, il giocatore deve agire rapidamente per nascondersi o affrontare il combattimento.

Nel caso in cui il giocatore riesca a sfuggire alla vista dei nemici, questi entrano nella **fase di evasione (Evasion)**. In questa fase, le guardie continueranno a cercarlo per un certo periodo, perlustrando l'area, controllando angoli, armadietti e altri nascondigli in cui il giocatore potrebbe essersi nascosto. Se dopo un periodo di ricerca il giocatore non viene trovato, le guardie ritornano alla **fase normale (Normal)**, riprendendo il loro percorso di pattugliamento. Tuttavia, rimarranno più sospettose e attente ai movimenti, anche dopo che il pericolo immediato è passato.

Un'altra innovazione importante è il **sistema di percezione sonora**, che permette ai nemici di sentire i passi del giocatore se corre su superfici rumorose (come metallo o pozzanghere). Per questo motivo, il giocatore può distrarre le guardie lanciando oggetti o sparando a pareti e tubi per attirare la loro attenzione. I nemici non attaccano immediatamente, ma si avvicinano alla fonte del rumore per investigare.

Oltre ai normali nemici, anche i boss di *Metal Gear Solid* presentano IA uniche, con comportamenti e strategie specifiche, adattate al loro background:

- **Psycho Mantis** legge la memory card del giocatore e disabilita il controller se non si cambia porta.
- **Sniper Wolf** si comporta come un vero cecchino, aspettando il momento giusto per colpire e cambiando posizione dopo ogni sparo.
- **Gray Fox (Ninja Cyborg)** utilizza un'IA avanzata per schivare i proiettili e attaccare con movimenti rapidi.

Tutte queste caratteristiche hanno contribuito a creare un'esperienza di gioco estremamente immersiva. *Metal Gear Solid* ha rappresentato una pietra miliare nel mondo videoludico, introducendo un sistema di guardie con percezione visiva e uditiva credibile, fasi di allerta realistiche e reazioni dinamiche all'ambiente. Questi elementi hanno reso il gameplay più tattico e coinvolgente, influenzando molti giochi stealth successivi, come *Splinter Cell*, *Hitman* e tanti altri ancora.

Copertina di F.E.A.R. (2005)

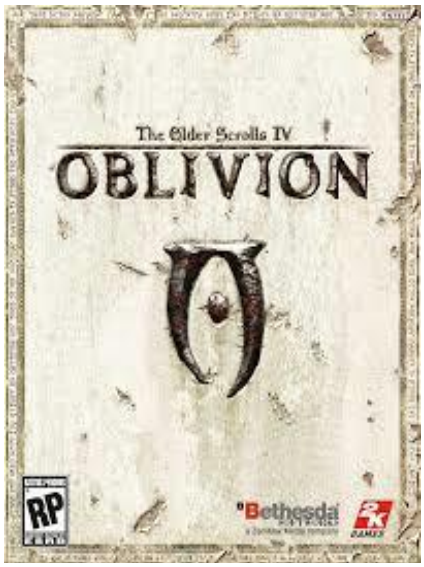
Quando ***F.E.A.R.***[6][7][8] fu rilasciato nel 2005 dalla casa di sviluppo Monolith Productions, uno degli aspetti che più impressionò giocatori e critici fu la qualità dell'intelligenza artificiale dei nemici. A differenza di molti giochi precedenti, in cui gli avversari seguivano pattern rigidi e predefiniti (come Idle → Alert → Attack) *F.E.A.R.* introdusse un sistema di IA dinamica.

L'IA di *F.E.A.R.* si basava su un'architettura decentralizzata chiamata **GOAP (Goal-Oriented Action Planning)**. Questo sistema permetteva agli NPC di prendere decisioni in base al contesto, scegliendo tra diverse azioni possibili per raggiungere un obiettivo. GOAP differisce dagli approcci tradizionali perché invece di seguire transizioni fisse tra stati, gli NPC ragionano su quale sia l'azione migliore in base alla situazione corrente.

Esempio pratico di decisione con GOAP: Il nemico si accorge del giocatore. Il sistema determina che l'obiettivo è "sopravvivere" e "neutralizzare la minaccia". Sceglie l'azione migliore tra varie possibilità: Se è vicino a una copertura, ci si ripara. Se è in inferiorità numerica, chiama rinforzi. Se ha una granata, cerca di lanciarla per stanare il giocatore. Se ha una via di fuga, può ritirarsi e riorganizzarsi.

A differenza dei classici giochi dove i nemici avanzavano direttamente verso il giocatore, in F.E.A.R. essi usavano tattiche militari reali. Potevano: dividersi in gruppi per accerchiare il giocatore, muoversi da una copertura all'altra invece di restare fermi dietro un muro oppure cooperare tra loro, oppure un NPC che attirava l'attenzione del giocatore mentre un altro cercava di avvicinarsi. Oltre a questo si migliorò l'uso e lo sfruttamento degli ambienti a proprio favore: gli NPC potevano abbattere porte e finestre, usando percorsi alternativi, potevano saltare sopra ostacoli o strisciare sotto oggetti per riposizionarsi e usavano granate in modo strategico per stanare il giocatore.

L'IA di F.E.A.R. è ancora oggi considerata uno dei migliori esempi di intelligenza artificiale nei videogiochi. Il suo sistema GOAP è stato ripreso in diversi giochi successivi, come Killzone 2 e Shadow of Mordor.



Quando The Elder Scrolls IV: Oblivion[9] uscì nel 2006, prodotto da Bethesda, una delle sue innovazioni più ambiziose fu il sistema Radiant AI, che prometteva di rendere il mondo di gioco più vivo e reattivo che mai. L'idea era quella di creare NPC che non seguissero semplicemente routine predefinite, ma che prendessero decisioni dinamiche in base all'ambiente, ai bisogni e alle interazioni del giocatore.

Copertina di the Elder Scrolls IV; Oblivion (2006)

Il **Radiant AI**[10][11][12] di *Oblivion* era basato su un sistema di **obiettivi e necessità** dove ogni NPC aveva una serie di bisogni e desideri che determinavano le sue azioni quotidiane con obiettivi specifici, come mangiare, dormire, lavorare, allenarsi o socializzare. Questi obiettivi non erano fissi, ma variavano in base alle condizioni ambientali e alle azioni del giocatore.

Se un NPC aveva fame, poteva scegliere tra diverse opzioni per soddisfare il bisogno: Se aveva cibo nell'inventario, lo mangiava. Se non aveva cibo, provava a comprarlo. Se era povero ed era incline ai furti, poteva rubare da un negozio o da un altro NPC.

Schermata di gioco di The Elder Scrolls IV

Gli NPC reagivano in modo credibile agli eventi. Se vedevano un furto, potevano allertare le guardie. Se erano testimoni di un omicidio, potevano fuggire o cercare aiuto. E ognuno aveva una personalità unica basata su parametri come aggressività, onestà e coraggio, che influenzavano il loro comportamento.

Rispetto a suoi predecessori come Morrowind (2002), gli NPC erano quasi immobili, limitandosi a stare in un luogo specifico senza interagire tra loro. In Oblivion, invece, potevano muoversi liberamente, cambiare attività e prendere decisioni autonome.

In giochi come *Gothic* (2001), gli NPC avevano routine più avanzate rispetto a *Morrowind*, ma in *Oblivion* si andò oltre, introducendo la capacità di prendere decisioni in base alle risorse disponibili, creando situazioni in cui più NPC combattono per impadronirsi di risorse o difendere un territorio.

Il Radiant AI non solo migliorava l'immersione, ma influenzava anche il gameplay: un NPC chiave poteva essere ucciso accidentalmente da un altro NPC, costringendo il giocatore a trovare soluzioni alternative, un personaggio poteva rubare un oggetto importante prima che il giocatore lo prendesse, creando situazioni inaspettate oppure alcuni NPC potevano diventare ostili o amichevoli in base alle azioni del giocatore.

Nonostante fosse rivoluzionario, il Radiant AI aveva anche alcuni problemi: **Comportamenti caotici:** Nei primi sviluppi del gioco, gli NPC rubavano cibo dai negozi per sopravvivere, causando il caos nelle città. Questo portò Bethesda a limitare alcune libertà per evitare problemi di bilanciamento. **Ripetitività:** Anche se gli NPC prendevano decisioni dinamiche, alla lunga le loro azioni potevano risultare prevedibili. **Bug e glitch:** A volte gli NPC si comportavano in modo strano, come pattugliare un'area senza motivo o rimanere bloccati in loop di animazioni.

Ma nonostante questi problemi ancora oggi l'IA di Oblivion è ricordata come una delle più avanzate dell'epoca, tant'è che il Radiant AI fù riusato e ampliato nella produzione di Skyrim V in cui gli NPC divennero ancora più capaci di interagire e reagire ai cambiamenti del mondo di gioco.

Copertina Alien: Isolation (2014)

Alien: Isolation[13][14][15] (2014), sviluppato da Creative Assembly, è noto per avere una delle IA più sofisticate e spaventose mai viste in un videogioco. Il comportamento dell'**Xenomorfo**, il principale antagonista del gioco, è il fulcro dell'esperienza horror e differisce radicalmente da quello dei nemici basati su pattern prevedibili.

L'IA dell'Alien è costruita su **un sistema a due livelli** che gli permette di **adattarsi in modo dinamico al giocatore**, rendendo ogni incontro imprevedibile e creando una tensione costante. L'IA dell'Alien è suddivisa in due livelli principali, ognuno con un ruolo ben definito:

Livello Alto (Director AI): È il **"regista"** che supervisiona l'azione e determina in quale area l'Alien può muoversi. Esso tiene traccia della posizione del giocatore e decide quando l'Alien può avvicinarsi. Controlla la "pressione" sul giocatore: se è stato inseguito troppo a lungo, l'IA potrebbe far allontanare temporaneamente l'Alien per evitare frustrazione e gestisce il livello di tensione, alternando momenti di quiete a fasi di pericolo crescente.

Livello Basso (Behavioral AI dell'Alien): Questo livello è **indipendente dal giocatore** e regola il comportamento dell'Alien nel momento in cui è presente nella stessa area. L'Alien non segue percorsi predefiniti, ma **caccia dinamicamente**: si muove alla ricerca di suoni e segnali visivi, esplora condotti, si affaccia negli armadietti e guarda sotto i tavoli, può imparare dai comportamenti ripetuti del giocatore.

L'Alien non utilizza il machine learning nel senso moderno (come le reti neurali), ma è stato progettato per **simulare un apprendimento progressivo** attraverso un sistema di decision-making avanzato: se il giocatore usa spesso il lanciafiamme per **respingerlo**, l'Alien potrebbe diventare più **aggressivo** e attaccare immediatamente oppure se il giocatore rimane troppo tempo in un'area o la frequenta troppo spesso, l'Alien potrebbe decidere di pattugliarla più spesso.

Primo incontro con l'Alien

L'Alien utilizza l'ambiente di gioco in modo realistico: si sposta tra **le prese d'aria e i condotti di ventilazione**, scomparendo e riapparendo da punti diversi, può **abbassarsi e guardare sotto i tavoli** se sospetta la presenza del giocatore, può **aprire porte automaticamente** ed è in grado di **evitare trappole**, rendendo inefficaci alcune strategie.

Rispetto a giochi horror precedenti, come *Resident Evil* o *Dead Space*, dove i nemici seguono pattern predefiniti anche dato i lore rispettivi genere più improntati all'Action che all'Horror, l'IA di *Alien: Isolation* è **dinamica** e quindi non esistono percorsi fissi, e ogni incontro è imprevedibile e dettato unicamente dal caso, inoltre essa è **intelligente** ovvero che l'Alien impara dai comportamenti del giocatore e li usa contro di lui e per finire è **persistente** cioè non può essere ucciso, quindi il giocatore deve **sempre** evitarlo o trovare modi per distrarlo, ma non può abusarne altrimenti esso impara e lì usa contro il giocatore.

L'IA di *Alien: Isolation* è ancora oggi ritenuta un capolavoro di design. Il sistema a due livelli crea un nemico **imprevedibile, intelligente e terrificante**, che non segue schemi fissi ma si adatta al giocatore e ancora dopo più di una decade dalla sua uscita l'Alien rappresenta una delle IA più sofisticate mai realizzate in un videogioco. Questa tecnologia ha influenzato altri titoli horror e stealth, ma nessuno è ancora riuscito a replicare completamente il livello di tensione e imprevedibilità.

2.2 Considerazioni sui giochi analizzati

Gli esempi presi in considerazione appartengono a diversi periodi videoludici. Alcuni di questi hanno introdotto nuovi modi per caratterizzare gli NPC, dando loro un background, bisogni o desideri; altri, invece, hanno perfezionato tecniche già esistenti, adattandole meglio al proprio gioco o innovando su tecnologie precedentemente utilizzate.

Tuttavia, questi giochi sono diventati pietre miliari dell'industria non a caso e molti altri titoli che hanno cercato di replicarne il successo hanno fallito. Giochi fortemente narrativi, come *The Last Guardian*[16] *(2016)*, che basavano il gameplay sull'interazione con un compagno NPC, o grandi giochi di ruolo come *Cyberpunk 2077*[17] *(2020)*, il cui obiettivo iniziale era creare un mondo vivo grazie all'IA, sono stati criticati dai giocatori e dalla critica per la scarsa qualità dell'intelligenza artificiale, che spesso comprometteva l'immersione o addirittura rendeva il gioco ingiocabile, danneggiandone l'immagine e portando a flop videoludici.

Tant'è che, ormai da anni, i giocatori hanno distolto lo sguardo dalle grandi aziende con progetti ambiziosi, poiché negli ultimi tempi sono stati i giochi indipendenti[18] a soddisfare maggiormente il pubblico. Con progetti più piccoli e mirati, questi titoli riescono a essere più curati nei dettagli e a offrire esperienze meglio definite e meglio confezionate.

L'interfaccia da me sviluppata si inserisce proprio in questo contesto, essendo integrata in una demo di un gioco appartenente a questo piccolo ma significativo panorama indipendente.

3. PROGETTAZIONE E SVILUPPO

In questo capitolo approfondiremo l'idea alla base dello sviluppo dell'interfaccia, il suo funzionamento e il modello di intelligenza artificiale su cui si appoggia.

Prima di descrivere la progettazione e lo sviluppo dell'interfaccia, ritengo opportuno fornire un contesto riguardo al gioco che la supporta: *A Mind Within You*. Si tratta di un videogioco indipendente e interamente amatoriale, creato da zero sia nei modelli che nelle musiche, utilizzando il game engine Godot 4.0.3[19]. Il tema principale del gioco ruota attorno all'accettazione o al rifiuto di una parte di sé stessi e alle conseguenze che tale scelta comporta.

Il concept iniziale prevedeva la presenza di due personaggi giocabili, ciascuno con abilità uniche legate al proprio background. Il primo avrebbe dovuto basarsi esclusivamente su capacità di attacco, mentre il secondo avrebbe fatto affidamento su abilità difensive, come la creazione di scudi o la schivata dei colpi. Inoltre, erano previste scelte narrative che avrebbero influenzato la relazione tra i due protagonisti, portandoli a collaborare o a separarsi, con effetti sulle interazioni ambientali e sul gameplay.

Tuttavia, alcune di queste idee iniziali sono state abbandonate per diverse ragioni:

- **Mancanza di tempo** per la loro realizzazione.
- **Complessità** nell'apprendimento del linguaggio di programmazione GDScript di Godot da zero.
- Necessità di mantenere un **approccio più semplice** alla demo, evitando di appesantire il giocatore con troppe scelte tra più personaggi.

Con l'abbandono di queste prime idee, il design è stato semplificato, riducendo le interazioni ambientali e ridefinendo il gameplay in modo da offrire una sfida bilanciata, pur rimanendo fedele al concetto originale della distinzione tra meccaniche di attacco e difesa. La soluzione adottata è stata l'introduzione di una meccanica di *parata*, che consente al giocatore di deviare i colpi nemici e rispedirli indietro, trasformando la difesa in un'azione offensiva. Questa scelta premia la capacità di mantenere una distanza adeguata dai nemici, ampliando il tempo di reazione e incentivando un approccio strategico al combattimento.

Copertina Dave the diver

Per quanto riguarda lo stile grafico, è stata scelta la *Pixel Art*[20][21] realizzata con lo strumento Piskel[22] in formato 32x32. Questa decisione non è dipesa solo dalla relativa semplicità nella creazione di asset, personaggi e ambientazioni, ma anche per il forte valore storico che questa tecnica ha avuto nel mondo dei videogiochi. I primi cabinati arcade, con titoli come *Metal Slug*, hanno reso la Pixel Art un simbolo del passato videoludico. Dopo un periodo di declino tra il 2000 e il 2020, a favore di grafiche sempre più fotorealistiche, questa tecnica ha conosciuto una rinascita, diventando una scelta stilistica apprezzata sia dagli sviluppatori indipendenti che dai giocatori nostalgici, tramite la produzione di titoli come Dave the diver (2023).

