POLITECNICO DI TORINO

**Laurea Magistrale in Ingegneria Informatica,
orientamento Grafica e Multimedia**

Politecnico
di Torino

Tesi di Laurea Magistrale

Gentle Mocap: animazione di personaggi 3D tramite Motion Capture all'interno di un prodotto videoludico multigiocatore

Relatore	Candidato
Prof. MARCO MAZZAGLIA	GIANLUCA GRAVANTE

Aprile 2022

“Mi hai dato la vita
per vivere come mi pare,
mi hai dato amore e
supporto per seguire i miei sogni.
La tua bellezza vive per
sempre nel profondo della mia anima,
il ricordo del tuo amore
riempie il mio cuore
e non sono mai solo”

Abstract

Questa tesi di laurea è parte di un progetto più grande di sviluppo collaborativo di un videogioco multiplayer in Unreal Engine. Il prototipo di gioco ha come personaggi principali degli insetti umanoidi dotati di abilità di difesa e attacco, a cui si vogliono applicare animazioni che mantengano gli aspetti fisici e abitudinali di queste creature. La tesi si incentra sullo sviluppo di "Gentle Mocap", un software di Motion Capture full body innovativo e gratuito. Esso permette di animare sia modelli 3D umani sia modelli 3D di creature dalle forme antropomorfe. Da un punto di vista economico, la tecnica generata consente ai rigger e agli animator di utilizzare il Mocap a costi molto bassi. L'innovazione rispetto alla tecnica tradizionale è di fatti l'assenza di utilizzo di tute dotate di sensori inerziali o l'ausilio di numerose telecamere che registrino i movimenti del corpo. Lo studio di "Gentle Mocap" consiste nel semplificare questi sistemi costosi e adattarli ad un hardware casalingo, alla portata di tutti, ovvero permettere la cattura di movimento umano tramite una singola telecamera. Il lavoro di progettazione è stato suddiviso in tre fasi:

1. **Realizzazione di un applicativo di Motion Capture:** sviluppo di un'applicazione in Python che permetta di registrare tramite un'unica telecamera i movimenti del corpo umano, del viso e delle mani per mezzo di algoritmi di Motion Capture.

2. **Sviluppo di un addon per il Rigging e animazione dei personaggi:** progettazione di un addon in Blender che elabori i dati registrati dalla cattura per animare le espressioni facciali e il movimento del corpo dell'insetto. Tramite gesture delle mani è possibile gestire parti esterne del corpo come il battito alare e il movimento mandibolare, le antenne e le zampe dell'insetto. Il risultato finale è un'animazione fluida e precisa dei personaggi.

3. **Gestione personaggi su Unreal Engine:** i modelli così animati, sono trasferiti sul motore di gioco e preparati all'utilizzo dei game programmer. E' prevista una fase di modifica dei parametri e scripting per adattare le animazioni allo stile di gioco, migliorandone le transizioni e la responsività di esecuzione dei movimenti.

Ringraziamenti

Vorrei dedicare questa pagina alle persone, che con il loro supporto, mi hanno sostenuto e aiutato in questo percorso di crescita professionale.
In primis, ringrazio di cuore la mia famiglia, che nonostante la distanza mi sono sempre stati accanto, ascoltandomi e facendo il tifo per me. Sono onorato di far parte di questa famiglia amorevole, che nonostante gli ostacoli della vita è e continuerà ad essere unita per sempre.
Ringrazio mia mamma, la mia guerriera, che ha sempre dimostrato fierezza e forza di volontà nel concedermi la possibilità di affrontare questo percorso senza mai farmi mancare nulla. Lei è nelle mie mani, nelle mie gambe e nel mio cuore. Abbiamo raggiunto il nostro traguardo insieme. Mamma, cercherò di seguire il tuo insegnamento e spero di renderti fiera di me. Mi manchi.
Grazie a mio padre, una persona unica e speciale che si è preso cura di me. Sei stato sempre presente con il tuo incoraggiamento. A te che al mattino ti alzi e vai a lavorare per non farmi mancare niente, dandomi sempre tutto ciò di cui ho bisogno. Grazie per i tuoi sacrifici, ora è arrivato il tempo di sdebitarmi.
Ringrazio mio fratello, la mia guida, il mio braccio destro. Abbiamo condiviso tutto, sei stato il mio primo compagno di giochi e ora che siamo cresciuti ho capito il vero significato del tuo ruolo nella mia vita: essere un fratello maggiore. Sei come un supereroe da cui traggo sempre spunto per migliorare, perchè so per certo che tutto ciò che apprendo da te, è giusto. Grazie per tutte le volte che mi sei andato contro, lo hai fatto per istruirmi. Sei il mio punto di riferimento.
Un grazie particolare va alla mia dolce cagnolina Cindy, che aspetta constantemente il mio ritorno, pronta a festeggiare.
Ringrazio infinitamente il mio relatore Mazzaglia Marco che è stato per me fonte d'ispirazione per la realizzazione dell'elaborato. Una persona disponibile e amichevole che mi ha trasmesso l'emozione del concetto di videogioco. Grazie per le indicazioni e i preziosi consigli per la stesura della tesi.

Indice

Elenco delle figure

Capitolo 1

Introduzione

Che aspetto avrebbero i videogiochi senza animazioni?
Tutto risulterebbe privo di vita, statico.
L'animazione è un mezzo importante per rendere una scena di gioco quanto più realistica possibile, o addirittura permette di andare oltre le normali leggi della fisica.
Buona parte del lavoro del rigger e dell'animator viene svolto sui personaggi.
Il Rigging è una tecnica importante che permette di costruire uno scheletro animabile, composto da una serie di ossa interconnesse che alterano una mesh.
Ogni personaggio possiede delle caratteristiche che lo contraddistinguono da altri.
Può avere un movimento fluido o rigido, correre più veloce o saltare più in alto.
Ogni dettaglio, varia in base allo stile di gioco, conferendo uno stato e un'emozione che vengono trasmessi a chi controlla il personaggio.
Il videogiocatore avrà la sensazione di essere teletrasportato in un altro mondo.
L'avanzamento tecnologico ha introdotto alcune tecniche nell'ambito videoludico, che aiutano gli animatori a muovere dei corpi con precisione e a tempo ridotto.
La tecnica che ha preso il sopravvento su altre è il Motion Capture, un processo che, attraverso la cattura di movimenti di un soggetto fisico reale, permette la ricostruzione dei movimenti e delle espressioni facciali su dei corpi virtuali.
Il metodo di registrazione più utilizzato consiste nell'indossare una tuta con dei marcatori che trasmettono la loro posizione in coordinate cartesiane al software.
Questi dati vengono elaborati e applicati su di un modello 3D con lo scopo di replicare la cattura.
Tuttavia, il suddetto sistema presenta un costo rilevante che può essere sostenuto unicamente da parte di aziende videoludiche con elevata disposizione finanziaria.
Sono richieste infatti tute dotate di sensori inerziali e/o multiple telecamere per riprendere il soggetto; a tutto ciò, si aggiunge il costo del software per la gestione della cattura.
Questa tesi ha lo scopo di fornire ai Rigger e agli animator un tool gratuito che

replichi nella maniera più fedele possibile i risultati di un Motion Capture, ma senza l'ausilio di un'attrezzatura onerosa e senza la necessità di acquistare una licenza software per poterne usufruire. Il fine è dunque di dare la possibilità a chiunque di creare un prodotto videoludico con animazioni precise ed efficaci in modo gratuito. Si tratta di una soluzione semplice e alla portata di tutti, che richiede unicamente un comune hardware domestico. Per registrare i movimenti è sufficiente una singola fotocamera, persino quella di un normale laptop. Al contempo, per convertire i dati di cattura in animazione del personaggio, bisogna usufruire del software open source Blender e, in particolare, dell'apposito addon creato. Quest'ultimo permetterà il Rigging automatico, ovvero la creazione di uno scheletro che si adatta a un personaggio dalle sembianze antropomorfe. Conseguentemente sarà generata l'animazione del corpo basandosi sulla struttura dello scheletro impostato e infine le espressioni facciali saranno replicate sul modello 3D.
L'addon è uno strumento che dà al rigger la possibilità di customizzare la dimensione dello scheletro, perfezionare la lunghezza e la posizione dei segmenti scheletrici per adattarli alla struttura fisica del personaggio. La mesh del viso viene invece gestita tramite l'utilizzo delle Shape Key, anch'esse personalizzabili, per ottenere una combinazione di più espressioni facciali in contemporanea. Infine, le gesture delle mani saranno utilizzate per controllare parti esterne al corpo e nel caso specifico del presente progetto, sull'animare il battito alare, il movimento mandibolare e le antenne di un determinato insetto.

1.1 Motivazione

Sin dall'infanzia sono sempre rimasto affascinato ed attratto da figure che prendevano vita sullo schermo del mio televisore, da strane creature di cui ignoravo l'esistenza ma che potevano esistere in quanto inventate e trasposte con disegni e animazioni su un semplice display. Ho sempre avuto il desiderio di far parte di questi luoghi di fantasia e questo mi ha spinto ad esplorare e approfondire quelle ingegnosità che hanno reso l'immaginazione umana una realtà. Oggi sono in grado di comprendere come avviene la realizzazione di questi mondi animati, grazie ad un appassionato percorso di studi in Ingegneria Informatica al Politecnico di Torino e in particolare dell'orientamento di Grafica e Multimedia. Ho sviluppato conoscenze approfondite nello sviluppo dei videogiochi, della modellazione 3D e della cinematica animata, per merito dei corsi di Informatica Grafica, Computer Animation, Realtà Virtuale e Game Design. La passione e la curiosità in questi ambiti mi hanno spinto a redigere una tesi di lavoro su un prodotto videoludico multigiocatore, principalmente legato all'aspetto delle animazioni dei personaggi tramite l'utilizzo di una tecnica di Motion Capture alternativa a quella utilizzata nei videogiochi moderni. Il tipo di Mocap realizzato, oltre ad essere gratuito, offre

la possibilità ai ragazzi che si approcciano a questo nuovo mondo di usufruire di questa tecnica di animazione mantenendo comunque un'ottima accuratezza dei movimenti, pur non disponendo di strumentazione completa e onerosa. La peculiarità di questa soluzione è che l'animazione si adatta con un buon risultato, non solo a personaggi 3D umani, ma anche a modelli dalle forme antropomorfe. Lo scheletro del personaggio è modificabile in tutte le sue parti, adattandosi bene anche a corpi sproporzionati, in quanto la lunghezza e la posizione dei segmenti corporei possono essere personalizzate dai rigger senza influire sulla qualità del Motion Capture.

1.2 Outline

I capitoli di questa tesi sono suddivisi come segue:

- **Capitolo 2 - Stato dell'arte:** in ambito videoludico, viene analizzata l'evoluzione dell'animazione a partire dalla vecchia grafica 2D fino al 3D odierno, esaminando le varie tecniche che hanno reso i personaggi tridimensionali ricchi di movimenti animati.
- **Capitolo 3 - Prodotto videoludico multigiocatore:** descrizione del videogioco prettamente da un punto di vista grafico. Esposizione del ruolo di Technical Artist ricoperto all'interno del team e trattazione delle attività secondarie affrontate durante lo sviluppo.
- **Capitolo 4 - Studio correlato:** ricerca e innovazione per la creazione di "Gentle Mocap", un sistema di Motion Capture alternativo, che dia risultati ottimi e simili alla tecnica tradizionale, ma con l'ausilio di una singola telecamera.
- **Capitolo 5 - Motion Capture:** sviluppo di un'applicazione in Python che permetta di eseguire sessioni di Motion Capture che vadano a registrare i movimenti del corpo, i gesti delle mani e le espressioni facciali.
- **Capitolo 6 - Rigging e animazione personaggi:** programmazione di un'addon in Blender che converta i dati di Motion Capture in animazione sul personaggio. Si parte dalla generazione di uno scheletro e si conclude con l'applicazione del movimento e delle espressioni facciali sui modelli 3D realizzati.
- **Capitolo 7 - Gestione personaggi su Unreal:** trasferimento su Unreal Engine dei personaggi 3D e le relative animazioni. E' prevista una fase di ottimizzazione dei parametri di movimento e di fluidificazione della transizione tra un'animazione e l'altra.

Capitolo 2

Stato dell'arte

L'animazione è diventata una parte essenziale all'interno dei videogame. C'è una continua ricerca di animazioni che possano integrarsi bene nella narrazione ed essere responsive al tipo di gameplay. Ottenere dei movimenti digitali realistici o fantascientifici permette al videogiocatore di sentirsi parte dell'ambiente circostante e vivere un'esperienza di cui ne è il protagonista. Il risultato ottenuto oggi nella computer animation, deriva dallo studio di tecniche innovative. La tecnica moderna che è in grado di fornire animazioni di alta qualità è il Motion Capture, ovvero la cattura dei movimenti di un'attore che vengono poi trasferiti sul personaggio di gioco. La tecnica MoCap nasce nei primi anni '80 con l'esigenza di apprendere e analizzare il funzionamento dei muscoli in movimento di umani e animali. I primi utilizzi erano nel settore medico e militare; lo scopo iniziale era valutare la prestazione fisica o studiare i problemi di abilità motorie di una persona e intervenire con adeguate terapie. L'utilizzo del Motion Capture spazia nei primi anni '90 fino a raggiungere l'industria cinematografica, dando il via alla creazione di film animati. La strada che però ha portato verso questa nuova tecnologia in ambito gaming è stata complicata a causa dei limiti hardware imposti dai dispositivi di elaborazione di gioco. La storia racconta come la computer grafica e l'esperienza nel settore videoludico abbia fatto un balzo in avanti affrontando tre diverse tipologie di animazione: 2D, 2D/3D, 3D.

2.1 Animazione 2D

Le prime animazioni 2D adottano una tecnica molto semplice, dove le immagini che vengono renderizzate dal computer corrispondono a dei fotogrammi che visualizzano un elemento in posizioni diverse dando l'impressione di un movimento. Questa tipologia di animazione risale al 1958, quando Willian Higinbotham creò un semplice videogioco di tennis che non venne però commercializzato. Ai tempi si utilizzavano

delle forme geometriche di base come rettangoli o sfere che venivano renderizzate su schermi in bianco e nero dotati di pochi pixel. L'animazione bidimensionale consisteva nella traslazione sull'asse verticale e orizzontale di questi elementi.

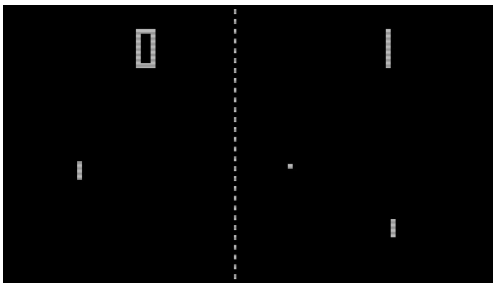


Figura 2.1: Pong

Il padre di tutti i videogiochi è "Pong". Si tratta di un simulatore di ping pong prodotto dalla Atari nel 1972, dove il giocatore controlla verticalmente una barra bianca con l'obiettivo di colpire la palla e lanciarla nel campo avversario. I movimenti sono basici e molto rigidi.