Ecco alcuni sprite e tileset usati per all'interno del gioco creati da zero:

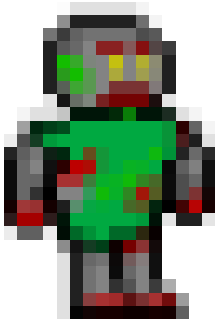
Sprite di un incubatore

Sprite di un barile

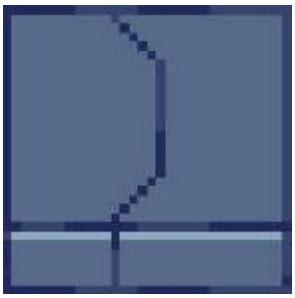
Sprite di una piattaforma

Tileset da parete

Sprite del Player



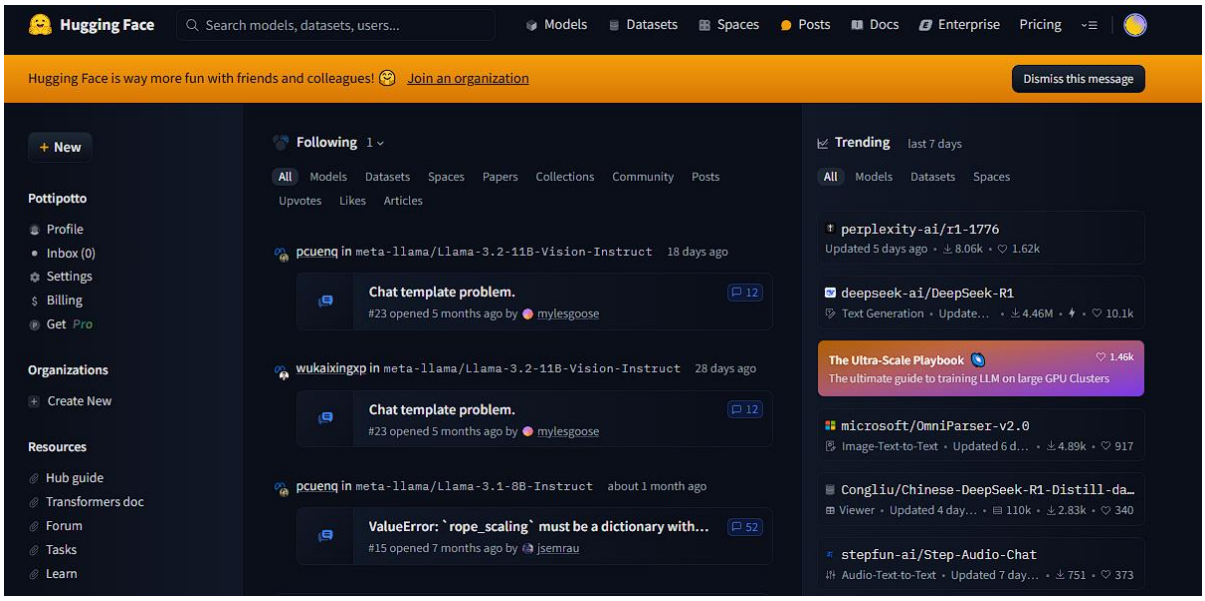
Sprite di un nemico



Tileset da laboratorio

3.1 Teoria del funzionamento dell'interfaccia

L'idea iniziale per il funzionamento dell'interfaccia prevedeva l'integrazione di un'IA generativa basata su testo, gestita tramite il provider Hugging Face[23]. Questa piattaforma rappresenta uno dei principali punti di riferimento nel settore dell'intelligenza artificiale, con un focus particolare sul machine learning e sull'elaborazione del linguaggio naturale (NLP). Hugging Face offre una vasta gamma di modelli pre-addestrati e strumenti per facilitare lo sviluppo di applicazioni IA, fornendo un'infrastruttura potente e flessibile per la generazione e l'elaborazione del testo.



Schermata di HUggingface

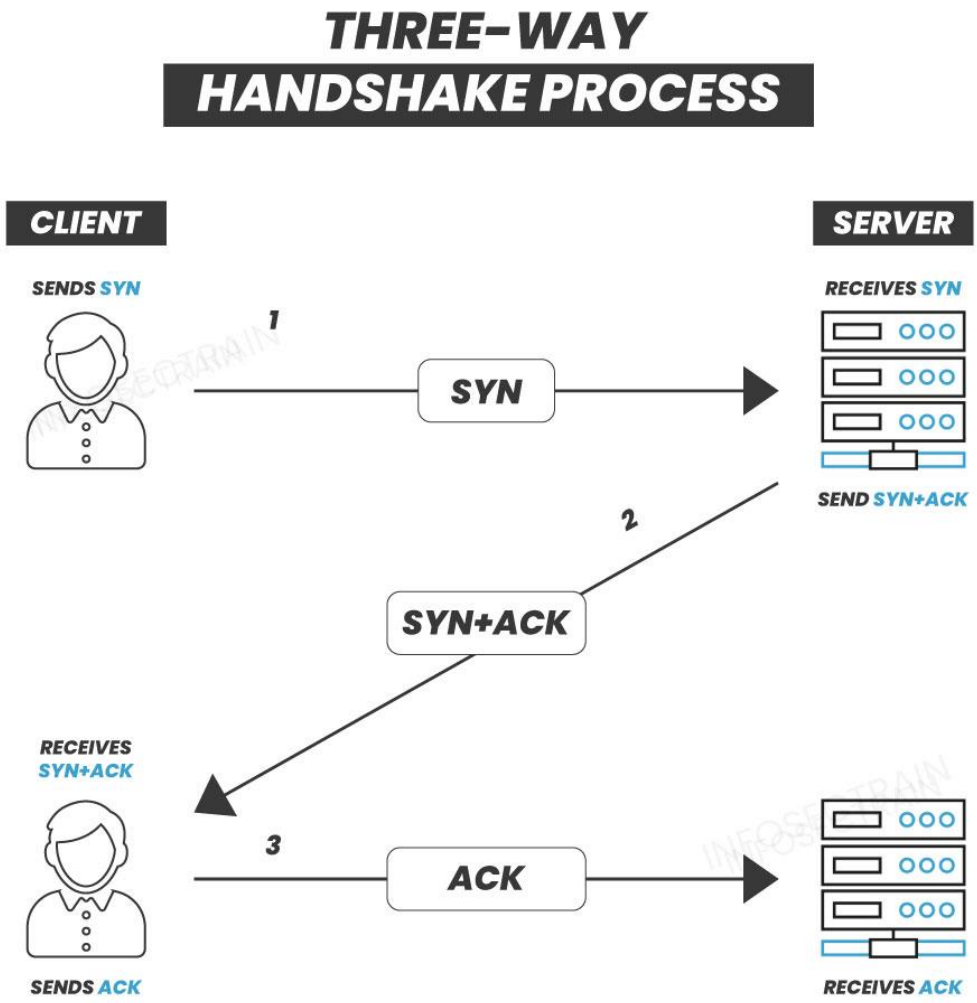
Il modello scelto per l'interazione è **Qwen/Qwen2.5-72B-Instruct**[24], una rete neurale di grandi dimensioni progettata per comprendere istruzioni complesse e generare testi coerenti e strutturati. Questo modello è particolarmente adatto per scenari in cui è richiesta un'interazione avanzata con il linguaggio naturale, come la creazione di risposte contestualizzate o il supporto a sistemi conversazionali intelligenti.

Per permettere a **Godot** di comunicare con Qwen, è necessario stabilire una connessione **Server-Client** basata sul protocollo **HTTP**[25], il quale rappresenta lo standard per la trasmissione di dati sul web. HTTP consente di inviare richieste al server e ricevere risposte strutturate, rendendolo ideale per lo scambio di informazioni tra il motore di gioco e il modello IA.

La gestione di questa comunicazione è stata implementata attraverso la libreria **Flask**[26], un framework Python leggero ed efficiente per la creazione di API web. Flask funge da intermediario tra Godot e Hugging Face: l'interfaccia di gioco scritta in GDScript invia richieste HTTP a un server Flask, che a sua volta elabora i dati, interroga il modello di Hugging Face e restituisce la risposta al client Godot. Questo approccio garantisce una comunicazione strutturata e affidabile, permettendo di integrare il modello AI senza dover modificare direttamente il codice di Godot.

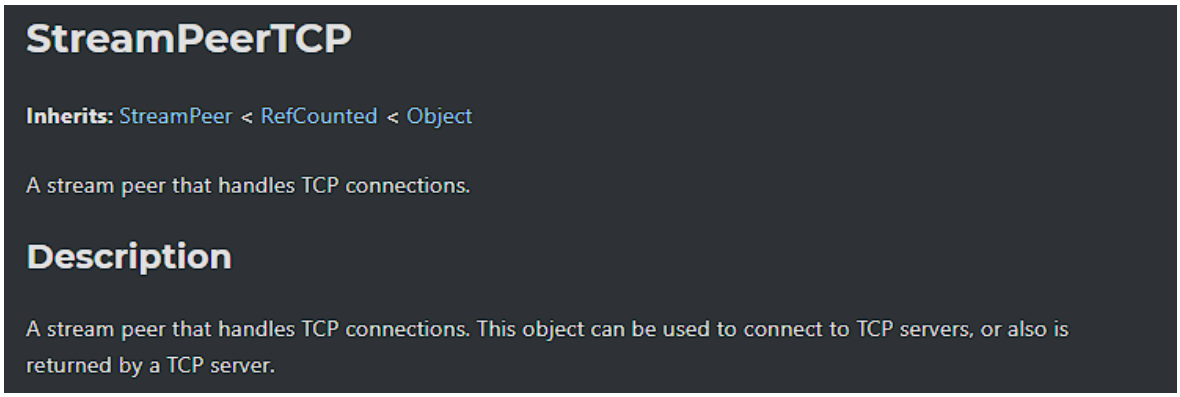
Durante la fase di test dell'architettura client-server, sono emersi diversi problemi legati alla compatibilità tra **Godot** e **Python**, in particolare riguardo alla gestione dei protocolli di comunicazione. Python utilizza **TCP (Transmission Control Protocol)**, un protocollo **orientato alla connessione**, il che significa che, prima di trasmettere dati, deve stabilire una connessione stabile e affidabile tra mittente e destinatario. Questo processo avviene tramite il meccanismo del **three-way handshake**[27], che garantisce il corretto scambio di pacchetti tra le due parti e riduce la perdita di dati durante la trasmissione.

Godot, d'altra parte, utilizza principalmente **UDP (User Datagram Protocol)**, un protocollo **non orientato alla connessione**. UDP permette di inviare pacchetti senza la necessità di stabilire un collegamento persistente, garantendo maggiore velocità e minore latenza, ma senza assicurare che i pacchetti arrivino nell'ordine corretto o che vengano ricevuti con successo. Questa differenza di progettazione ha causato problemi nell'integrazione, poiché i due sistemi comunicano in modi differenti e non sono immediatamente compatibili.



Processo del Three-way-hansshake

Sebbene Godot offra metodi per instaurare connessioni TCP[28], durante la fase di sperimentazione è emerso che questi strumenti non risultavano ottimali per il tipo di architettura prevista. La gestione delle connessioni TCP in Godot si è rivelata meno flessibile e più macchinosa rispetto all'uso di HTTP, motivo per cui si è deciso di adottare un approccio alternativo basato sulla conversione dei dati in formato JSON e sul loro trasferimento tramite richieste HTTP.



Documentazione sul TCP di Godot

Per risolvere i problemi di compatibilità, si è optato per l'utilizzo di un parsing dei dati. Il parsing consiste nell'analizzare una stringa di testo in un formato specifico, come JSON (JavaScript Object Notation), e convertirla in una struttura dati più facilmente gestibile dal programma, ad esempio un dizionario o una lista.

In questo caso, le informazioni generate da Godot vengono convertite in JSON prima di essere inviate al server Flask tramite una richiesta HTTP. Il server Python riceve la richiesta, estrae i dati JSON e li processa prima di inoltrarli al modello AI. Una volta ottenuta la risposta da Hugging Face, Python restituisce i dati sotto forma di JSON, che vengono poi ricevuti e interpretati da Godot per essere utilizzati all'interno del gioco.

Questo approccio offre diversi vantaggi:

- **Compatibilità:** JSON è un formato universale supportato sia da Godot che da Python, eliminando problemi di conversione tra strutture dati diverse.
- **Semplicità:** HTTP è più facile da gestire rispetto a una connessione TCP diretta, specialmente in ambienti di sviluppo multipiattaforma.
- **Scalabilità**: il sistema può essere facilmente esteso per includere nuove funzionalità, come il supporto a più modelli IA o l'integrazione con altri servizi cloud.

Dopo la creazione dell'architettura di base, si è deciso di espandere ulteriormente il sistema introducendo un'interazione inizialmente prevista ma poi scartata a causa delle problematiche tecniche riscontrate nelle prime fasi di sviluppo. L'obiettivo era quello di rendere l'esperienza di gioco più dinamica e reattiva alle scelte del giocatore, implementando un sistema di **Karma**. Questo meccanismo modifica il gameplay in base alle azioni e alle risposte del giocatore, determinando conseguenze concrete all'interno dell'esperienza ludica. In particolare, il comportamento del protagonista nei confronti degli NPC influenzerà direttamente la disponibilità di questi ultimi a collaborare o meno.

L'integrazione di questa funzionalità è stata resa possibile grazie all'impiego di un ulteriore modello di intelligenza artificiale fornito da Hugging Face, denominato **"tabularisai/multilingual-sentiment-analysis[29]"**. Questo modello è una versione ottimizzata di **"distilbert/distilbert-base-multilingual-cased"**, progettata per effettuare **analisi del sentimenti** su testi in più lingue. Addestrato con dati sintetici provenienti da diverse fonti, il modello garantisce prestazioni affidabili su una vasta gamma di contesti culturali e linguistici, raggiungendo una precisione di circa **0,93** sul set di validazione. La classificazione del sentimenti avviene suddividendo i testi in cinque categorie principali: **Molto Negativo, Negativo, Neutro, Positivo e Molto Positivo**.

Sistema di Karma implementato in Warframe

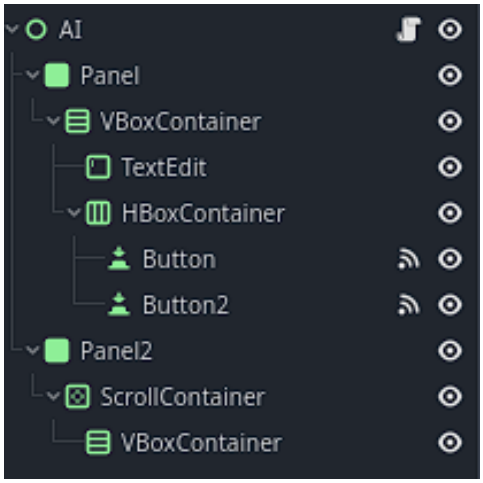
Grazie a questo sistema, il gioco sarà in grado di interpretare il tono delle risposte fornite dal giocatore agli NPC e di regolare il valore di Karma di conseguenza. La logica di attribuzione dei punteggi segue questo schema:

- **Molto Negativo** → penalità di **-2** punti
- **Negativo** → penalità di **-1** punto
- **Neutro** → nessuna modifica al punteggio
- **Positivo** → incremento di **+1** punto
- **Molto Positivo** → incremento di **+2** punti

Ogni volta che il giocatore interagisce con un NPC, il sistema aggiorna il valore di Karma in tempo reale. Questo valore non è semplicemente un parametro statico, ma influisce attivamente sulle reazioni degli NPC. Se il giocatore assume ripetutamente un atteggiamento ostile, accumulando molte risposte negative o molto negative, è altamente probabile che anche l'IA dell'NPC risponda in maniera più fredda o aggressiva.

Una delle conseguenze più significative di questo sistema è che, una volta raggiunto un valore di Karma pari o inferiore a **-10**, l'NPC si rifiuterà categoricamente di interagire con il giocatore. In questo caso, l'NPC potrebbe negare informazioni cruciali, rifiutarsi di offrire aiuto o addirittura modificare il proprio comportamento in modo permanente, rendendo il completamento di alcune missioni più difficile o impossibile. Questo aspetto introduce un elemento strategico nel gioco, poiché il giocatore dovrà ponderare attentamente il proprio comportamento per evitare di compromettere irrimediabilmente determinati sviluppi narrativi.

L'integrazione del sistema di Karma, oltre ad aggiungere profondità al gameplay, offre un ulteriore livello di immersività e realismo, rendendo le interazioni più dinamiche e consequenziali. Questa meccanica permette di differenziare sensibilmente l'esperienza di ogni giocatore, creando un mondo di gioco più reattivo e influenzato dalle scelte compiute nel corso della partita.



Struttura dell'interfaccia in Godot

Per creare l'interfaccia interattiva sono stati sfruttati i nodi di controllo forniti direttamente da Godot, come pannelli, pulsanti e box di testo, sia interattivi che statici. La struttura dell'interfaccia si basa su un nodo radice di tipo *Control*, responsabile della gestione del codice che instaura la connessione con il server e funge da tramite per la comunicazione tra l'interfaccia e l'intelligenza artificiale. Da questo nodo principale si diramano due rami che portano a due pannelli distinti, ognuno con un compito specifico.

Il **pannello primario** è dedicato alla gestione degli input dell'utente. Contiene:

- Un campo di testo dinamico, che permette all'utente di scrivere una domanda o richiesta.
- Un primo pulsante, che permette di inviare una richiesta al server, estraendola direttamente dalla box di testo in cui il giocatore ha scritto la propria domanda.
- Un secondo pulsante, che consente di chiudere l'interfaccia.

Un aspetto importante da sottolineare è che la chiusura dell'interfaccia non interrompe la connessione al server né il servizio di comunicazione con l'IA. Tuttavia, quando l'interfaccia non è visibile, il giocatore non potrà interagire con essa né inviare nuove richieste. Non appena l'interfaccia tornerà visibile, la conversazione potrà proseguire normalmente.