Un accenno di animazione articolata si ha dieci anni dopo, quando nel 1981 viene sviluppato il famoso Donkey Kong su schermo a colori. La caratteristica di questo videogioco è l'inserimento di alcune pose associate sia a Kong che a Mario. Kong possiede ad esempio le animazioni di presa e lancio del barile, mentre Mario è in grado di camminare o salire per le scale.

Figura 2.2: Donkey Kong

Figura 2.3: Dragon's Lair

Nel 1983 l'animazione 2D fu completamente stravolta quando venne creato "Dragon's Lair", un videogioco basato su filmati in stile cartoonesco che vengono riprodotti in sequenza in base alla decisione intrapresa dal giocatore.

La tecnica di animazione utilizzata per Dragon's Lair è una tecnica cinematografica che prende il nome di **Animazione Tradizionale** o Cel Animation. Si tratta di una serie di disegni, che fotogramma per fotogramma portano alla generazione di un filmato finale. In genere non si disegna daccapo ogni ambiente, ma lo sfondo rimane fisso mentre si muove solo il personaggio, come in questo caso.

I personaggi iniziano a prendere vita. Abbiamo la conferma di ciò quando nel 1987 viene commercializzato "Legend of Zelda", un gioco di avventura che spinge ulteriormente le animazioni dotando il personaggio di abilità, mosse di attacco e introducendo dei nemici anch'essi responsivi nell'ambiente di gioco.

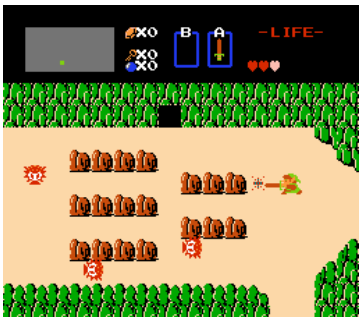


Figura 2.4: Legend of Zelda

2.2 Animazione 2D/3D

Il passaggio dall'animazione bidimensionale a quella tridimensionale ha incontrato diverse problematiche in quanto le console dell'epoca non possedevano ancora la potenza di calcolo necessaria per poter gestire numerosi poligoni in scena. Sebbene nel 1994 la Sony mise in commercio la Playstation quale primo avvicinamento verso il mondo 3D, gli sviluppatori dovettero limitare le proprie idee in quanto per realizzare qualcosa di godibile si è dovuti scendere a compromessi dal punto di vista grafico. Sul piatto della bilancia si presentavano due alternative: realizzare un gameplay 3D avvincente con ottime meccaniche e animazioni a discapito di un aspetto grafico scadente, oppure ritornare alla vecchia grafica 2D abbandonando amaramente la tridimensionalità. Viste le scomode alternative, un compromesso tra le due tecniche ha preso piede: il finto 3D .

Figura 2.5: Wolfenstein 3D

Il primo a introdurre questo nuovo approccio è "Wolfenstein 3D", uno sparatutto in prima persona del 1992 che è in realta un 2D puro. Si cerca di dare un senso di profondità e tridimensionalità tramite lo studio di ambienti ad hoc e un movimento del giocatore che è simil-3D. Le aspettative aumentavano e questo passo ha quindi rappresentato l'incipit verso l'innovazione grafica e dell'animazione nell'ambito del vero 3D.

L'uscita di "Resident Evil 1" nel 1996, rappresenta un mix discreto tra 2D e 3D. Per venire incontro alle limitazioni hardware, gli sviluppatori hanno deciso di mantenere dei background bidimensionali fissi e ben dettagliati, per lasciare computazione di calcolo alla renderizzazione e animazione dei personaggi 3D, che si potevano muovere in questo spazio scenografico bidimensionale. La differenza tra i personaggi dalle forme poligonali e un ambiente retrostante piatto ma accurato, non influisce sulla giocabilità e la libertà di movimento, ma crea solo una leggera dissomiglianza a livello grafico.

Figura 2.6: Resident Evil 1

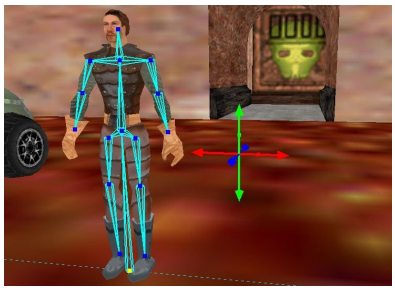


Figura 2.7: Skeletal Animation

Figura 2.8: Face Texture Animation

Il movimento dei personaggi 3D sfrutta la **Skeletal animation**, ovvero si genera uno scheletro adatto al personaggio, dove lo spostamento o rotazione di un osso influenza la mesh. I rigger e gli animatori hanno utilizzato questa tecnica muovendo manualmente lo scheletro frame per frame, dando delle posizioni chiave agli arti per poter ottenere l'animazione finale.
Per quanto riguarda invece le espressioni facciali non si parla ancora di rig facciale. Il **Face Texture Animation** è la tecnica utilizzata per dare vita al viso di un personaggio. Ma in realtà questo viso risulta piatto in quanto esso si anima tramite il cambio di texture. Il passaggio da un'espressione triste a una felice, oppure la chiusura e apertura degli occhi, consiste in una permuta di immagini. Il risultato è discreto e accettabile, ma è sicuramente lontano dal realismo.

Le aspettative dei giocatori accrescevano man mano che la grafica veniva implementata, prospettando un livello di animazione dei personaggi sempre più vicino alle richieste.

2.3 Animazione 3D

Figura 2.9: Super Mario 64 (Menu)

Il pioniere dell'animazione e grafica 3D è Super Mario 64, un'esclusiva pubblicata nel 1996 per il Nintendo 64. L'innovazione portata da questo videogioco si può notare già dal menu principale, dove viene data la possibilità di manipolare la mesh del viso di Mario come allungare le orecchie, il naso o chiudere gli occhi. L'obiettivo principale era quello di mostrare la potenza del Nintendo 64. Difatti, l'alterazione del viso di Mario era solo possibile nel menu, ma durante il gameplay il tipo di animazione facciale era di gran lunga inferiore, poichè il gioco sarebbe risultato pesante.

La tecnica utilizzata per la gestione del volto nel menu è il **Morph Target Animation**. Viene assegnato al modello un'espressione base e per ogni emozione si associa una Shape Key differente in grado di alterare la mesh. La combinazione di più Shape Keys genera l'espressione facciale finale.

L'animazione fece un balzo in avanti anche per quanto riguarda il movimento del corpo, tramite l'applicazione della tecnica "Skeleton Animation". Erano finiti i giorni in cui un semplice salto corrispondeva ad un'unica posa rigida. Mario ha un set di 200 animazioni che gli permettono di correre, saltare, nuotare e addirittura volare. Le braccia e le gambe del personaggio si muovono con fluidità dando al videogiocatore un'esperienza migliore. Super Mario 64 ha spianato le strade verso il 3D e in particolare sullo sviluppo di tecniche innovative che possano rendere le animazioni precise e variegate.