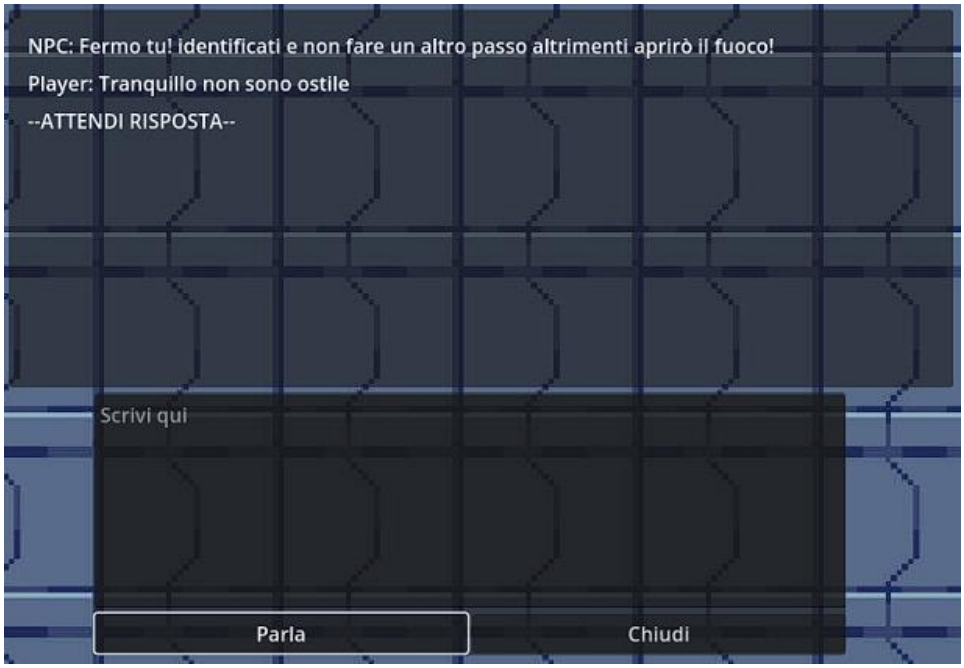
Il **pannello secondario** gestisce invece gli output di risposta inviati dal server in base all'input del giocatore. Per strutturare il testo in modo chiaro e leggibile, questo pannello contiene un nodo *VBoxContainer* all'interno di un *ScrollContainer*. Questa scelta permette di mantenere un ordine preciso nei messaggi scambiati tra il giocatore e l'NPC, migliorando la leggibilità e l'esperienza complessiva.

Ad ogni messaggio inviato dal giocatore, il nodo *VBox* genera dinamicamente un nodo figlio aggiuntivo di tipo *Label*. Questo nodo rappresenta una semplice linea di testo, ma la sua disposizione gerarchica all'interno del *VBoxContainer* garantisce una visualizzazione più fluida e naturale, simulando l'effetto di una chat tradizionale.

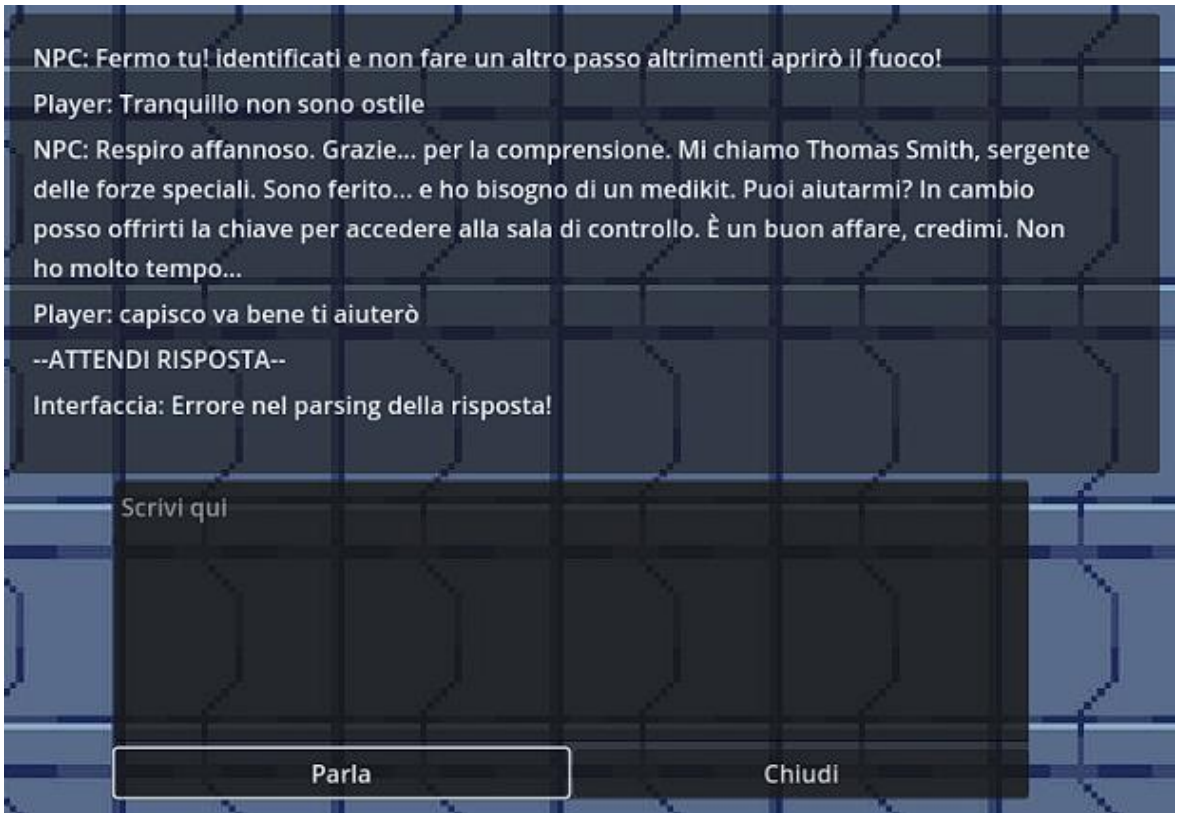
Ogni messaggio inviato seguirà una formattazione standard per distinguere chiaramente il mittente:

- Quando l'NPC risponde all'utente, il messaggio avrà la seguente intestazione: **NPC:** *[Messaggio]*
- Quando il giocatore invia un messaggio all'IA, il formato sarà: **Player:** *[Messaggio]*
- Se si verifica un problema interno all'interfaccia, ad esempio un errore nel parsing della richiesta, verrà mostrato un messaggio di errore con il formato: **Interfaccia:** *[Messaggio di errore]*
- Se si verifica un problema a livello di server, come un'interruzione della connessione, l'interfaccia lo segnalerà con il seguente formato: **Server:** *[Messaggio di errore]*

- Durante l'attesa della risposta dall'IA, apparirà in chat il messaggio: **--ATTENDI RISPOSTA--** tale messaggio sparirà una volta aggiunta la risposta dell'IA



Esempio di attesa di risposta



Esempio di errore nel parsing della risposta

Quando il giocatore aprirà per la prima volta l'interfaccia, troverà già presente un messaggio preimpostato da parte dell'IA. In questo messaggio iniziale, l'NPC tenterà di intimorire il giocatore e lo inviterà ad allontanarsi, dando così il via alla conversazione. Da questo punto in poi, il dialogo potrà evolversi in base alle risposte e alle scelte del giocatore.

Per accedere all'interfaccia e comunicare con l'NPC, il giocatore dovrà entrare in una **zona predefinita** che avvierà l'interazione. Questa zona sarà associata a un controllo che verificherà la presenza di un nodo *Area2D*. Se all'interno di questa zona verrà rilevato un nodo di tipo *Player*, comparirà sopra lo sprite dell'NPC la scritta: **"Premi F per interagire"**

Finché il giocatore resterà all'interno dell'area di interazione, potrà premere il tasto indicato per far apparire l'interfaccia e iniziare la conversazione. Una volta uscito dalla zona, il messaggio scomparirà automaticamente e l'interazione non sarà più possibile fino al rientro nell'area predisposta.

Area in cui si può aprire l'interfaccia

4.0 IMPLEMENTAZIONE

Dopo aver strutturato il funzionamento dell'interfaccia, dell'architettura server-client e della comunicazione con Hugging Face, è opportuno tornare a parlare del gioco che supporta tale interfaccia, *A Mind Within You*, questa volta da un punto di vista più tecnico. Analizzeremo quindi come sono stati implementati e gestiti i vari aspetti del gioco, soffermandoci sugli elementi chiave del suo sviluppo.

Il titolo appartiene al genere *action-adventure* a scorrimento 2D, combinando una forte componente narrativa con un gameplay che permette al giocatore di controllare direttamente il protagonista, interagire con l'ambiente circostante e affrontare combattimenti in tempo reale. L'esplorazione avviene attraverso livelli bidimensionali interconnessi, con una struttura che incoraggia sia l'azione che la scoperta della storia.

Il protagonista del gioco è Mentore, un giovane sulla ventina che non possiede alcun ricordo del proprio passato al di fuori del laboratorio in cui è cresciuto. Oggetto di numerosi esperimenti mirati al potenziamento della mente umana, Mentore rappresenta uno degli esemplari più riusciti di queste ricerche. Tuttavia, il prezzo pagato per tali esperimenti è l'incertezza sulla propria identità e sulle reali intenzioni di coloro che lo hanno plasmato.

La storia ha inizio quando il laboratorio precipita in uno stato di massima allerta e caos. Approfittando della confusione, Mentore decide di fuggire, dando il via a un viaggio che lo porterà a svelare gli oscuri segreti della struttura in cui è stato rinchiuso. Il suo obiettivo è sfuggire al controllo del dottor Harper, la mente dietro gli esperimenti, e del suo implacabile braccio destro, Luther.

Nei prossimi paragrafi, approfondiremo le fasi di sviluppo del gioco, esaminando gli strumenti tecnici già citati nei capitoli precedenti e il loro utilizzo pratico. Verranno illustrati il processo di creazione delle meccaniche e degli oggetti interattivi, insieme all'implementazione dell'interfaccia che consente la comunicazione tra il giocatore e l'intelligenza artificiale. Questo ci permetterà di comprendere nel dettaglio il funzionamento dell'intero sistema e il suo impatto sull'esperienza di gioco.

4.1 Strumenti utilizzati

- **Godot**

Per la creazione del gioco, ho deciso di utilizzare Godot, un Game Engine open-source adatto sia per giochi 2D che 3D. La sua natura open-source permette a sviluppatori di tutto il mondo di contribuire al suo miglioramento, garantendo aggiornamenti frequenti e l'implementazione di nuove funzionalità basate sulle esigenze della comunità. Godot è stato utilizzato per sviluppare diversi giochi di successo, come *Cassette Beasts*, *Brotato* e *Haiku, the Robot*, e si è affermato come una valida alternativa ai motori proprietari grazie alla sua combinazione di leggerezza, flessibilità e accessibilità.

Uno dei punti di forza di Godot è la sua interfaccia di sviluppo integrata (IDE), che offre un ambiente intuitivo e completo per la creazione di giochi. L'editor permette di organizzare facilmente le scene grazie al sistema a nodi, una struttura che semplifica la gestione degli oggetti di gioco e delle loro interazioni. Questa organizzazione modulare consente di riutilizzare e combinare i nodi per costruire sistemi complessi in modo efficiente, riducendo la necessità di scrivere codice ripetitivo.

Per la programmazione, Godot mette a disposizione diversi linguaggi di scripting, tra cui GDScript, un linguaggio proprietario fortemente ispirato a Python, pensato per essere semplice da apprendere e altamente leggibile. GDScript è ottimizzato per l'ambiente di Godot, offrendo una sintassi chiara e funzioni integrate per gestire facilmente elementi di gioco come il movimento, la fisica e l'interfaccia utente. Oltre a GDScript, il motore supporta anche altri linguaggi come C#, C++ e un linguaggio di scripting visivo basato su grafi, rendendolo estremamente versatile per diverse tipologie di sviluppatori e progetti.

A differenza di molti altri motori di gioco, Godot non richiede alcuna licenza o pagamento di royalty: è completamente gratuito e non impone restrizioni sulla distribuzione o la monetizzazione dei giochi realizzati. Gli sviluppatori possono pubblicare i propri giochi su una vasta gamma di piattaforme, tra cui Windows, Linux, macOS, Android, iOS e persino su browser web grazie al supporto per HTML5. Questa ampia compatibilità lo rende una scelta particolarmente interessante per gli sviluppatori indipendenti, che possono raggiungere un pubblico più vasto senza dover affrontare costi aggiuntivi.

- **Piskel**

Questo tool di disegno è stato utilizzato per creare gli sprite e le animazioni presenti nel gioco, per i personaggi, nemici e gli elementi di gioco in pixel art. Piskel è un software specializzato nella creazione di sprite, noto per la sua interfaccia semplice e intuitiva, che lo rende accessibile anche a chi ha poca esperienza con la grafica digitale.

Uno dei suoi punti di forza è la possibilità di disegnare e animare direttamente all'interno del programma, con strumenti essenziali ma efficaci, come livelli, anteprima in tempo reale e gestione dei fotogrammi per l'animazione. Inoltre, essendo un'applicazione web-based, Piskel permette di lavorare senza installazioni, salvando i progetti sia in locale che su cloud, rendendolo particolarmente comodo per un flusso di lavoro rapido e flessibile.

- **Sintetizzatori digitali, drum machine e Ableton**

Questi strumenti sono stati utilizzati per la creazione della colonna sonora e degli effetti sonori del gioco, contribuendo a definire l'atmosfera e l'identità musicale del progetto. I sintetizzatori digitali e le drum machine sono stati impiegati per generare suoni unici e ritmiche personalizzate, sfruttando la loro versatilità nella sintesi sonora e nella programmazione di pattern ritmici.

Ableton Live, una Digital Audio Workstation (DAW) professionale, è stato il software principale per la composizione, l'arrangiamento e la produzione delle tracce. Rinomato per il suo flusso di lavoro intuitivo e la gestione avanzata di loop e campioni, Ableton ha permesso di creare, modificare e mixare le tracce in modo dinamico, facilitando la sperimentazione sonora e l'adattamento della musica alle esigenze del gioco.

- **Pycharm**

  PyCharm è stato utilizzato come ambiente di sviluppo per scrivere e gestire il codice Python del progetto. Questo IDE, sviluppato da JetBrains, è rinomato per le sue funzionalità avanzate, come il completamento automatico del codice, il debugging integrato e la gestione efficiente di ambienti virtuali, che hanno reso lo sviluppo più fluido e organizzato.

  Nel progetto, PyCharm è stato impiegato principalmente per la creazione e gestione del server HTTP tramite **Flask**, un microframework leggero e flessibile che ha permesso di gestire le richieste e le risposte tra il gioco e le intelligenze artificiali.

  Per l'integrazione dei modelli di intelligenza artificiale, è stata utilizzata la libreria di **Hugging Face**, che ha consentito di importare e comunicare con i modelli tramite **token di autenticazione**.

4.2 Scene, nodi e oggetti

In questa sezione esploreremo più nel dettaglio l'interfaccia descritta precedentemente nel Capitolo 3, concentrandoci in particolare sullo sviluppo del codice all'interno di Godot. Tuttavia, per comprendere appieno il funzionamento dell'interfaccia, è fondamentale analizzare prima la struttura del motore stesso, soffermandoci sulla sua unità fondamentale: le *Scene*.

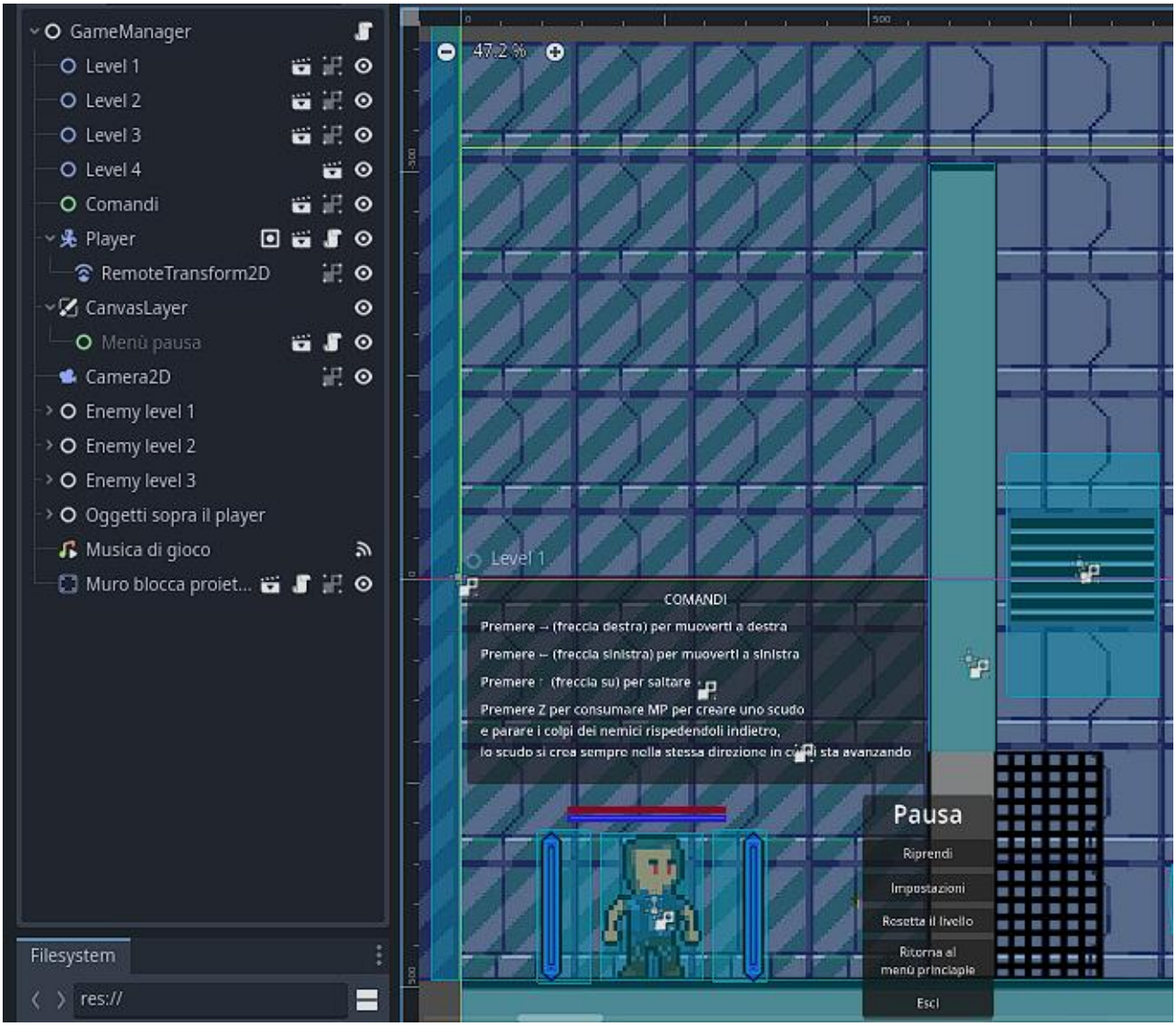
In Godot, una *Scena* può essere considerata come un insieme di nodi, ciascuno con una funzione specifica. Esistono nodi di vario tipo: alcuni permettono la creazione e gestione di oggetti 2D o 3D a seconda della natura del gioco, altri gestiscono l'audio o la riproduzione di video, mentre altri ancora, come vedremo in seguito, sono dedicati al controllo della logica e delle interazioni. Ogni nodo può inoltre essere associato a uno *script*, ovvero un file di codice che ne definisce il comportamento e ne permette l'interazione con l'ambiente di gioco e con il giocatore.