Negli anni successivi, inizia a prendere piede una delle tecniche di animazione digitale più utilizzata oggi: il **Motion Capture**. Questa innovazione viene incontro agli animatori che erano costretti ad animare i personaggi in modo manuale, una tecnica molto macchinosa che richiedeva tempo e precisione. Con il Mocap avviene invece una cattura realistica del movimento e delle espressioni facciali del soggetto in ripresa. Il corpo di un attore viene ricoperto di riferimenti, i cosiddetti marker, in genere di materiale riflettente o emissivo, applicati su aree chiave del corpo come mani, gomiti, ginocchia e piedi. I movimenti dei marcatori sono registrati da speciali videocamere e poi trasferiti sul modello del personaggio creato digitalmente. Lo stesso sistema si può applicare anche al viso dell'attore per ottenere delle espressioni facciali quanto più veritiere possibile.

Le origini di questa innovazione risalgono al 1994, quando fu pubblicato uno tra i primi picchiaduro che sfrutta l'animazione 3D dei personaggi: Virtua Fighter 2. La novità del gioco fu quella di registrare i set di mosse dei personaggi tramite motion capture e applicarle ai modelli tridimensionali. Il successo di questo titolo incoraggiò altre case di produzione a investire nella nuova tecnologia.

Figura 2.10: Virtua Fighter 2

Questa tecnica inizialmente era resa possibile solo grazie a sofisticate, scomode tute ricoperte di sensori collegati tramite cavi lunghi e ingombranti. Oggi, invece abbiamo a disposizione algoritmi sempre più performanti e accurati e una strumentazione adeguata ad abbattere questi problemi. Per le produzioni a basso budget vi è inoltre la possibilità di sostituire il processo di marcatura con una metodologia più semplice, sfruttando i colori sulla pelle o palline da ping-pong per sostituire i marker.
Lo studio del Motion Capture ha reso il livello di animazione dei personaggi molto alto, paragonabile ai film cinematografici e questi risultati si possono apprezzare con il lancio di nuove console come la Playstation 3 e l'Xbox 360.

Figura 2.11: L.A. Noire

Un titolo molto apprezzato del 2011 che propone un'esperienza di gioco con cinematiche e dettagli espressivi facciali nei personaggi è "L.A. Noire". La storia ha a che fare col mondo dell'investigazione, con la presenza di detective e criminali. Sono previsti interrogatori, dove i giocatori devono essere in grado di leggere le espressioni facciali dei testimoni per riconoscere tra bugia e verità.

A tal scopo la Rockstar Games ha introdotto un sistema di motion capture alternativo a quello tradizionale: il **Motion Scan**. 32 telecamere ad alta definizione circondano e riprendono il viso di ciascun attore, catturando tutte le espressioni facciali che saranno poi trasferite come animazioni sul volto dei personaggi 3D. L'avvento della Playstation 4 e Xbox One spinge ancor di più l'aspetto grafico e animato dei videogiochi.

Figura 2.12: Ryse: Son of Rome

L'esclusiva del 2013 per Xbox One è "Ryse Son of Rome" che rappresenta un connubio perfetto tra motion capture e cinematica. L'obiettivo è ricreare la storia di un comandante romano a capo di un esercito contro un'invasione barbarica e quindi lo scopo è trasportare il giocatore in quell'epoca.

Il Motion Capture ha svolto un ruolo importante in questa grande opera, dove gli attori hanno replicato i movimenti, le mosse e le tecniche di battaglia utilizzati dai legionari. I sensori di cattura non erano solo sugli attori ma erano posti anche sulle armi o scudi per poterne registrare con precisione gli spostamenti. Il risultato finale è una storia avvincente con animazioni mozzafiato.

Capitolo 3

Prodotto videoludico multigiocatore

Lo scopo è sviluppare un prototipo avanzato di un videogioco multiplayer in real-time. La presente tesi tratta di un progetto di gruppo svolta assieme a due studenti magistrali del Politecnico di Torino. Il team, dal nome 3Bugstards, ha lavorato su tre fronti principali, ciascuno dei quali supervisionato e curato da uno dei tre membri. I tre ambiti sono di seguito elencati:

- **Tech Art:** caratteristiche estetiche del videogioco.
  - Modellazione
  - Animazione
  - UI Design
  - VFX
  - Audio design
  - Shader Programming
- **Engineering:** caratteristiche funzionali del videogioco.
  - Gestione del multiplayer
  - Ottimizzazione
  - Collezione e pulizia dei dati di gioco
  - Gestione dell'IA
  - Gameplay Programming
- **Design:** figura essenziale che fa da regia nella creazione del prodotto e si occupa del design del videogioco.

3.1 Descrizione

Il prodotto videoludico realizzato è "BugBall", un RTS-sportivo, ovvero uno strategico in tempo reale che richiede ai giocatori capacità meccaniche e strategiche, individuali e di squadra. L'ispirazione proviene dal Dodgeball, ma con la modifica di gran parte delle regole. Il campo della partita è suddiviso in due aree, ogni area ospita una squadra di tre giocatori. Trattasi dunque di un 3 vs 3, ove l'obiettivo è spingere fuori i componenti del team avversario per accumulare punti e vincere la partita. Ogni giocatore eliminato deve aspettare un tempo di respawn per tornare in campo. Le partite sono a tempo e si svolgono al meglio dei 3 incontri, pertanto si può ottenere una vittoria matematica concludendo in trionfo 2 match su 3. Le condizioni di vittoria del singolo match sono le seguenti:

- Eliminare tutti e tre gli avversari prima dello scadere del tempo.
- Allo scadere del tempo, aver totalizzato un numero di punti maggiore della squadra avversaria
- In caso di pareggio, si giocheranno i tempi supplementari e vincerà la squadra che eliminerà tutti i giocatori avversari (un giocatore eliminato durante i supplementari non può rientrare)

La peculiarità del videogioco è il concept dei personaggi realizzati, rappresentati da insetti umanoidi. Ognuno di essi ha caratteristiche diverse dall'altro, può essere più difensivo, più portato all'attacco o equilibrato. In aggiunta, il personaggio ha a disposizione tre slot abilità e un'abilità passiva unica che lo contraddistinguono. Le abilità sono varie e tra le più importanti ci sono gli attacchi speciali contro l'avversario (Skillshot) o attachi ad area (AoE), dando anche la possibilità di fornire supporto al team con dei potenziamenti (Buff).
Prima dell'inizio di ogni match la squadra ha a disposizione del tempo per pensare ad una strategia e impostare il set di abilità, scegliendo quelle più appropriate per scardinare il team avversario.

3.2 Ruolo primario

3.2.1 Technical Artist

Il ruolo di primo piano svolto all'interno del team è rappresentato dal Technical Artist, che fa da collegamento tra gli artist e i game programmer. E' una figura professionale nell'industria videoludica che acquisisce svariati significati, ma che ha come funzione principale l'integrazione di risorse artistiche/visive per rendere

il prodotto finale dotato di un'estetica che affascini il giocatore. Questa professione richiede non solo conoscenze grafiche ed artistiche, ma anche padronanza dell'ambito tecnico inerente alla programmazione, al rigging e all'animazione. Per la realizzazione del prodotto videoludico multigiocatore, è indispensabile il suo ruolo di "ponte tecnico" tra reparto animazione e sviluppatori, assicurando un'integrazione tra le animazioni e le risorse, senza sacrificare la visione artistica.

3.3 Ruoli secondari

Lo svolgimento della tesi prevede in aggiunta all'attività di Technical Artist, l'impiego di tutti i ruoli inerenti alla sezione tecnico-artistica del videogioco. Trattandosi di un piccolo team di tre persone, è stato ritenuto necessario che ognuno ricoprisse dei ruoli secondari, descritti qui di seguito.