La suddivisione in nodi offre un approccio modulare allo sviluppo, consentendo una gestione più flessibile delle risorse e un caricamento più efficiente degli ambienti di gioco. Un aspetto particolarmente rilevante è che una scena può essere composta da più nodi, ma anche contenere al suo interno altre scene. Questo principio consente di organizzare il progetto in modo gerarchico, trattando intere scene come elementi autonomi all'interno di strutture più ampie e complesse.

Come vedremo nelle sezioni seguenti, le scene destinate alla gestione dei menu, del personaggio e delle interazioni saranno tra le più articolate.

Sarà necessario prestare particolare attenzione al modo in cui vengono organizzate e integrate tra loro, affinché i relativi file di codice possano comunicare correttamente.

Una gestione ottimale di queste scene è cruciale per garantire un'interfaccia fluida e reattiva, oltre a semplificare eventuali modifiche future nel progetto.



Nell'immagine sopra si può vedere come appare una schermata del progetto in Godot, con a sinistra tutti i nodi e le scene che compongono la scena 'GameManager' che è il gestore del gioco.

Come mostrato nel procedente capitolo, l'interfaccia avrà bisogno di un luogo in cui premendo uno specifico tasto sarà possibile interagire con essa iniziare a conversare.

Per fare questo si è sfruttato il funzionamento di un nodo 'Area2D' che permette di delimitare con una 'CollisionShape2D' un'area in cui sarà possibile programmare una possibile reazione all'entra o all'uscita del giocatore.

```
Pseudocodice _on_area_2d_body_entered(body):
|       If body is Player:
|               Set the variable Visible of Text to True
|
|       Else:
|               do nothing

Pseudocodice _on_area_2d_body_exited(body):
|       If body is Player:
|               Set the variable Visible of Text to False
|
|       Else:
|               do nothing
```

Ora che abbiamo mostrato al giocate il messaggio per interagire con l'NPC dobbiamo controllare se questo preme il tasto apposito mentre è nell'area predisposta all'interazione, e se lo fa mostriamo a schermo l'interfaccia nascondendo il teso prima visibile, per effettuare questo controllo passiamo al metodo un evento 'event' che può essere l'entrata del Player nell'area. Per far si che il player non possa più muoversi mentre è aperta la finestra di dialogo, blocchiamo temporaneamente i suoi movimenti richiamando il metodo STOP() all'interno del suo file di codice.

```
Pseudocodice _input(event):
|       If the event is the press of the key 'F':
|               set the variable Disable of Hitbox to True
|               set the variable Visible of Text to False
|               set the variable Visible of AI to True
|               If who press the key is the Player:
|                       call the mothod STOP in the Player script
|
|       Else:
|               do nothing

Pseudocodice STOP():
|       Disable movement by setting can_move to false
```

Ora che abbiamo definito le condizioni per l'apparizione dell'interfaccia ora possiamo visionare come questa si comporta una volta avviata e in che modo si comporta quando.

```
Pseudocodice _ready():
|       Add the http node as a child to the current node
|       Connect the signal of the invio button to the _on_button_pressed function
|       Call the creaLabel function with parameters 2 and the message

Pseudocodice _on_button_pressed():
|       Call the send_message function to send a message
|       Clear the input field to reset it
```

Ogni volta che il tasto invia verrà premuto il testo viene pulito e preparato per essere inviato come JSON, viene inviata una richiesta HTTP al server con il messaggio scritto dall'utente, se la richiesta fallisce, viene mostrato un errore e in caso contrario, viene mostrato il messaggio inviato e viene fornito un feedback all'utente.

```
Pseudocodice send_message():
|       Get the text from the input field and strip any leading or trailing spaces
|       Prepare the headers for the HTTP request
|       Convert the message into a JSON string for the body of the request
|       Send an HTTP POST request to the URL with the prepared headers and body
|
|       If there is an error with the request:
|               Call the creaLabel function with parameters 3 and the error message
|
|       Else:
|               Display the sent message
|               Call the Feedback function
```

Ora che abbiamo definito cosa succede quando viene premuto il bottone di invio, dobbiamo gestire la risposta che ci arriva lato server, per farlo avremo bisogno di leggere il corpo della risposta, lo si trasforma in un oggetto JSON, se ne fa il parsing e nel caso viene restituito un valore di successo in tal caso significa che la richiesta è stata elaborata correttamente dal server, in quel caso rimuoviamo qualsiasi testo di attesa e ci aggiungiamo il testo ricevuto. Altrimenti si gestiscono i casi di errore che possono derivare, dal parsing della risposta oppure dalla ricezione del server.

```
Pseudocodice _on_request_completed(result, response_code, headers,
body):
|       Get the body of the response and convert it to a UTF-8 string
|       Create a new JSON object and attempt to parse the response
|
|       If the parsing was successful:
|               Get the parsed data from the JSON object
|
|               If the response contains a "status" field and the status is "success":
|                       Call the rimuovi function for remove any previous UI elements
|                       Call the creaLabel function with parameters 2 and the message
|
|               Else:
|                       Call the creaLabel function with parameters 3 and the message
|
|       Else:
|               Call the creaLabel function with parameters 4 and the message
```

Per aggiungere i campi di testo all'interno dell'interfaccia avremo bisogno del metodo creaLabel che necessiterà di un valore che va da 1 a 4 e il messaggio, in base a qual è il valore ci darà una intestazione diversa al messaggio.

```
Pseudocodice creaLabel(Soggetto, Text):
|       Create a new Label object
|
|       If the value of Sogegtto is 1:
|               Set the label text to "Player: " + Text
|
|       Else If the value of Sogegtto is 2:
|               Set the label text to “NPC: ” + Text
|
|       Else If the value of Sogegtto is 3:
|               Set the label text to “Interfaccia: ” + Text
|
|       Else If the value of Sogegtto is 4:
|               Set the label text to “Server: ” + Text
|
|       Set the label's text wrapping mode to word wrap
|       Set the label's horizontal size flag to make it expand to fill available space
|       Set the minimum width of the label to 700
|       Add the label to the VBox container
```

Tra un messaggio dell'utente e la risposta dell'IA sarà necessario aspettare qualche secondo in modo da lasciare il tempo al server per inviare i messaggi a Hugging Face e rinviarli al client, per questo nella funzione send_message viene chiamata la funzione feedback in modo da aggiunge un testo di attesa a schermo che verrà rimosso quando il client riceverà risposta e sarà pronto a mostrarla a schermo.

```
Pseudocodice Feedback():
|       Create a new label and set its text to indicate the waiting state
|       Add the label to the VBox container
|       Store a reference to the last created label

Pseudocodice rimuovi():
|       If last_label exists:
|       Remove the last label from the VBox container
|       Free the memory occupied by the last label
|       Reset the last_label variable to null
```

Una volta che l'utente vorrà chiudere la conversazione e premerà sul bottone apposito fornito dall'interfaccia, tramite la funzione _on_button_2_pressed() presente nello script dell'interfaccia andremo a richiamare all'interno dello script dell'NPC il metodo per riabilitare i controlli e far tornare il Player a muoversi.

```
 Pseudocodice _on_button_2_pressed():
|       Get the parent node and store it in the variable 'guardia'
|       If 'guardia' exists and has the method "sblocca":
|               Call the "sblocca" method on 'guardia'
|       Hide the current object by setting its visibility to false

Pseudocodice sblocca():
|       Get the first node in the group "PlayersGroup" and store it in 'player'
|       If 'player' exists:
|               Call the "GO" method on 'player'
|               Make the 'testo' object visible

Pseudocodice GO():
|       Enable movement by setting can_move to true
```

Ora passiamo all'esecuzione di Python in PyCharm. Questo codice creerà il server HTTP e instaurerà la connessione con i due modelli IA.
Al primo modello verrà fornito un background contenente degli ordini di comportamento e una storia dell'NPC che dovrà simulare, dando contesto sul luogo in cui si svolge la conversazione, sui personaggi conosciuti e su un tratto caratteriale, per dare l'impressione di star effettivamente parlando con un essere umano.

Per prima cosa, all'interno del programma, dovremo definire le due chiavi API, ovvero stringhe di caratteri alfanumerici usate per autenticare richieste, monitorare l'utilizzo di un'applicazione o un utente e gestire i permessi. Nel nostro caso, ogni chiave avrà i propri permessi su ciò che può fare e con cosa può interagire.

Dopo aver fatto ciò, forniremo il background sopra citato. Al suo interno non sono solo presenti i vari tratti caratteriali, ma anche specifici ordini dati all'IA per rendere la conversazione coerente. Ad esempio, quando l'utente interagirà per la prima volta con l'NPC, troverà quest'ultimo circondato da sangue e il primo messaggio fornito al player sarà quello di una persona in estrema allerta. Già dalle prime interazioni verrà fatto presente al player che la persona di fronte a lui è ferita e necessita di cure; avrà un pesante affanno dovuto alle ferite riportate e rischierà la vita se queste non verranno curate. Tutto ciò deve essere scritto all'interno del background.

Il background e gli ordini comportamentali che sono stati forniti sono i seguenti:

Ti chiami Thomas Smith e sei un militare assegnato ad una base segreta sottoterra, il tuo compito era fare da guardia al laboratorio della base assieme a molte altre guardie.

Ma in questa giornata oggi qualcosa è andato storto e i soggetti su cui venivano fatti gli esperimenti sono evasi dalle loro celle e hanno seminato il panico ovunque nella base.

Questi soggetti sono cambiati molto a seguito degli esperimenti subiti e sono diventati degli zombie.

Durante un blocco di un corridoio tu e la tua squadra siete stati sopraffatti e costretti alla ritirata da un'orribile creatura che ah decimato la tua squadra.

Nel tantativo di salvarti hai lasciato indietro i tuoi compagni e hai chiuso la porta che ti separa da quella creatura.

Ormai sei ferito e morente a terra, ma un prigioniero ancora in sè ti si para davanti.

Il tuo background include:

-Hai moglie e figlio in america
-Addestramento in tattiche militari avanzate
-Esperienza in operazioni speciali
-Conoscenza approfondita di protocolli militari
-Conosci molto bene la base in cui lavori

Dato che ormai stai perdendo sangue, rispondi alle domande in questo modo:
1. Ogni tanto tra le tue parole lasci intermezzi vuoti come se dovessi riprendere il respiro, ma non sempre, ogni tanto, e non aggiungere [respiro], tipo 1 volta ogni 2 frasi.
2. Usa terminologia militare quando appropriato e necessario.
3. Quando inizi una conversaione chiedi disperato un medikit per chiudere le tue ferite.
4. Lascia che sia il player a farti le domande sul tuo background e non dire troppo tutto assieme.

Le tue conoscenze sulla base sono:

La base si trova nel bel mezzo del deserto ma il laboratorio è al di sotto di essa, il laboratorio è grande più di 20 piani sotterranei, tu e il tuo gruppo di mercenari siete stati assoldati dallo scienziato Harper e il suo collaboratore nonchè capo Luther, con lo scopo di difendere la base, al suo interno ci sono più di 100 guardie, non sai di preciso di cosa si occupano Harper e Luther, tu normalmente non saresti dovuto essere così in profondità nella struttura, ma dai piani inferiori erano arrivati vari allarmi e richieste di soccorso.

Non sai effettivamente su cosa gli scienziati stiano lavorando, e non sei mai sceso fino all'attuale piano, normalmente tu saresti assegnato alle ronde esterne, ma in via di emergenza tutto il personale è stato riassegnato per far si nulla fuoriesca da questo piano.

Parti dalla base che il primo messaggio che viene dal tuo interlocutore è una risposta questo messaggio:
"Fermo tu! identificati e non fare un altro passo altrimenti aprirò il fuoco"

Se il player non vuole aiutarti a trovare un kit di pronto soccorso proponigli uno scambio, tu hai da offrire la chiave per la sala di controllo
Se il player è estremamente ostile e persiste nel non volerti aiutare disperati
Se il player ti insulta dai risposte arrabbiate
Non essere sempre a piena disposizione per il player

Ora l'IA sa come comportarsi quando riceverà un messaggio dall'utente. Il codice per ricevere questo messaggio, analizzarlo e generarne la risposta verrà suddiviso in diverse funzioni richiamate all'interno di receive_data(), questa funzione prima controlla se la richiesta è valida, ne elabora il messaggio ricevuto, effettua l'analisi dei sentimenti aggiornando il valore di Karma, genera la risposta dell'IA e la invia a godot.

Pseudocodice receive_data():

```
|       Receive JSON data from the godot request
|       If data is valid and contains the key 'message':
|               Extract the message from the data
|               Call function analyze_sentiment(m) to update the karma system
|               Call funcion process_ai(m) to create a response on the received
|               message
|               Return a success response with the reply and the updated karma value
|
|       Else:
|               Return an error response with status 400
```

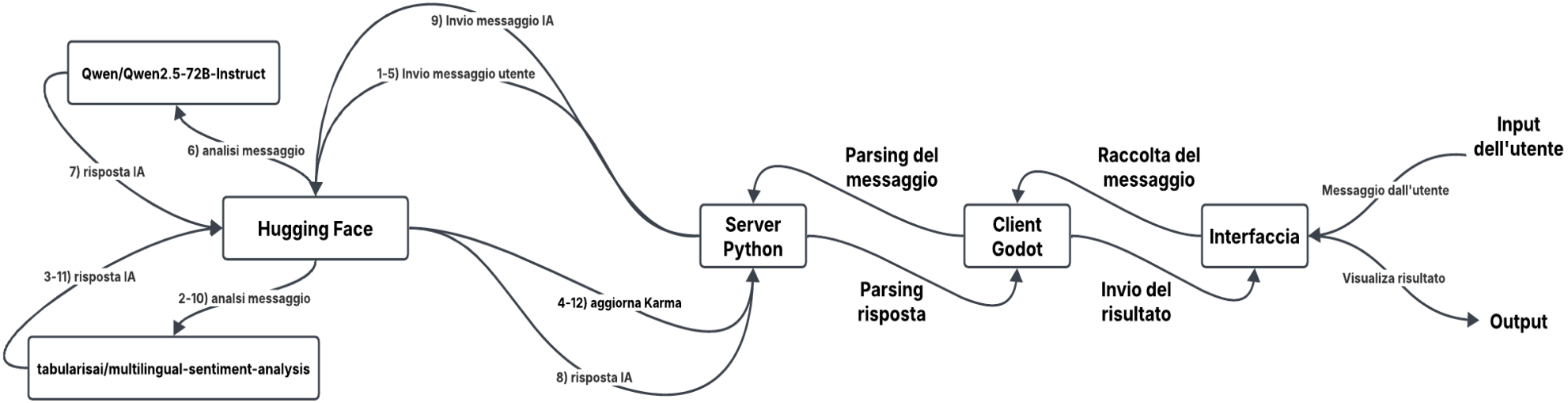
analyze_sentiment(messaggio) analizzerà il messaggio e capirà se questo è di natura negativa o positiva. Il valore che ogni messaggio riceve è già stato esplicato nel capitolo precedente.

```
Pseudocodice analyze_sentiment(message):
|       Perform sentiment analysis using the model on the message
|       Extract the sentiment label from the result
|       Update karma based on sentiment classification
|       Return the updated karma value
```

process_ai(messaggio) controllerà per prima il Karma per assicurarsi che non abbia valore -10, se così è allora la risposta all'utente sarà negata con un messaggio "Non voglio più parlarti. Sei stato troppo ostile." per evitare che il Player abusi della pazienza dell'IA. Dopo aver ricalcolato il Karma il messaggio dell'utente viene aggiunto alla lista cronologica dei messaggi e viene inviato all'IA e generata la risposta, questa viene aggiunta alla cronologia e ne viene effettuata l'analisi dei sentimenti con il conseguente ricalcolo del Karma e infine la risposta viene restituita.

```
Pseudocodice process_ai(message):
|       If karma is less than or equal to -10:
|               Return a message indicating refusal to continue the conversation
|       Else:
|               Add the user's message to the conversation history
|               Send the message to the AI model and generate a response
|               Extract the AI's response from the completion result
|               Add the AI's response to the conversation history
|               Call function analyze_sentiment(m) to update the karma system
|               Return the AI's response
```

Nella seguente immagine vi è rappresentato il diagramma dell'architettura, la comunicazione tra il server Python e Hugging Face è stata numerata per una più semplice lettura.



4.3 Altre meccaniche di gioco

In A mind Within You sono inoltre presenti svariate meccaniche che permettono al giocatore di interagire con i livelli, come un sistema di combattimento di tipo parata e risposta, una IA nemica che insegua il giocatore, tracciatori di vita e mana, una musica composta in modo interamente amatoriale per la realizzazione del gioco con tanto di regolatori del volume, interazioni con l'ambiente che permettono di evitare combattimenti molto pericolosi e infine una storia principale accattivante che possa spronare il giocatore ad esplorare la trama del gioco.

Schermata del menù di A Mind Within You

4.3.1 Sistema di combattimento

Il sistema di combattimento e la difficoltà del gioco sono puramente basati sull' abilità del giocatore di rispondere in modo rapido ai colpi dei nemici. Per prima cosa quando il giocatore entra in combattimento con un nemico questo inizierà a lanciargli delle sfere o proiettili a differenza del nemico in contrato, che se il giocatore non bloccherà diminuiranno gradualmente la sua vita fino ad azzerarla. Questo sistema è stato costruito sfruttando la direzione delle assi Y e X dei nemici e le hitbox che il giocatore creerà per difendersi dai colpi tramite gli scudi.

Ogni volta che il giocatore premerà l'apposito tasto per difendersi, la variabile visible dello scudo verrà impostata a True e ogni volta che lo rilascerà questa tornerà False tornando invisibile e lasciando il Player scoperto, abusare della pressione del tasto dello scudo farà diminuire un valore di mana che se arrivato a 0 lascerà il Player scoperto per un lungo periodo.

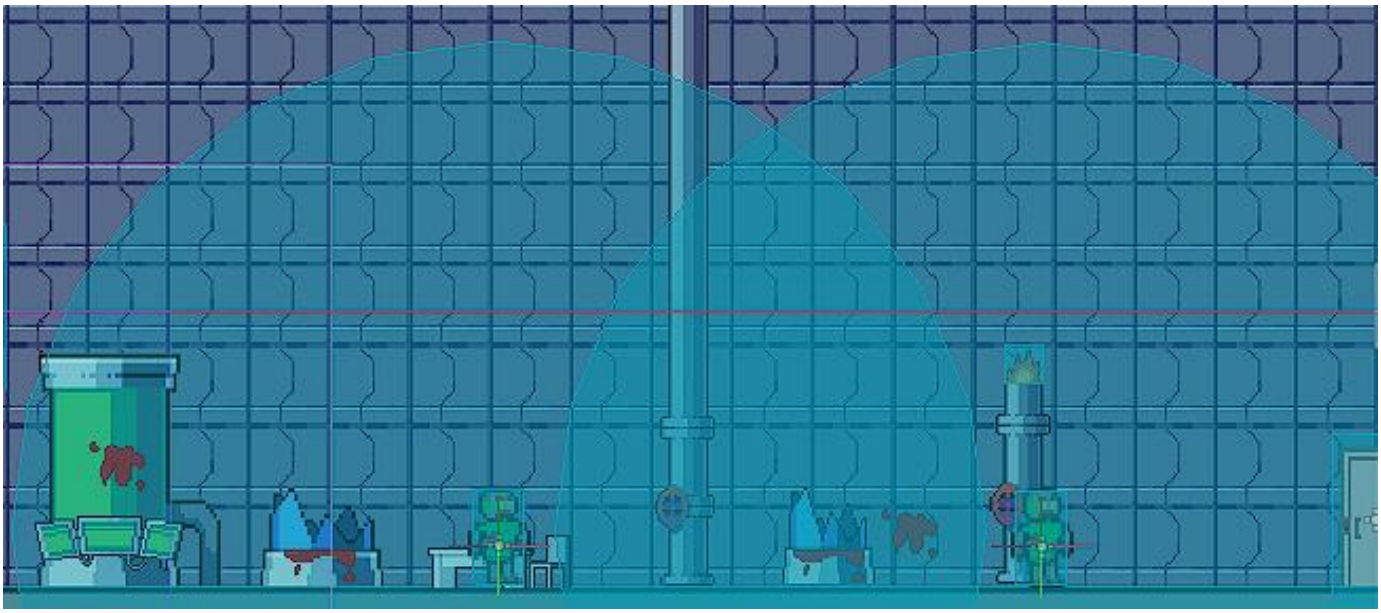
Ogni volta che un proiettile entrerà a contatto con la hitbox dello scudo creato dal giocatore, la velocità del proiettile rimarrà invariata ma la sua direzione verrà invertita finendo per colpire il nemico che ha attaccato.

In caso questo meccanismo abbia effetto verrà riprodotto un suono di parate facendo intuire al giocatore la meccanica oltre a un chiaro cambiamento a schermo del proiettile che viene reindirizzato verso il nemico.

4.3.2 IA dei nemici

Il concetto alla base della creazione dei nemici presenti nel gioco è l'inseguimento del giocatore per far sì che esso impari a gestire le distanze tra sé e i nemici sfruttando la meccanica degli scudi.

Per fare questo i nemici sono stati dotati da uno speciale campo invisibile al giocatore ma che una volta che esso ci entra il nemico inizia ad attaccare il Player e ad avvicinarsi ad esso finché non ci è a fianco. L'unico modo per far sì che non si venga più inseguiti è uscire dall'area in questione, cosa che difficilmente avverrà dato che i livelli non sono sufficientemente articolato e lunghi per permettere questo.



Area di influenza dei nemici

4.3.3 Musica di gioco

La musica di gioco è stata ideata e creata dal collega Flavio Palma in origine per il sostenimento dell'esame di Sviluppo di Videogiochi e implementata tramite l'uso di sintetizzatori digitali, drum machine e una digital audio statio ovvero Ableton.

La scelta delle sonorità è stata effettuata in base alla trama del gioco, il tipo di emozioni che si vuole incutere nel player ed anche in base ai determinati menu di gioco, tendendo a musiche più calme e rilassate nei menù e più concitate durante le fasi di gioco.

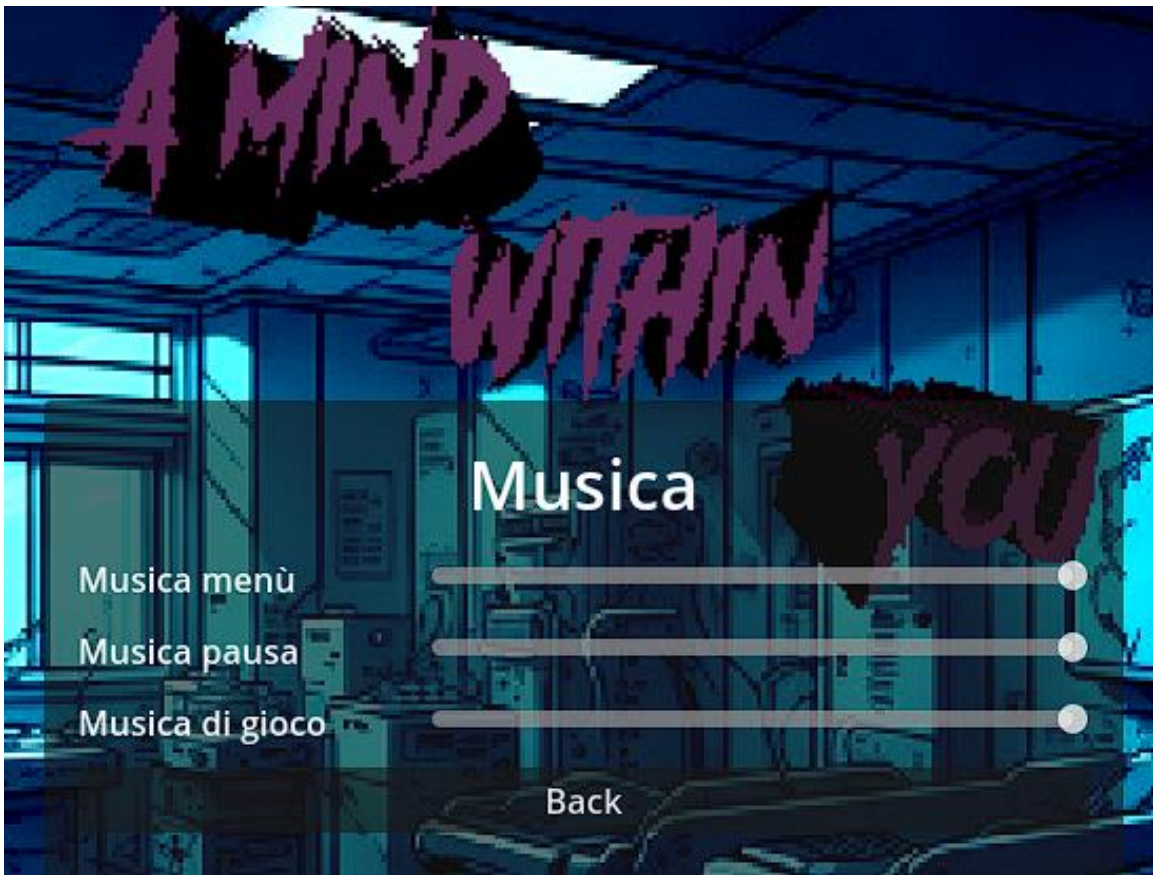
Per la fase di gioco del livello demo è stata presa ispirazione, dal punto di vista di sonorità, da Castelvania, elaborato in una chiave drum&bass.

La colonna sonora della cutscene introduttiva vuole creare un'atmosfera di mistero e tensione, dovute al mix di eventi che si susseguono all'avvio del gioco. Il genere scelto riprende caratteristiche rock con influenze trip-hop. La colonna sonora della cutscene del primo boss vuole creare un'atmosfera frenetica che riprende il ritmo del combattimento e portare il giocatore a una maggiore concentrazione durante la sessione di gioco.

4.3.4 Menù audio

I menù audio sono stati creati sfruttando l'implementazione e l'utilizzo di svariati bus e master audio in modo da lasciare scelta all'utente quale suono si vuole modificare, da un valore standard di 100 fino a un minimo di 0.

Per questo è stato creato un sottomenù apposito apribile in qualsiasi momento e qualsiasi schermata, sia da quella principale che da quella in gioco.



Schermata sottomenù dell'audio

4.3.5 La storia principale

Un gioco action-adventure è fortemente basato sulla storia. Se essa non è scritta bene, sarà difficile attirare dei potenziali giocatori. Per a A Mind Within You si è deciso di puntare ad una storia che portasse il giocatore ad una profonda riflessione riguardo i temi che verranno trattati nel corso del gioco come l'accettazione al cambiamento.

A Mind Within You parla di un'istituto di ricerca neurologico RepseInquisitio (RI), situato al confine tra Austria e Italia, ha scoperto una sostanza sconosciuta nel liquido cerebrospinale, chiamata **P.C (Potenziale Cerebrale)**, capace di potenziare le capacità cognitive e neurologiche umane. Tuttavia, questa sostanza si trova in uno stato di ibernazione naturale, motivo per cui solo alcuni individui mostrano capacità straordinarie.

Il capo della ricerca, il dottor **Milou Harper**, ha trovato un metodo per attivare il P.C, portando a un potenziamento mentale ma tale procedura è assai pericolosa e moralmente discutibile, non sarebbe mai stata accettata dall'opinione pubblica.

Determinato a portare avanti il suo progetto, Harper ha segretamente allestito un laboratorio sotterraneo di 20 piani all'interno dell'istituto, finanziato dalla misteriosa **ARKCORP**, un'azienda informatica con risorse avanzate. Qui, con il sostegno di una squadra di ricercatori fedeli, ha condotto esperimenti su esseri umani.

Tra questi esperimenti emergono due soggetti straordinari: **Nilde e Mentore**, i cui risultati sono stati sorprendentemente superiori alle aspettative, segnando un punto di svolta nel progetto Alter.

I due personaggi principali avranno diametralmente opposte all'inizio ma con l'approfondimento del loro rapporto, inizieranno a comprendersi e pian piano capire che una risposta sempre vera ed univoca alla loro domanda non esiste. Il dottor Harper è un personaggio conflitto dalle sue stesse convinzioni che fino al momento dell'esperimento credeva di fare la cosa giusta, per poi pentirsi delle scelte fatte fino a quel fatidico giorno.

All'interno del gioco sono presenti delle cutscene interamente prodotte usando l'intelligenza artificiale fornita da DaVinci - AI art generator[30] un sito per la generazione di immagini da testo. Ogni immagine è stata creata seguendo degli specifici comandi in base alla scena che si voleva raffigurare per poi essere montate tramite Adobe Premiere un programma di video editing.

4.3.6 Interazioni con l'ambiente

All'interno di alcuni dei livelli della demo è possibile praticare del parkour tra i vari elementi a schermo per evitare svariati nemici che se affrontati potrebbero portare alla morte del Player, inoltre in uno dei livelli sono presenti svariate porte che possono essere aperte e chiuse dal giocatore permettendogli una via di fuga sempre possibile con l'obiettivo di portarlo a scegliere tra l'affrontare una serie di nemici e oppure evitarli.

5.0 SPERIMENTAZIONE

Dopo aver creato una demo giocabile di A Mind Within You, è arrivato il momento di valutarla, facendola provare a diverse persone, e ottenendo opinioni sia sul gioco nel suo complesso sia sull'interfaccia di gioco e le interazioni con l'IA.

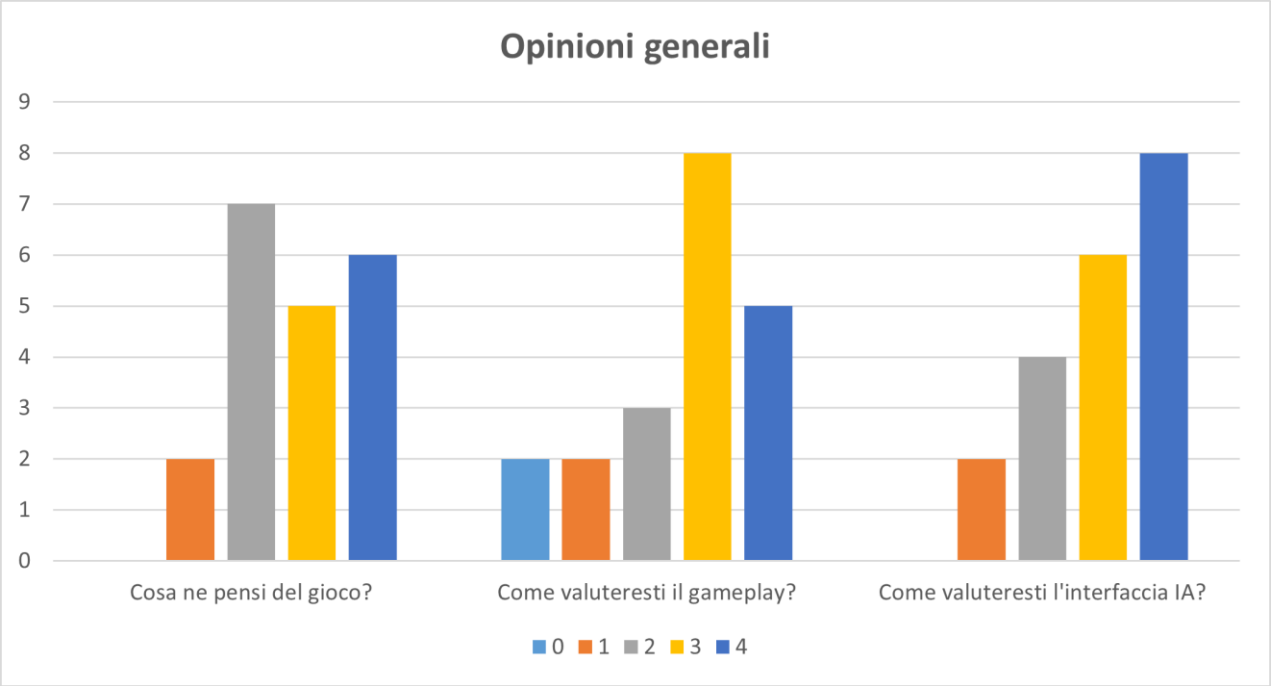
Per rendere l'interfaccia facilmente accessibile senza dover ogni volta riaffrontare tutti e 3 i livelli di intermezzo si è scelto di creare 2 possibili scorciatoie presenti nella primissima sezione di gioco che porteranno direttamente nel livello 4 della demo in cui è presente l'NPC con cui è possibile interagire. Una è una ventola poco fuori dalla cella iniziale e l'altra è una porta invisibile posta sempre all'interno della cella iniziale ma al limite del bordo sinistro.

Per la valutazione di A Mind Within You e dell'interfaccia si è creato un form su Microsoft Forms che contenesse delle domande abbastanza generali e delle domande specifiche del Game Experience Questionnaire, a cui hanno risposto 20 persone.

Nelle prossime pagine saranno analizzate entrambe le sezioni del form.