3.3.1 Modeler e Texture Artist

Il 3D modeler si occupa di fornire gli asset che saranno utilizzati all'interno del progetto, tra i quali i personaggi, le ambientazioni e l'oggettistica. Per il prototipo avanzato sono stati modellati sei personaggi, nel dettaglio degli insetti umanoidi che si scontreranno tra di loro nel campo di gioco. Si tratta di una realizzazione personalizzata che permette di avere come risultato finale dei modelli ottimizzati animati tramite Motion Capture. Il primo step è stato realizzare un concept art andando a disegnare l'idea di personaggio che si ha in mente, cercando di mantenere le caratteristiche fisiche che contraddistinguono lo stesso, nello specifico caso un insetto. Avendo come riferimento la bozza, il passo successivo è stato creare l'appropriata copia tridimensionale. Per far questo è stato utilizzato il software di modellazione Blender, seguendo degli step ben precisi:

1. **Sculpt Mode:**

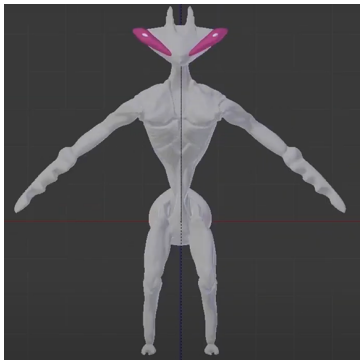


Figura 3.1: Sculpt mode
Verts: 82.134
Faces:162.258
Tris:164.258

La prima fase è stata quella di dare una forma iniziale al personaggio, assegnando dimensioni adeguate alla corporatura. Nella modalità "Sculpt", si è perfezionata la modellazione dell'insetto sfruttando i brush, ovvero gli strumenti messi a disposizione da Blender per alterare la mesh nei minimi dettagli. Il risultato ottenuto è ad una risoluzione molto alta ma è costituito da decine di migliaia di vertici e da una geometria non omogenea, in quanto la mesh è triangolarizzata (notare i valori registrati nella *figura 3.1*)

2. **Retopology:**

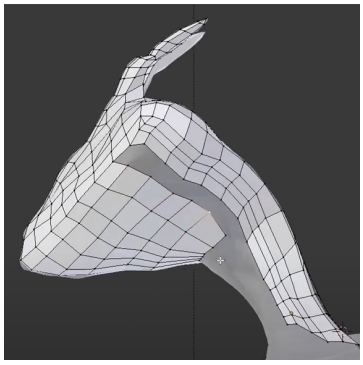


Figura 3.2: Retopology
Verts: 2.773
Faces:2.769
Tris:5.387

La retopology ha permesso di ricreare la superficie generando una geometria ottimale con un numero ridotto di poligoni. L'obiettivo è stato quello di creare una mesh pulita che sostituisse le facce triangolari con facce quadrate che si adattano perfettamente alle forme curve e piatte del personaggio (*figura 3.2*). Questo processo prende il nome di quadrangolazione della mesh e apporta vantaggi anche sulla colorazione e animazione del modello. Dopo la fase di retopology è stato ritenuto necessario applicare un modificatore di "Subdivision Surface" per poter suddividere ulteriormente il modello, in modo tale da avere una superficie più liscia del personaggio.

3. **Paint Texture:**

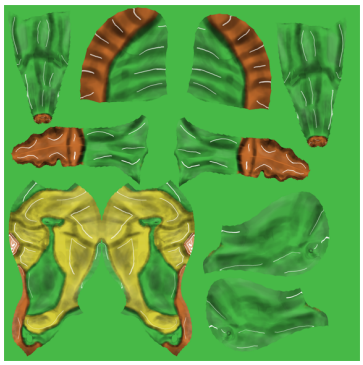


Figura 3.3: Diffuse Texture Resolution: 2048x2048

L'ultimo step è la colorazione dell'insetto modellato. Tramite il processo di Unwrap il personaggio viene sezionato in parti distinte. La texture realizzata ha una risoluzione in pixel in potenza di 2 (*figura 3.2*), adeguata a contenere tutte le informazioni sul colore applicato ad ogni punto della mesh. E' essenziale questa accortezza in quanto Unreal Engine applica tecniche di compressione che dimezzano la risoluzione della texture in base all'esigenza in game.

Adottando queste procedure di modellazione e applicazione della texture, sono stati creati sei insetti dalle sembianze antropomorfe utilizzati come personaggi all'interno del videogioco. L'estetica dei modelli 3D è rappresentata nelle figure sottostanti:

Figura 3.4: Formica

Figura 3.5: Ape

Figura 3.6: Scarabeo

Figura 3.7: Coccinella

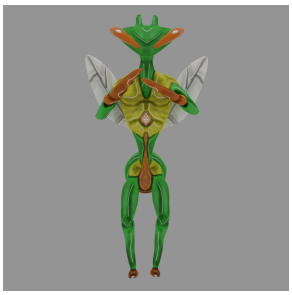


Figura 3.8: Mantide

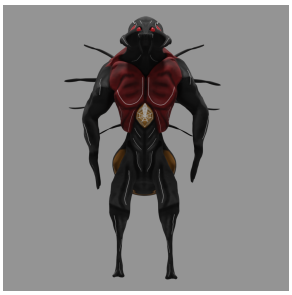


Figura 3.9: Ragno

3.3.2 Rigger e Animator

Il rigger è il responsabile della creazione dello scheletro, le cui ossa con il loro movimento animeranno parti specifiche del corpo e in questo specifico caso dell'insetto. L'animator ha il compito di sfruttare l'interconnessione di segmenti scheletrici generati dal rig, per poter animare e render vivo il personaggio. Questi due ruoli rappresentano le mansioni principali della tesi, con lo scopo di animare i modelli 3D tramite il processo di Motion Capture (*Capitolo 5* in poi).

3.3.3 VFX Designer

Il Visual Effects Designer si occupa dell'inserimento in game di effetti speciali. Nel caso di questo prodotto videoludico è stata condotta una ricerca di asset gratuiti di test che si adattassero alle abilità di cui gli insetti sono dotati. Gli effetti hanno subito modifiche dei parametri per rispettare le tempistiche delle skill e per generare una forma particellare adeguata all'idea finale. Si tratta però di effetti temporanei che saranno poi sostituiti da VFX appositi. Nelle figure sottostanti sono presenti esempi di applicazione dei visual effects:

Figura 3.10: Dash

Figura 3.11: Raggio

Figura 3.12: Cura

3.3.4 Lighter e Environment Artist

L'Environment Artist si occupa di modellare ambienti interni ed esterni del videogioco. Nel prototipo è prevista solo la creazione dell'ambiente esterno. E' stata realizzata una sola arena di testing, ma per la versione finale di gioco sono previste più arene. A tal fine, è stato necessario utilizzare asset di prova, con l'obiettivo di comporre la struttura del campo e del perimetro di gioco. Il pavimento è stato modellato secondo dimensioni precise rispettando i limiti dell'area dove i giocatori si muoveranno. Inoltre, sono state assegnate delle texture scure per permettere un buon contrasto di visione tra pavimento e personaggi/effetti speciali. Altri modelli dell'environment provengono invece da "Edit Finch", un package free del marketplace di Epic Games contenente edifici e oggettistica per realizzare il background dell'area di gioco.

Il Lighter svolge invece il ruolo di impostare l'illuminazione dell'ambiente. L'arena è irradiata da una sorgente di luce principale, la "Light Source" che rappresenta digitalmente il sole. In aggiunta è presente una seconda luce che è la "Sky Light", l'illuminazione atmosferica che aggiunge ombre e riflessi più realistici.