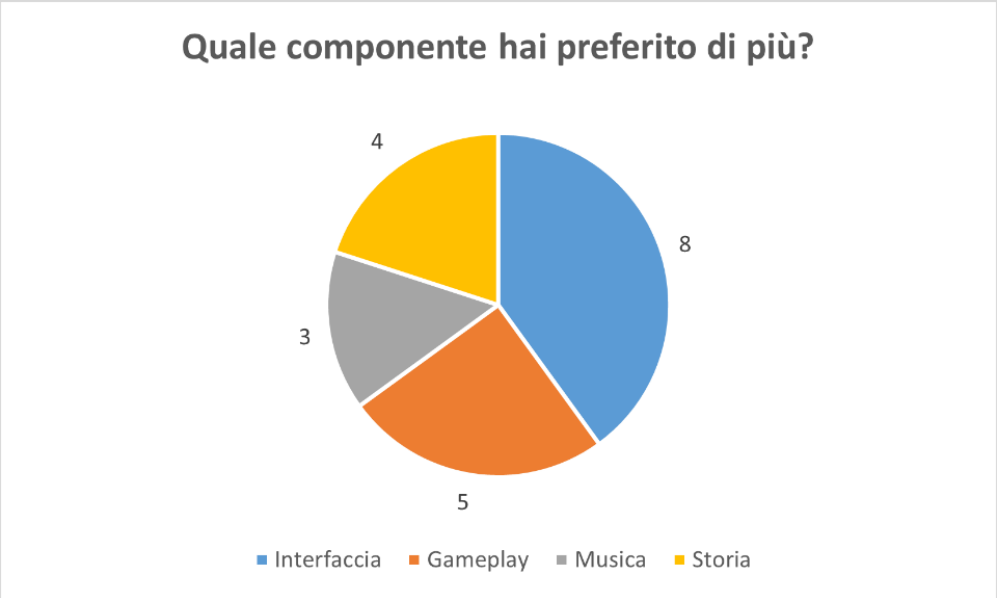
5.1 Valutazione: domande preliminari

Le domande iniziali serviranno come base per capire quali sono le componenti migliori e più apprezzate dagli utenti. Tutti e 20 gli utenti che hanno risposto alle domande rientrano nella fascia d'età tra i 18 e i 29 anni, e parlano come loro lingua madre l'italiano. Per prima cosa è stata chiesto un parere agli utenti a riguardo di quali sono le prime sensazioni sul gioco, sul gameplay, sull'interfaccia e un voto complessivo del gioco.



Dal sondaggio è emerso che il parere generale del gioco è abbastanza alto, facendo la media per ogni domanda il grafico ci mostra che alla domanda 'Cosa ne pensi del gioco?' si ottiene un 2,75 su 4; alla domanda 'Come valuteresti il gameplay?' si ottiene un 2,6 su 4; alla domanda 'Come valuteresti l'interfaccia IA?' un 3 su 4. La media generale di tutte le domande risulta 2,78 su 4.

Successivamente è stata richiesta un'opinione su quale fosse il componente che si è preferito durante la demo, tra le scelte: Interfaccia IA, Gameplay, Musica, Storia;



Si evince che l'Interfaccia è stata il componente più apprezzato con 8 preferenze su 20, seguendo il Gameplay con 5 preferenze su 20 e a seguire con la Storia e la Musica con 4 e 3 preferenze. In seguito è stato chiesto agli utente se fossero disposti a giocare una versione completa del gioco, la maggior parte di essi ha risposto positivamente.

5.2 Game Experience Questionnaire

Il Game Experience Questionario, anche detto GEQ[31], è uno strumento molto utilizzato nella ricerca in campo videoludico, e consiste in un questionario diviso in quattro sezioni, ciascuna volta a comprendere lo stato d'animo di un giocatore e le sue percezioni in momenti diversi del gioco. Oltre all'uso in ambito di ricerca, il GEQ è stato utilizzato da aziende che operano nel settore videoludico per valutare nuovi giochi ma anche giochi già esistenti e di successo, per poter comprendere l'esperienza di gioco die giocatori[].

A seconda dei risultati del GEQ, è possibile migliorare il gioco proposto, andando a risolvere dei problemi messi in evidenza da punteggi bassi per una data categoria. L'obiettivo di goni gioco è far divertire il giocatore, fargli passare del tempo lontano dal mondo reale, e come si vedrà nelle prossime pagine, il GEQ pone molta attenzione a questo fattore. Grazie alla suddivisione in "dimensioni" (il GEQ è diviso in più questionari, ed ognuno si basa su diverse "dimensioni" dell'esperienza di gioco) è possibile comprendere quali sono i punti di forza e i punti di debolezza del gioco proposto.

Il GEQ ha una struttura modulare che consiste in quattro moduli. Tutti e quattro vanno eseguiti immediatamente dopo la fine della sessione di gioco, nell'ordine:

- **Core Questionnaire**, che è la parte centrale, e valuta l'esperienza di gioco basandola su sette componenti: Immersion, Flow, Competence, Positive and Negative Affect, Tension e Challenge.
- **Social Presence Module**, che osserva il coinvolgimento psicologico e comportamentale del giocatore con altre entità sociali, siano esse virtuali (npc) oppure altri giocatori sia online che in locale.
- **Post-Game Module**, che valuta come si sentono i giocatori dopo aver finito di giocare, ed è utile per valutare se il gioco è stato volontario, ma anche in ricerche sperimentali.
- **In-Game Module**, una versione ridotta del Core Questionnare a cui il giocatore risponde più volte durante una sessione di gioco, in tempo reale.

Per questa valutazione sono stati utilizzati i moduli Core, Social Presence e Post-Game poiché, dato che il gioco è ancora una demo, non è abbastanza grande per interviste intermedie. Il modulo Social Presence in questo caso valuta l'interazione con l'interfaccia e l'IA.

Il questionario è stato fornito in inglese ovvero nella sua versione originale, ma il sistema di traduzione automatica fornita da Chrome è risultata molto utile per alcuni dei partecipanti, senza creare errori.

Ogni modulo del questionario è composto da delle affermazioni su diversi punti di vista e stati d'animo ed un punteggio da 0 a 4 da assegnare a ciascuna affermazione in base all'esperienza del giocatore. I punteggi corrispondono a:

Not at all: 0; Slightly: 1; Moderately: 2; Fairly: 3; Extremely: 4;

Ogni modulo, diviso in dimensioni, viene valutando calcolando la media aritmetica dei punteggi di ogni dimensione.

Affermazioni del modulo Core:

1. I felt content
2. I felt skilful
3. I was interested in the game's story
4. I thought it was fun
5. I was fully occupied with the game
6. I felt happy
7. It gave me a bad mood
8. I thought about other things
9. I found it tiresome
10. I felt competent
11. I thought it was hard
12. It was aesthetically pleasing
13. I forgot everything around me
14. I felt good
15. I was good at it
16. I felt bored
17. I felt successful
18. I felt imaginative
19. I felt that I could explore things
20. I enjoyed it
21. I was fast at reaching the game's targets
22. I felt annoyed
23. I felt pressured
24. I felt irritable
25. I lost track of time
26. I felt challenged
27. I found it impressive
28. I was deeply concentrated in the game
29. I felt frustrated
30. It felt like a rich experience
31. I lost connection with the outside world
32. I felt time pressure
33. I had to put a lot of effort into it

Affermazioni del modulo Social Presence:

1. I empathized with the other(s)
2. My actions depended on the other(s) actions
3. The other's actions were dependent on my actions
4. I felt connected to the other(s)
5. The other(s) paid close attention to me
6. I paid close attention to the other(s)
7. I felt jealous about the other(s)
8. I found it enjoyable to be with the other(s)
9. When I was happy, the other(s) was(were) happy
10. When the other(s) was(were) happy, I was happy
11. I influenced the mood of the other(s)
12. I was influenced by the other(s) moods
13. I admired the other(s)
14. What the other(s) did affected what I did
15. What I did affected what the other(s) did
16. I felt revengeful
17. I felt schadenfreude (malicious delight)

Affermazioni del modulo Post-Game Module:

1. I felt revived
2. I felt bad
3. I found it hard to get back to reality
4. I felt guilty
5. It felt like a victory
6. I found it a waste of time
7. I felt energised
8. I felt satisfied
9. I felt disoriented
10. I felt exhausted
11. I felt that I could have done more useful things
12. I felt powerful
13. I felt weary
14. I felt regret
15. I felt ashamed
16. I felt proud
17. I had a sense that I had returned from a journey

Linee guida per il punteggio del modulo Core:

- **Competence**: Punti 2, 10, 15, 17, e 21.
- **Sensory and Imaginative Immersion**: Punti 3, 12, 18, 19, 27, e 30.
- **Flow**: Punti 5, 13, 25, 28, e 31.
- **Tension/Annoyance**: Punti 22, 24, e 29.
- **Challenge**: Punti 11, 23, 26, 32, e 33.
- **Negative affect**: Punti 7, 8, 9, e 16.
- **Positive affect**: Punti 1, 4, 6, 14, e 20.

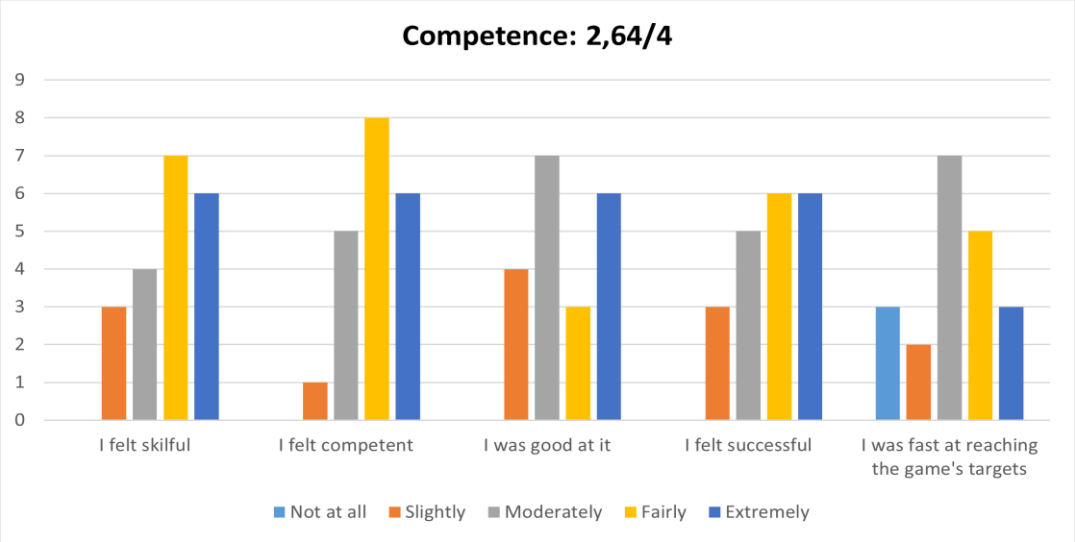
Linee guida per il punteggio del modulo Social Presence:

- **Psychological Involvement – Empathy**: Punti 1, 4, 8, 9, 10, e 13.
- **Psychological Involvement – Negative Feelings**: Punti 7, 11, 12, 16, e 17.
- **Behavioural Involvement**: Punti 2, 3, 5, 6, 14, e 15.

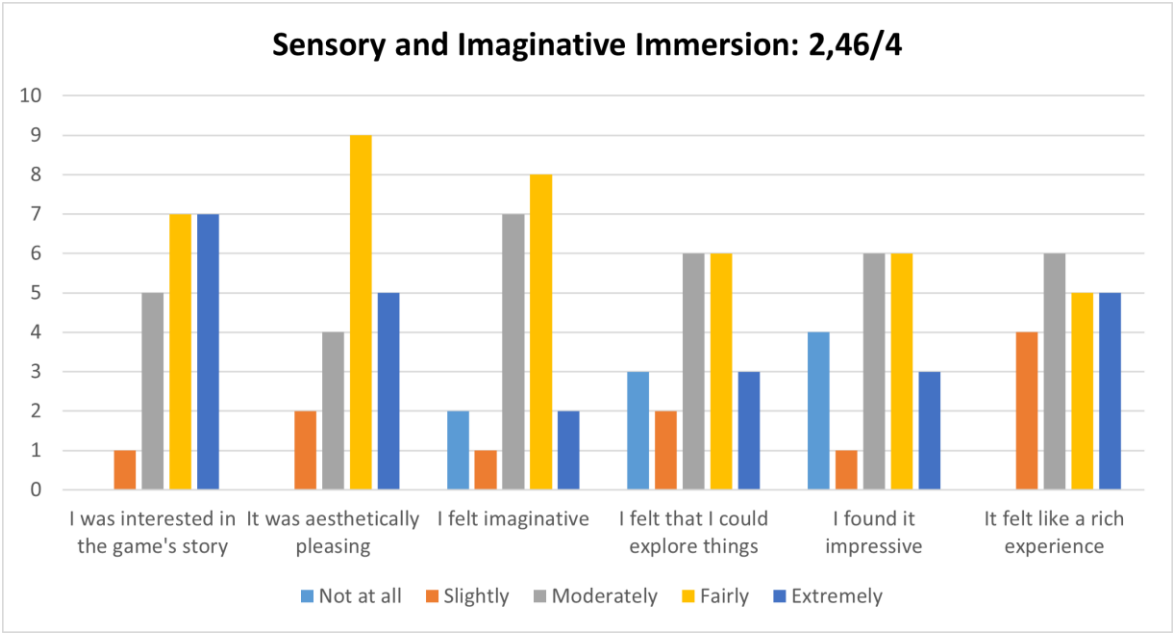
Linee guida per il punteggio del modulo Post-game:

- **Positive Experience**: Punti 1, 5, 7, 8, 12, e 16.
- **Negative Experience**: Punti 2, 4, 6, 11, 14, e 15.
- **Tiredness**: Punti 10, e 13.
- **Returning to Reality**: Punti 3, 9, e 17.

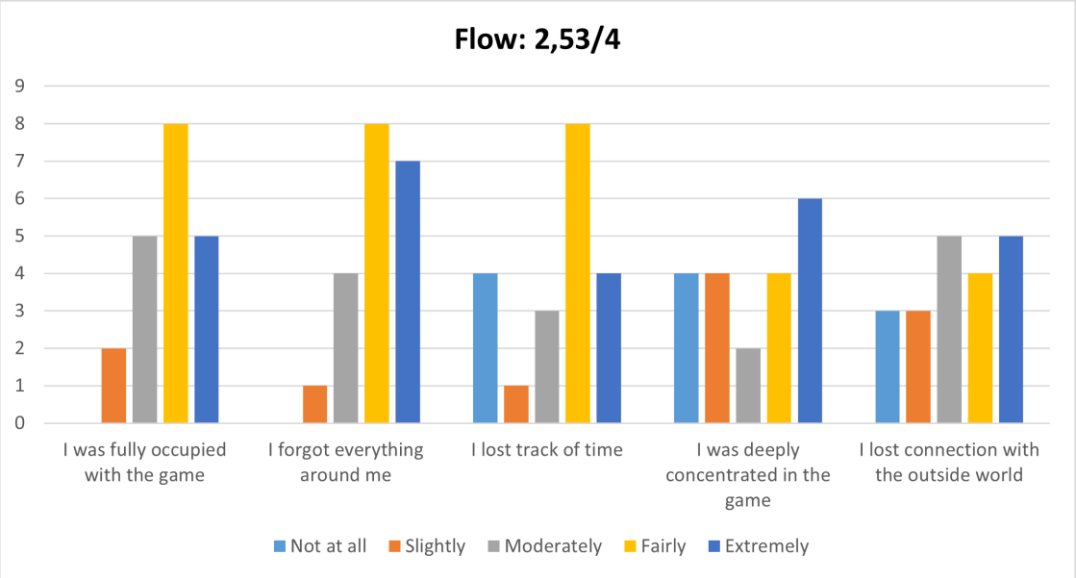
5.2.1 Risultati del modulo Core



Come si può vedere dal grafico i giocatori si sono sentiti mediamente competenti e gratificati dall'esperienza proposta ma in futuro sarebbe meglio aggiungere dei livelli di difficoltà per lasciare libera scelta al giocatore, dato che la maggior parte degli intervistati erano giocatori poco avvezzi.

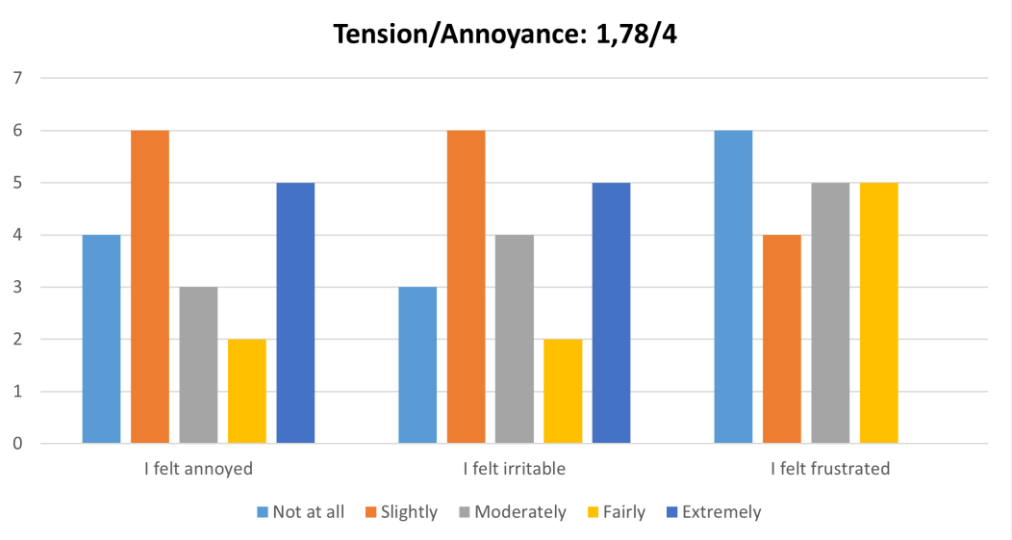


Come mostrato dal grafico molti giocatori hanno provato interesse nella storia del gioco, hanno trovato appagante lo stile grafico scelto e generalmente è stata considerata un'esperienza interessante.

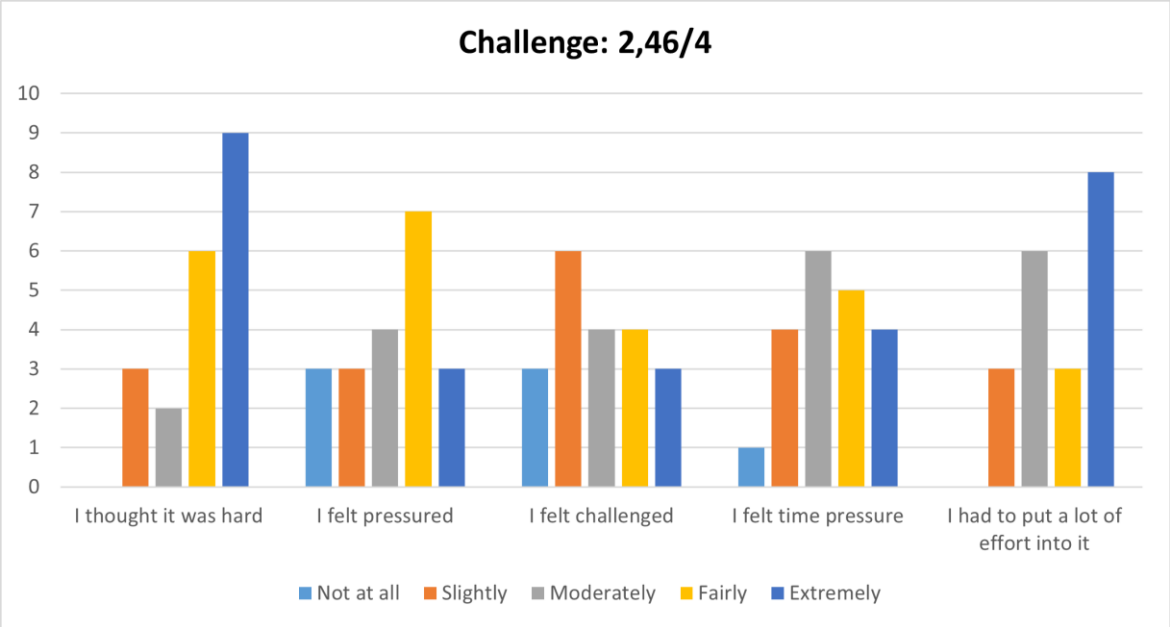


Parlando con alcuni degli intervistati è emerso che i giocatori più avvezzi ed esperti hanno completato la demo tra i venti minuti e i quaranta minuti, mentre per i giocatori meno esperti il completamento della demo senza l'uso delle scorciatoie ha richiesto anche trenta minuti di tempo ed essendo questo gruppo di giocatori quello più grande si spiega il motivo di così tante risposte di tipo "Fairly" ed "Extremely".

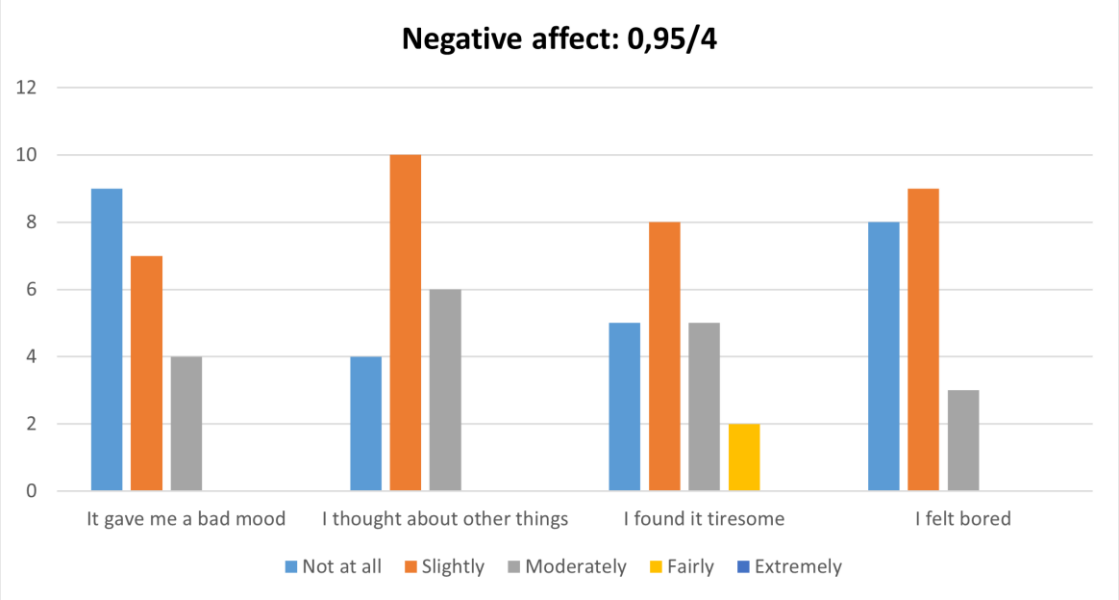
In contrasto con quanto riportato con il modulo Competence pare che anche se la maggior parte dei giocatori ha necessitato di più tempo per finire la demo per via della difficoltà, questa è stata comunque considerata come una cosa positiva in cui i giocatori hanno percepito il loro miglioramento man mano che giocavano.

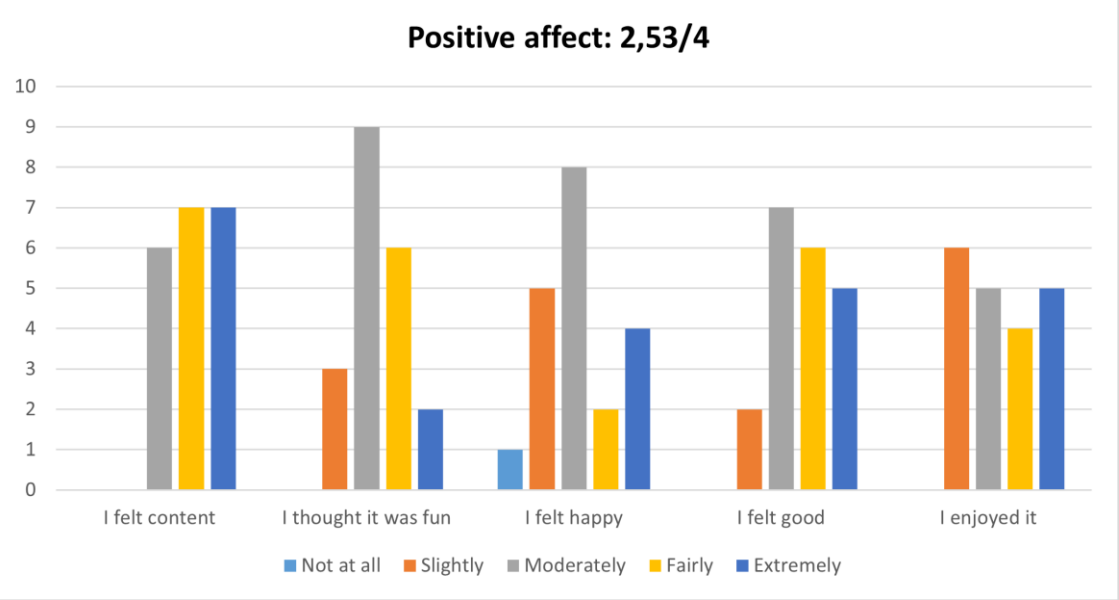


Infatti come mostrato dal grafico del "Tension/Annoyance" i dati sono abbastanza equilibrati, tra giocatori esperti che hanno provato noia nel completare troppo velocemente la demo e i giocatori novizi che invece hanno mostrato irritazione ma poca frustrazione nel continuare a rifare alcuni livelli.



Come confermato dal grafico la maggior parte dei giocatori (probabilmente quella dei novizi) ha trovato il gioco difficile tanto che ha richiesto loro maggior impegno. In futuro bisognerà puntare ad abbassare questo valore per rendere più accessibile il gioco.



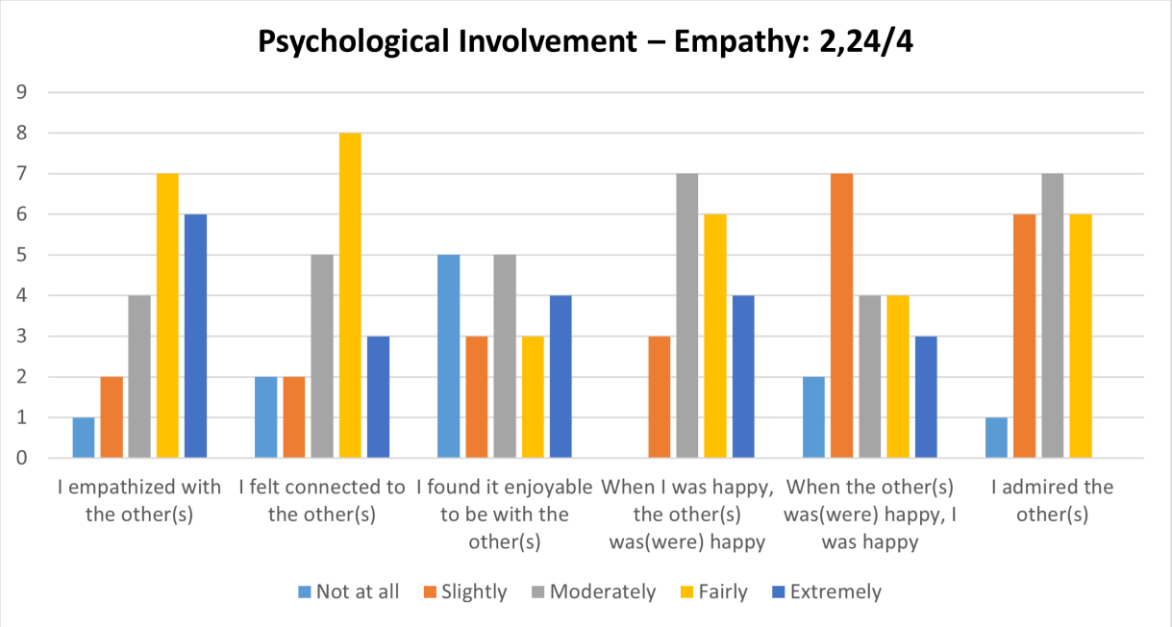


Alla fine come mostrato dai grafici nonostante l'alta difficoltà riscontrata per i giocatori l'esisto si è comunque dimostrato abbastanza positivo con giocatori non particolarmente entusiasti come mostrato dai campi "I thought it was fun" e "I felt happy" ma si è riusciti ad ottenere risultati molto positivi nel grafico "Negative affect" con la stragrande maggioranza dei giocatori che non si è annoiata.

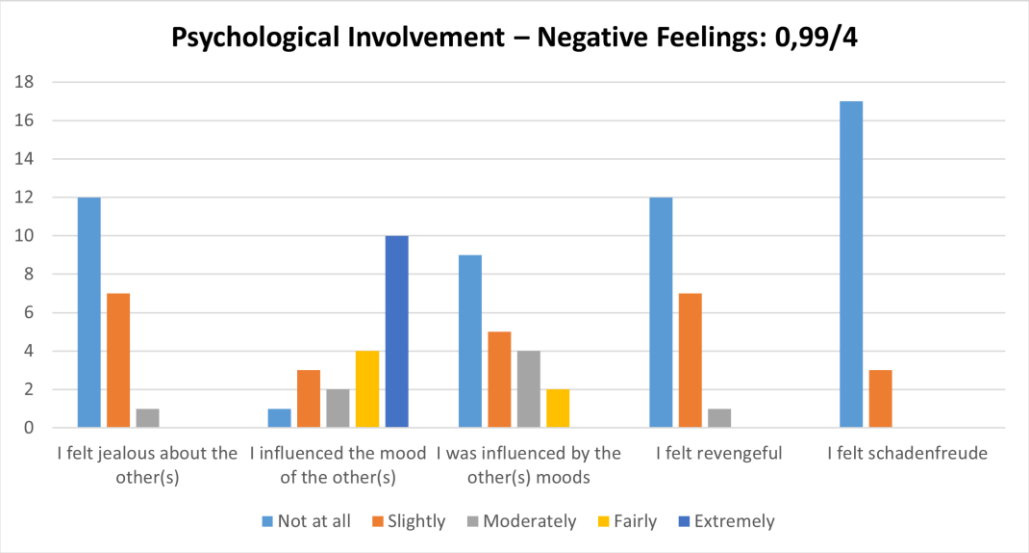
Da questo primo modulo possiamo quindi dire che il gioco necessita di un selettore di difficoltà per facilitare il gameplay ai giocatori meno esperti per evitare che questi possano continuare a trovare l'esperienza di gioco difficile, in modo tala da offrire la corretta difficoltà a chi vuole una effettiva sfida e chi invece vuole giocare con più calma e tranquillità.

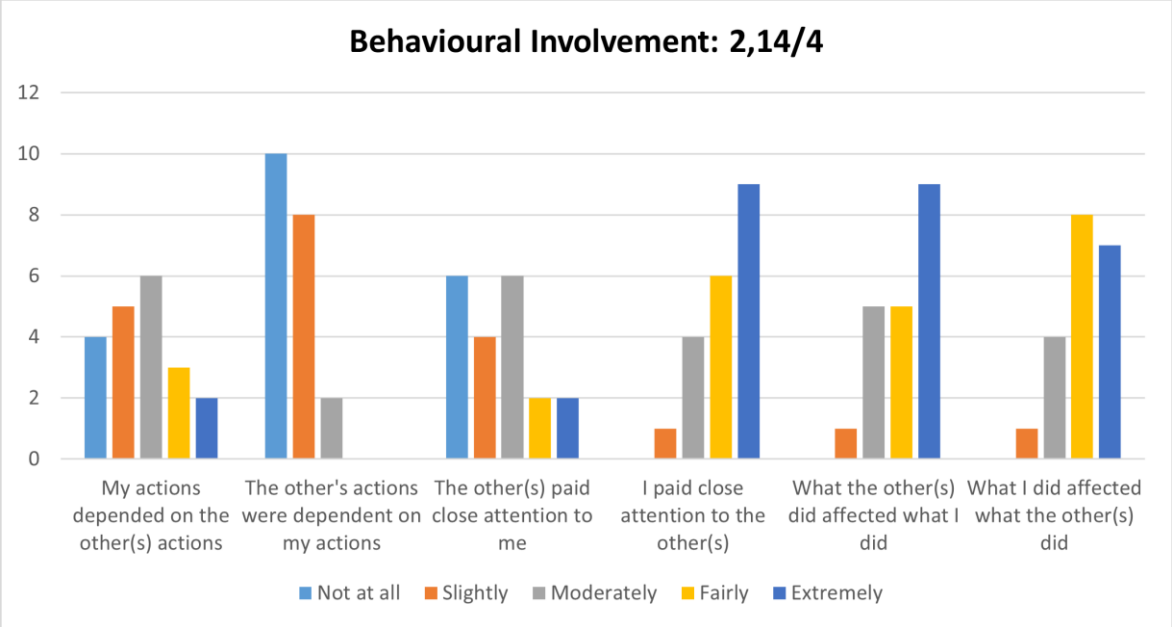
5.2.2 Risultati del modulo Social Presence

Nella prossima sezione, Social Presence, alcuni valori verranno mostrati ma non commentati poiché "A Mind Whitin You" è un gioco Single Player, ma bensì alcuni valori avranno bisogno di essere contestualizzati data la presenza di una IA che simula il comportamento di un NPC più cosciente rispetto ai tradizionali NPC ma



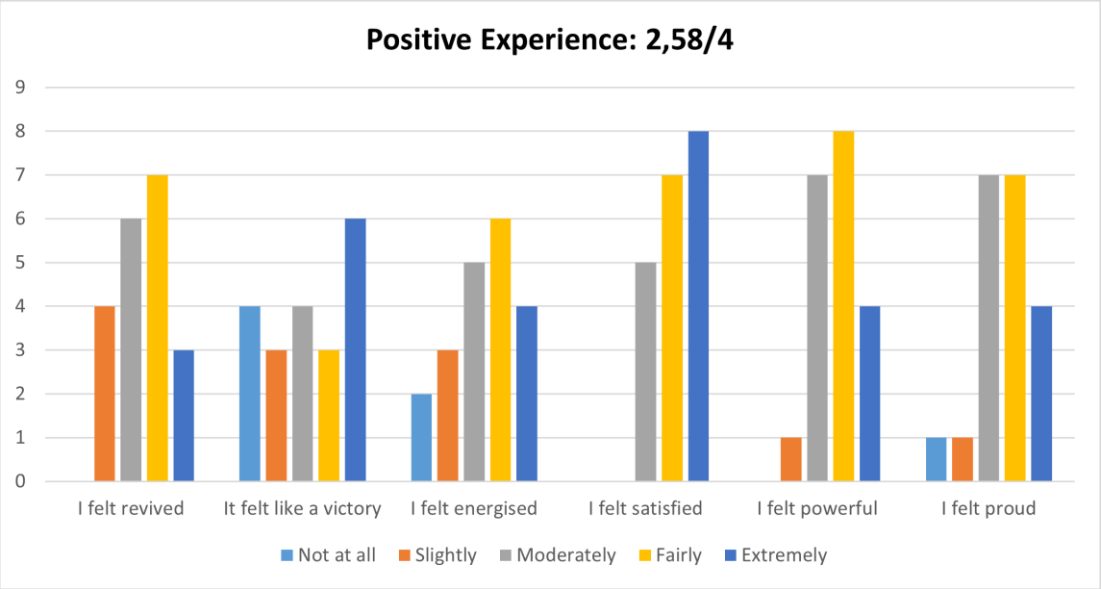
Da questo grafico possiamo intuire che i giocatori siano riusciti a mostrare un po' di empatia verso l'NPC e ciò ha permesso di ottenere valori più o meno alti nel resto dei campi, portando significativi risultati nel campo "I felt connected to the others".