Figura 3.13: Arena

3.3.5 UI Designer

Il User Interface Design è l'attività di progettazione visiva e interattiva dell'interfaccia utente. Il videogioco multiplayer in analisi è stato dotato di un UI di sistema che permetta al giocatore di navigare al suo interno. L'UI di sistema si compone di due parti:

1. **Menu principale:** è l'interfaccia con cui si interagisce all'esecuzione del gioco. L'interfaccia è molto semplice e intuitiva, dotata di una cinematica propria. Durante la navigazione la telecamera si sposta in punti diversi dell'arena di gioco in base alla selezione di quattro sezioni differenti:

    - PLAY: e' la schermata che permette al giocatore di cercare un match disponibile. Verrà visualizzata una lista di sessioni disponibili e una volta selezionata una, il giocatore si metterà in attesa dell'avvio della partita, mentre una barra di caricamento segnalerà l'attesa su quella sessione. I match disponibili sono continuamente aggiornati, quindi si potrà stimare il tempo di avvio della partita osservando il numero di giocatori in attesa

su quel match. Prima dell'avvio della partita l'utente interagirà con due sezioni in sequenza:

(a) *SELECT A CHAMPION:* : il player avrà modo di selezionare il personaggio da utilizzare in game. Il personaggio dovrà essere impostato entro un certo limite di tempo. Inoltre, la lista di personaggi disponibili si aggiornerà in base alla scelta dei due compagni di squadra. Ad esempio se un compagno di squadra sceglie un personaggio dalla lista, questo sarà bloccato e l'utente non potrà selezionare lo stesso. Sono visualizzati i modelli 3d dei personaggi scelti dal proprio team in tempo reale, in modo da dare un'associazione grafica all'utente.

Figura 3.14: UI: Selezione del personaggio

(b) *SELECT ABILITIES:* dopo aver scelto il personaggio si passa alla selezione delle abilità. Ogni personaggio avrà a disposizione un pool di abilità, tra quali potrà sceglierne solo tre. Questo è l'ultimo step che porta poi allo start del game.

- CHAMPIONS: costituisce una wiki dei personaggi disponibili in gioco, per dare la possibilità all'utente di avere una conoscenza base dei campioni. Si possono visualizzare informazioni dell'insetto come il nome, statistiche di gioco (armatura, resistenza, ...) e le relative abilità con descrizione.
- OPTIONS: permette di impostare i setting legati al comparto grafico e audio. Lato grafico si possono modificare le impostazioni visive come la qualità delle texture, ombre, post processing secondo cinque modalità: Low, Medium, High, Epic, Cinematic. Lato sonoro, si può gestire il volume di gioco tra classi differenti del mixer audio: Ambient, Effects e Music. Applicate le modifiche, viene effettuato un salvataggio delle impostazioni, in modo tale che al riavvio del gioco esse vengano mantenute come all'ultimo settaggio.
- QUIT: si esce dal gioco, l'applicazione viene chiusa.

2. **Menu di pausa:** durante la partita vi è la possibilità di accedere al menu di pausa per poter ritornare al menu principale o modificare le impostazioni.

3.3.6 Sound Designer

Il designer del suono si occupa della generazione di soundtrack, effetti ambientali e sonori che possano essere importati nel prodotto videoludico. Il comparto audio è stato gestito tramite una raccolta di suoni messi a disposizione dal web non coperti da alcun copyright. Il prototipo avanzato, oggetto della tesi, dispone di feedback sonori per permettere all'utente di percepire l'avvio di un'abilità o per ascoltare cosa accade nell'ambiente di gioco.

Capitolo 4

Studio correlato

La storia ci insegna come l'animazione nei videogame evolve continuamente verso un livello di realismo completo. E' un viaggio che ha inizio negli anni '70, con animazioni rigide e minimali, fino ad arrivare ai giorni d'oggi, dove il gioco è senza intoppi e il personaggio è fluido alla pari di film completamente animati. L'animazione infonde vita ai personaggi, come gli esseri umani che respirano e vivono, ma con la possibilità di compiere movimenti al di fuori di ogni immaginazione.
La tecnica principale che ha reso possibile questo balzo in avanti è il Motion Capture. Lo studio del prototipo videoludico della tesi ha come obiettivo finale la generazione di animazioni che abbiano un livello conforme alla categoria di videogiochi tripla A, che sfruttano anch'essi la registrazione dei movimenti di un attore. Si tratta di animazioni precise, realistiche o immaginarie, che a livello visivo trasmettono emozioni al giocatore. Con questa premessa, lo stato attuale della tesi ha lo scopo di analizzare e modificare la tecnica tradizionale, per realizzare un'alternativa di Motion Capture full body a costo zero e alla portata di tutti: **Gentle Mocap**. La parola Gentle, dall'inglese gentile, indica come questi tool, in modo generoso, offrano al comparto di rigging e animazione un metodo intuitivo ed efficace per poter ottenere animazioni 3D simili a quelle derivanti dal Motion Capture utilizzato nei videogiochi odierni. Il software sviluppato, non richiede tute indossabili, né una miriade di telecamere ad alta definizione che registrino l'attore. Gentle Mocap fa sì che la cattura avvenga da una singola telecamera, senza la necessità di un sensore di profondità dedicato. Tramite algoritmi di machine learning, la cattura avviene in tre step differenti:

- Riconoscere i punti chiave del corpo e stimarne la posizione.
- Individuare dei riferimenti sul viso per gestire le espressioni facciali.
- Identificare gesti della mano per il controllo di parti esterne del corpo.

I movimenti registrati dalla telecamera vengono succesivamente elaborati sul software di modellazione, in questo caso Blender, dando vita al personaggio 3D.
Gli animator e i rigger possono apportare modifiche sullo scheletro del modello per perfezionare il risultato finale dell'animazione. La procedura di realizzazione del software, comporta una ricerca e analisi di tecniche di rigging ottimali da applicare ai modelli 3D animabili. Lo studio correlato alla tecnologia di Motion Capture e Rigging dei personaggi richiede la valutazione accurata di tre punti specifici:

- **Espressioni facciali:**
  l'umano è in grado di modificare e alterare molti muscoli facciali per dare una propria espressione rappresentante la sua emotività. A riguardo, alcuni videogiochi sfruttano il face capture per animare il viso dei personaggi. Il punto su cui si sofferma la tesi è la conservazione del realismo delle espressioni facciali rilevando pochi ed efficaci landmark del viso. Catturare pochi landmark, però, non dovrà risultare uno svantaggio nella precisione di movimenti. L'obiettivo è quello di gestire con accuratezza l'apertura/chiusura della bocca, il battito degli occhi e il movimento del naso, sfruttando un numero discreto di punti di riferimento e facendo sì che non si noti la differenza dal catturare un numero elevato di landmark. Questo è molto utile in quanto nei software di modellazione (es: Blender), gestire poche Shape Key, permette di animare il viso di un modello 3d in un tempo rapido. Inoltre, in ambito gaming, ha il vantaggio di effettuare una buona animazione del personaggio, mantenere una certa fluidità del gioco e non pesare troppo sull'elaborazione di animazione facciale in real-time. L'idea del prototipo di gioco, si presta bene alla tecnica di motion capture facciale, in quanto è utilizzata non solo per animare un modello di viso umano, ma anche di un insetto che ha delle proprie conformazioni facciali. La sfida è far sì che ogni tipo di insetto sviluppato adatti il proprio viso a quello umano. Ad esempio, l'apertura/chiusura degli occhi di un insetto dovrà adattarsi agli occhi umani, il muso dell'insetto corrisponderà al naso di un umano, le sopracciglia di una persona gestiranno il movimento delle antenne di un insetto, ecc... Questo adattamento sarà poi modificabile e perfezionato dal rigger e dall'animator in base alle proprie scelte e a quale tipo di risultato si vuole ottenere. Il tool prevede infatti la possibilità di personalizzazione delle espressioni facciali.

- **Cattura del corpo:**
  il tracking del corpo si occupa di registrare il movimento delle gambe, del busto, delle braccia e dell'inclinazione della testa. Quando il motion capture viene convertito in animazione sul personaggio, sono applicate delle ottimizzazioni qualora il movimento risulti "oscillante", ovvero la posizione di un punto del corpo cambia rapidamente in frame vicini provocando una sensazione di tremolio. Il metodo di smooth è la procedura messa in atto per smussare i

movimenti e renderli fluidi. I punti del corpo che vengono persi/non rilevati durante il capture vengono invece stimati, difatti il software si occupa di ricostruire un'azione discretamente valida. Inoltre, si sfrutta il sistema di Inverse Kinematics in Blender, col quale, si stima una posa adeguata tramite parametri matematici applicati su un'armatura flessibile e articolata. In riferimento al videogioco basato sugli insetti, bisogna adattare il corpo dell'umano alla forma umanoide dell'insetto. Le zampe che possono essere corte o lunghe faranno riferimento alle braccia/gambe dell'umano sulle quali si è fatto il capture. Il rigger, in Blender, avrà a disposizione un tool che gli permetterà di impostare la dimensione dell'armatura e la grandezza di ogni singolo osso, in modo tale da adattare la corporatura umana alle diverse forme degli insetti.

- **Tracking delle mani:**
  il rilevamento delle mani e quindi la cattura dei gesti fa da controllo alle parti "esterne" dell'insetto: le ali, le antenne, le mandibole e altre appendici del corpo. Per esempio, il movimento delle ali di un'ape può essere animato tramite gesture delle mani, con la chiusura o apertura di tutte le dita per gestire l'intensità del battito alare. Un altro esempio è legato al ragno, che nel retro della schiena ha 8 zampe (4 a destra, 4 a sinistra) ed esse si animano contando quante dita sulla mano sono alzate o meno, controllando una coppia simmetrica di zampe. Le mandibole dello scarabeo, invece, si chiudono/cambiano dimensioni posizionando le dita a forma di pinza e in base alla distanza tra pollice e indice si anima l'allungamento/chiusura delle zanne. La formica è dotata di due antenne che cambiano lunghezza, si abbassano e si alzano in base al gesto delle "virgolette", eseguito con dito indice e medio.

Capitolo 5

Motion Capture

5.1 Applicazione in Python

Il primo tool di Gentle Mocap si occupa del processo di registrazione dei movimenti del corpo umano, delle espressioni facciali e del tracciamento delle mani. Il linguaggio di programmazione utilizzato è Python, supportato dall'ambiente di sviluppo integrato PyCharm. La libreria principale che ha permesso il tracking full body è MediaPipe. Per la realizzazione dell'interfaccia utente (GUI) sono stati, invece, utilizzati i moduli Tkinter e PIL, mentre per il processamento di immagini e video si è impiegata la libreria cv2. Per la gestione e l'avvio di una sessione di Motion Capture, sono stati programmati quattro script differenti suddivisi come segue:

- **MocapUI.py:** in qualità di interfaccia utente iniziale, corrisponde al menu principale che compare all'avvio dell'applicazione e rimanda all'apertura di nuove finestre di esecuzione per la cattura del corpo, delle mani o del viso.

- **Mocap_Face_Mesh.py:** rappresenta lo script che si occupa di raccogliere informazioni sulla mesh del viso che viene alterata dalle espressioni facciali.

- **Mocap_Body.py:** è il modulo inerente alla registrazione dei movimenti del corpo, che effettua un tracking di punti di riferimento come gomiti, spalle, ginocchia.

- **Mocap_Hand.py:** è il codice esecutivo di cattura di una singola mano che stima la posizione delle articolazioni nello spazio.

Lo sviluppo del tool in questione è descritto, come segue, in due sezioni: l'interfaccia utente e la struttura elaborativa che esegue il Motion Capture.

5.2 Interfaccia utente

La GUI è stata studiata appositamente per offrire un'interfaccia veloce, intuitiva e con pochi passaggi da seguire per poter realizzare sessioni di Motion Capture.
Le dimensioni delle finestre di interazione hanno una dimensione in pixel che va dai 400x525 per il menu principale ai 600x720 per visualizzare su schermo il video della cattura in tempo reale. Utilizzare una grandezza della finestra discreta permette la navigazione su piccoli display: in particolare, in presenza di un ampio schermo, si può gestire la cattura su un lato del display e sfruttare l'altro lato per testare il risultato dell'animazione sul software di modellazione.
La disposizione degli elementi della GUI all'interno delle finestre di navigazione seguono un allineamento ben definito in quanto si sfrutta la presenza di una griglia verticale. Inoltre, l'organizzazione dei widget facilita la lettura dell'interfaccia all'utente. Da un punto di vista cromatico, la palette dei colori è composta dal nero, dal bianco e dal blu, comprese le sue tonalità più chiare: alla luce di quanto detto, la scelta dei colori è stata studiata appositamente non solo per garantire una certa armonia dal punto di vista estetico, ma per creare, allo stesso tempo, contrasto tra sfondo, scritte ed icone, necessario per assicurare immediatezza nella distinzione degli elementi all'occhio umano.

LIBRERIE

Per la componente grafica del tool, sono stati utilizzati due package:

- **Tkinter:** dispone di widget per consentire all'utente di interagire con l'interfaccia. La libreria gestisce, all'interno del tool, l'apertura/chiusura di finestre di navigazione, la visualizzazione di icone e pulsanti (Button), la scelta di una colorazione adeguata e l'interazione con gli elementi della GUI.
- **PIL:** è la libreria che funge da supporto a Tkinter, consentendo la visualizzazione di immagini all'interno dei widget. In particolare, sono stati importati i moduli *ImageTk*, che permettono di creare un oggetto PhotoImage, e *Image* dal package *PIL*. Un widget della libreria Tkinter possiede una proprietà 'Image' che necessita di un PhotoImage da visualizzare (nel caso dell'applicazione sono delle icone in formato png). Per quanto riguarda il caricamento delle immagini tramite path, si è utilizzato il package *Image*.

5.2.1 Finestre di interazione

L'avvio del Motion Capture e la registrazione dell'attore avviene in modo rapido, grazie all'organizzazione efficiente delle window. L'utente dovrà interagire seguendo tre soli step, che corrispondono a tre schermate in totale:

1. **Menu principale**

Figura 5.1: Main Menu

La prima schermata dell'applicazione Python visualizza una finestra di selezione per decidere la tipologia di Motion Capture da avviare.
L'utente può scegliere tra tre pulsanti schierati in verticale, caratterizzati da un'icona creata appositamente per indicare il modello di cattura preferito: del viso, del corpo o di una singola mano.
Una volta che l'utente procede con la selezione, si apre la finestra corrispondente al button premuto, come viene elencato di seguito.