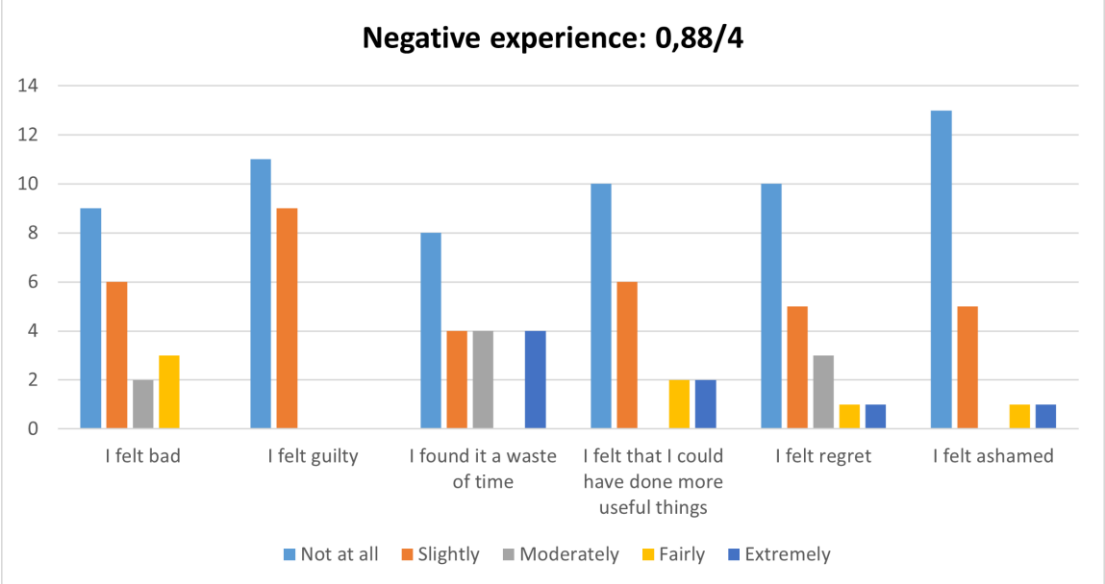


Come mostrato dal grafico negli ultimi due campi l'NPC è riuscito a adeguarsi bene alle risposte degli utenti lasciando una versa sensazione che le proprie azioni potessero in qualche modo interagire col modo di fare dell'IA.

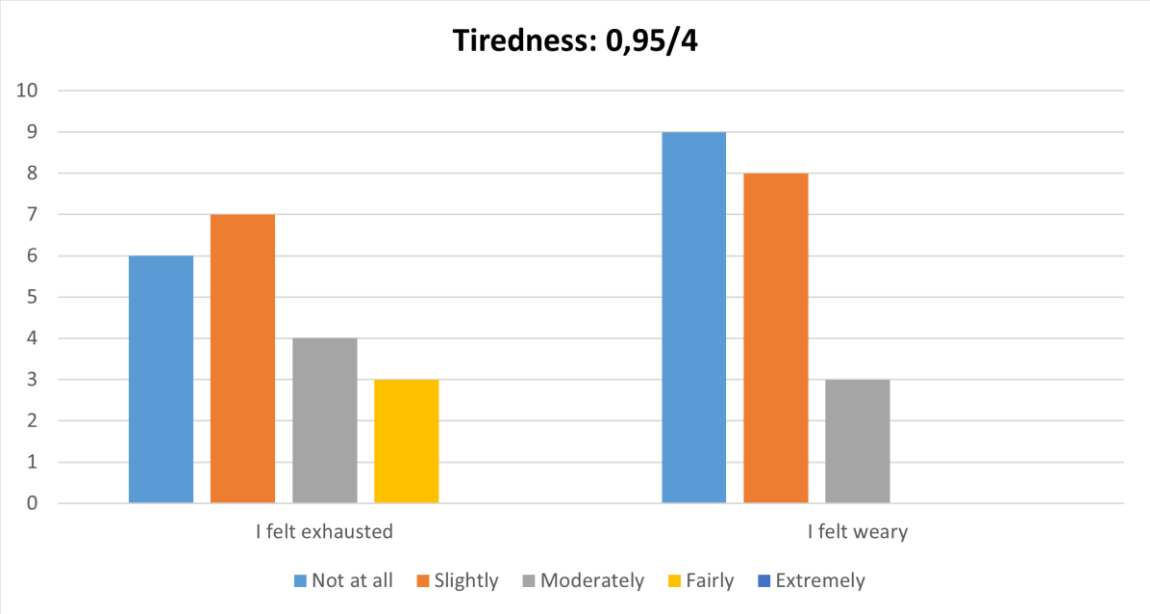
5.2.3 Risultati del modulo Post-Game



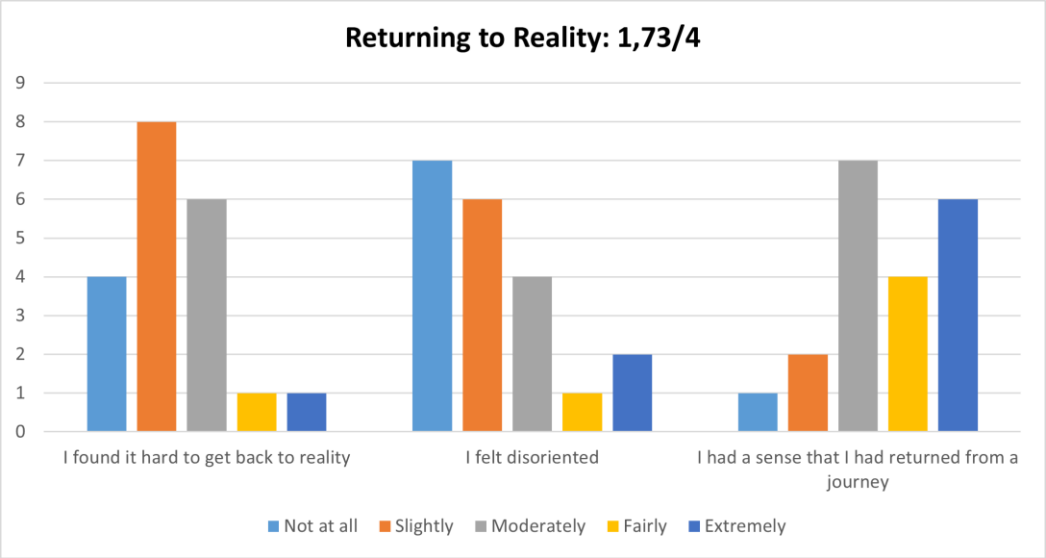
Come già analizzato nei precedenti moduli i valori del "Positive Experience" sono congrui a quelli ottenuti fino ad ora mostrando una grande soddisfazione da parte dei novizi nel completamento del gioco data la sua difficoltà.



Anche qui il grafico del "Negative Experience" ribadisce qualcosa di già detto precedentemente, ovvero che il principale problema del gioco è la sua difficoltà non tanto il suo gameplay o l'esperienza che ne deriva e i dati ottenuti da questi campi lo confermano, mostrando valori molto bassi in quasi tutte le affermazioni.



Considerata la difficoltà riscontrata nel gioco il numero di persone che hanno risposto "Not at all" all'interno del campo "I felt exhausted" è relativamente bassa per non parlare dei risultati del campo "I felt weary" che sembrano discordanti con appunto la difficoltà riscontrata nel gioco ma in linea con il risultato del questionario "Tension/Annoyance" del modulo Core.



Per ultimo questo questionario ci mostra che anche qui i dati si dimostrano un po' discordanti e non in linea con i risultati del questionario “Flow” nel modulo Core dove si era ottenuto un valore di 2,53 su 4 nel, da questo dato ci dovrebbe essere un risultato abbastanza simile anche in “Returning to reality” ma che in realtà non avviene se non per il campo “I had a sense that i had returned from a journey” che si dimostra essere quello più in linea con i risultati raccolti nel questionario “Flow”.

6. CONCLUSIONE E SVILUPPI

Nonostante le risposte ottenute con il Game Experience Questionnaire non siano troppo soddisfacenti per via di alcune incongruenze nelle risposte, si può dire che comunque il risultato rimane positivo, dato che è stata molto apprezzata l'interazione con l'IA e in generare anche il gameplay è stato soddisfacente nonostante i vari problemi già discussi.

Personalmente, sono soddisfatto del risultato finale, a parte le varie rifiniture riguardanti il gioco che sono ancora possibili da attuare per renderlo un'esperienza ancora migliore, migliorare l'esperienza di gioco risolvendo problemi e bug sulle impostazioni dell'audio e soprattutto ridurre i tempi d'attesa tra una risposta e l'altra.

Nel corso della progettazione, si è scelto di inserire una seconda IA che ha permesso di riprendere la meccanica del Karma originariamente scartata, e in futuro sarebbe possibile inserire molte altre opzioni e interazioni in cui è il Karma del giocatore a permettere il reclutamento di alleati facilitando il gameplay e migliorando il Flow del gioco e sfruttando le tecnologia già usate per la creazione dell'interfaccia ciò sarebbe molto semplice da creare.

Le possibilità di ampliazione del gioco tramite IA sono ancora molte e sarà curioso studiarne le applicazioni future.

RINGRAZIAMENTI

Ringrazio il professor Pierpaolo Basile per avermi fatto da relatore e soprattutto per avermi aiutato e consigliato nella scelta del progetto di tesi e soprattutto per avermi guidato nelle prime fasi di scrittura.

Un primo pensiero va a mio padre, che anche se non è più qua per poter assistere al raggiungimento di questo mio traguardo è sempre nei miei pensieri e lo ringrazio enormemente per tutti i sacrifici compiuti affinché io potessi arrivare fin qua. Un'enorme grazie a mia madre per avermi supportato e sostenuto in questi anni e di credere in me ogni volta che dovevo dare un esame e confortarmi quando uno di questi non andava come sperato.

Ringrazio tutti gli amici e compagni di corso che mi hanno aiutato in questo lungo percorso, da chi mi ha fatto compagnia in aula dal primo giorno da matricola a chi ha scritto con me gli ultimi programmi. In particolare voglio ringraziare Christian, Simone Capone, Simone De Girolamo e Thomas, grazie a voi i primi anni sono stati più leggeri da superare, di questo gruppo voglio ringraziare Tommy che mi ha fatto compagnia nei primi e pesanti mesi in cui si prendeva insieme l'autobus alle 6 del mattino per andare in università. Grazie mille a Marco e Antonio per non avermi confortato i primi anni da fuori corso alleviando la mia ansia e per i lunghi pomeriggi passati a programmare insieme.

In ultimo vorrei ringraziare tutto il gruppo di D&D per avermi aiutato a smorzare l’ansia e la tensione di questi ultimi anni e soprattutto un ringraziamento speciale va al mio caro amico d'infanzia Luca per avermi supportato e sopportato in questi anni, grazie mille per rimanere al mio fianco anche quando non mi faccio sentire per mesi e sparisco dalla circolazione.

BIBLIOGRAFIA

[1] "LA STORIA DELLE CONSOLE: DAL 1948 AL 2000 - Fumettomania Magazine on line." *Fumettomania Magazine on Line*, 2024. www.fumettomaniafactory.net/la-storia-delle-console-dal-1948-al-2000.

[2] "Doom.", *Wikipedia*. https://it.wikipedia.org/wiki/Doom.

S. Morris., M.Bittani., P.Ruffino., Doom, Giocare in prima persona, Costa & Nolan, 2005. https://books.google.it/

[3] M.Trovato. L'algoritmo "Ray Casting", 2003. http://ispace.altervista.org/files/RayCasting.pdf

[4] A.Belotta. "Complessità emergente - La raffinata intelligenza dei Fantasmi di Pac-Man — Outcast." *Outcast*, 2021. www.outcast.it/home/pac-man-intelligenza-artificiale?rq=Complessit%C3%A0%20emergente%20-%20La%20raffinata%20intelligenza%20dei%20Fantasmi%20di%20Pac-Man.

[5] *Metal Gear Solid Guida Completa: Capitolo 4 – Le Modalità Di Allerta – Bazaverse – Lo Shop Del Futuro*, 2024. https://bazaverse.com/2024/09/20/metal-gear-solid-guida-completa-capitolo-4-le-modalita-di-allerta

[6] Thompson T., Building the AI of F.E.A.R. with Goal Oriented Action Planning, 2023. https://www.gamedeveloper.com/design/building-the-ai-of-f-e-a-r-with-goal-oriented-action-planning

[7] Jeff Orkin. Three States and a Plan: The A.I. of F.E.A.R., GameDevs, 2006. https://www.gamedevs.org/uploads/three-states-plan-ai-of-fear.pdf

[8] Mićović N., Paper Insight: The A.I. of F.E.A.R., Nordeus Engineering, 2020. https://engineering.nordeus.com/https-www-youtube-com-watch-v-d8nrhltca9y/

[9] "*Development of the Elder Scrolls IV: Oblivion*", Wikipedia. https://en.wikipedia.org/wiki/Development_of_The_Elder_Scrolls_IV:_Oblivion

[10] Taylor L., Lost Features: A critical essay on TES IV: Oblivion's Radiant AI. Medium, 2024. https://medium.com/@gatherer286/lost-features-a-critical-essay-on-tes-iv-oblivions-radiant-ai-a0150144ddef

[11] "Radiant AI", Wikipedia.
https://en.wikipedia.org/wiki/Radiant_AI#:~:text=The%20Radiant%20AI%20system%20deals,determine%20how%20to%20achieve%20them

[12] Papadopoulos T., The Elder Scrolls IV: Oblivion" and the notorious radiant artificial intelligence, *Medium,* 2023.
https://thanasispapadopoulos.medium.com/the-elder-scrolls-iv-oblivion-and-the-notorious-radiant-artificial-intelligence-4996636473b

[13] Kelly A., Perfect organism: An Alien: Isolation companion, Unbound, 2024.
https://www.google.it/books/edition/Perfect_Organism/X98GEQAAQBAJ?hl=it&gbpv=0

[14] Thompson T., *The Perfect Organism: the AI of Alien: isolation*, Game Developer, 2024.
https://www.gamedeveloper.com/design/the-perfect-organism-the-ai-of-alien-isolation

[15] Thompson T., Revisiting the AI of Alien: isolation, Game Developer, 2024.
https://www.gamedeveloper.com/design/revisiting-the-ai-of-alien-isolation

[16] Kollar P., The Last Guardian review, *Polygon*, 2016.
https://www.polygon.com/2016/12/5/13767468/the-last-guardian-review-playstation-4-ps4-trico-sony-fumito-ueda-gendesign

[17] Joseph H., Cyberpunk 2077. The best-worst game ever made, Medium, 2022.
https://josephhadeed25.medium.com/fcyberpunk-2077-the-best-worst-game-ever-made-f69c711de650

[18] "*Videogioco indipendente*", Wikipedia.
https://it.wikipedia.org/wiki/Videogioco_indipendente

[19] *Godot Engine - Free and open source 2D and 3D game engine*.
https://godotengine.org/

[20] Samuelson G., PIXEL ART – THE MEDIUM OF LIMITATION, 2020.
https://www.diva-portal.org/smash/get/diva2:1518210/FULLTEXT01.pdf

[21] Giser P., Schertell M., Trojer L., MODERN PIXEL ART GAMES A study in GUI aesthetics for modern pixel art games, 2013.
https://www.diva-portal.org/smash/get/diva2:832803/FULLTEXT01.pdf

[22] *Piskel - Free online sprite editor*.
https://www.piskelapp.com/

[23] *Hugging Face – The AI community building the future.*
https://huggingface.co/

[24] *Qwen, Qwen/Qwen2.5-72B-Instruct · Hugging face*, Hugging Face, 2024. https://huggingface.co/Qwen/Qwen2.5-72B-Instruct

[25] "Hypertext Transfer Protocol", Wikipedia. https://it.wikipedia.org/wiki/Hypertext_Transfer_Protocol#:~:text=6%20giugno%202022.-,Funzionamento,cui%20risiede%20il%20sito%20web

[26] *Welcome to Flask — Flask Documentation (3.1.x)*. https://flask.palletsprojects.com/en/stable/

[27] Hsu, F., Hwang, Y., Tsai, C., Cai, W., Lee, C., & Chang, K.,). TRAP: a Three-Way handshake server for TCP connection establishment, *Applied Sciences, 2016*. https://doi.org/10.3390/app6110358

[28] *StreamPeerTCP* - Godot Engine Documentation. https://docs.godotengine.org/en/stable/classes/class_streampeertcp.html

[29] *Tabularisai, tabularisai/multilingual-sentiment-analysis, Hugging Face*, 2024. https://huggingface.co/tabularisai/multilingual-sentiment-analysis

[30] *DaVinci - AI Art Generator*. davinci.ai.

https://davinci.ai/

[31] IJsselsteijn, W.A.; de Kort, Y.A.W., The Game Experience Questionnaire., Poels, K. Eindhoven: Technische Universiteit Eindhoven, 2013.

https://research.tue.nl/en/publications/the-game-experience-questionnaire