2. **Istruzioni e file di salvataggio**

Figura 5.2: Face capture

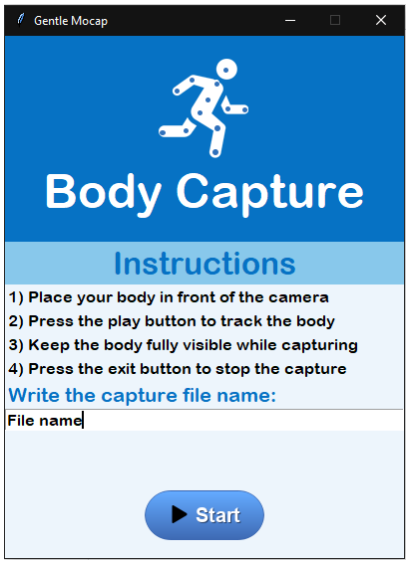


Figura 5.3: Body capture

Figura 5.4: Hand capture

Selezionata la tipologia di Motion Capture, si apre la finestra corrispondente alla sezione del corpo che si vuole registrare. Su schermo viene visualizzata una lista di istruzioni che l'utente deve leggere e rispettare per eseguire correttamente la cattura. E' presente, inoltre, una casella di testo da riempire con una stringa, corrispondente al nome del file dove saranno salvati i dati di movimento dell'attore. Il bottone Start, collocato in basso, permette l'avvio del Motion Capture, con conseguente apertura della finestra di registrazione.

3. **Esecuzione Motion Capture**
   L'ultimo step prevede la registrazione dei movimenti dell'attore, la lettura delle espressioni facciali e il tracciamento della mano. Un riquadro visualizza, frame per frame, in che modo avviene la cattura e i landmark del corpo che il software sta rilevando, facilmente individuabili per il loro colore celeste.

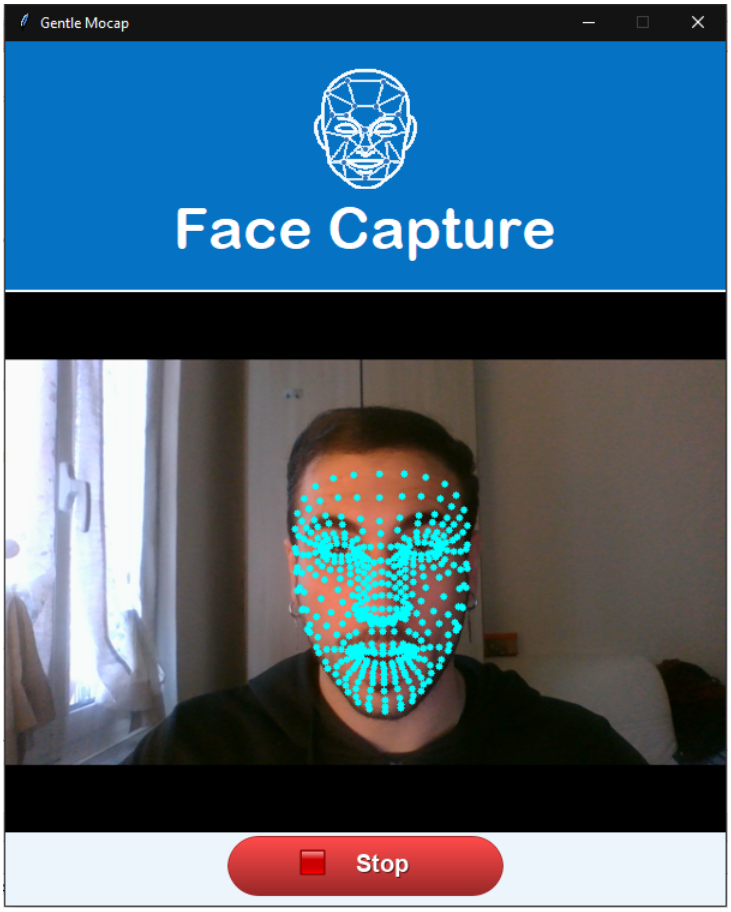


Figura 5.5: Face recording [468 landmarks]

L'alterazione del viso è indicata dai landmark posizionati sull'attore che si estendono a partire dal mento fino alla fronte. In totale, i punti di cattura sono 468, suddivisi equamente tra lato destro e sinistro della faccia.
Nonostante si tratti di una cattura molto precisa, visto l'elevato numero di riferimenti sul viso, nel secondo tool di generazione dell'animazione, verranno elaborati solo un discreto numero di punti, i più importanti. Il motivo a monte di tale scelta è l'ottimizzazione dei tempi di calcolo ed elaborazione dei dati, assicurando, comunque, un'adeguata qualità nella rielaborazione dell'espressione facciale.

Ai fini del prodotto videoludico non sono necessari numerosi landmark in quanto i personaggi, gli insetti 3D, non possiedono così tante espressioni a differenza di un essere umano.

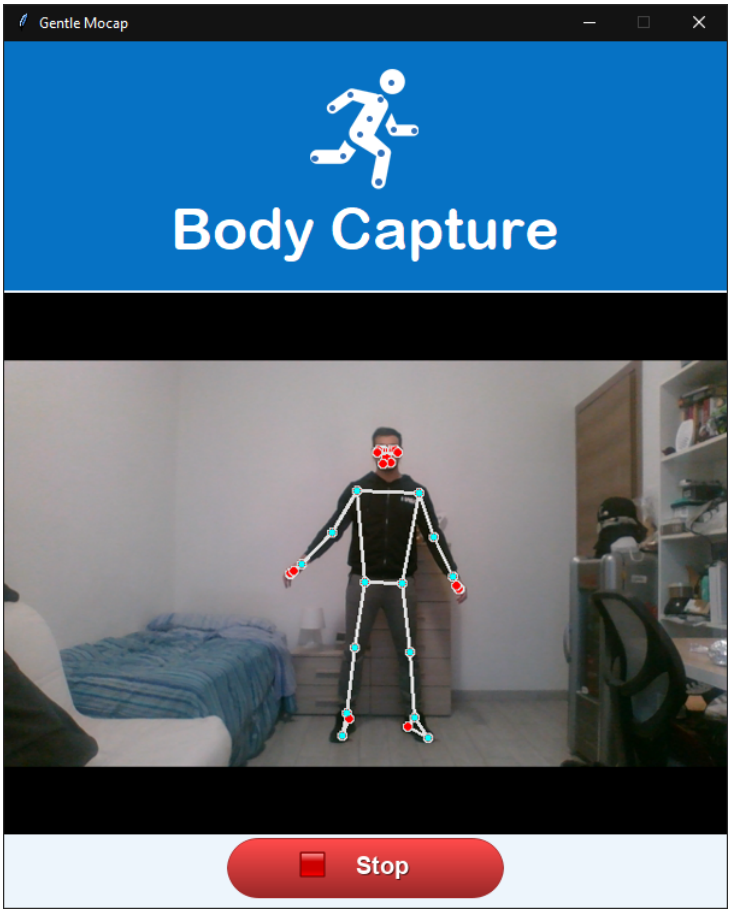


Figura 5.6: Body recording [14 landmarks]

La registrazione dei movimenti del corpo si basa su 14 landmark posizionati in punti chiave come spalle, ginocchia, gomiti, piedi. In figura si può notare come i riferimenti elaborati sono solo quelli dal colore celeste. I dati non utili ai fini del Motion Capture sono segnalati con il colore rosso, come nel caso della figura in alto, poiché la testa o le mani devono essere registrati a parte, selezionando la tipologia di Motion Capture nel menu principale. Il punto rosso del tallone non viene considerato in quanto la stesura del piede sarà stimata e ricostruita dall'add-on, realizzato in Blender, che analizza i riferimenti della caviglia e della punta del piede. Lo stesso ragionamento vale per le mani, la cui posizione sarà stimata prendendo come riferimento il polso.

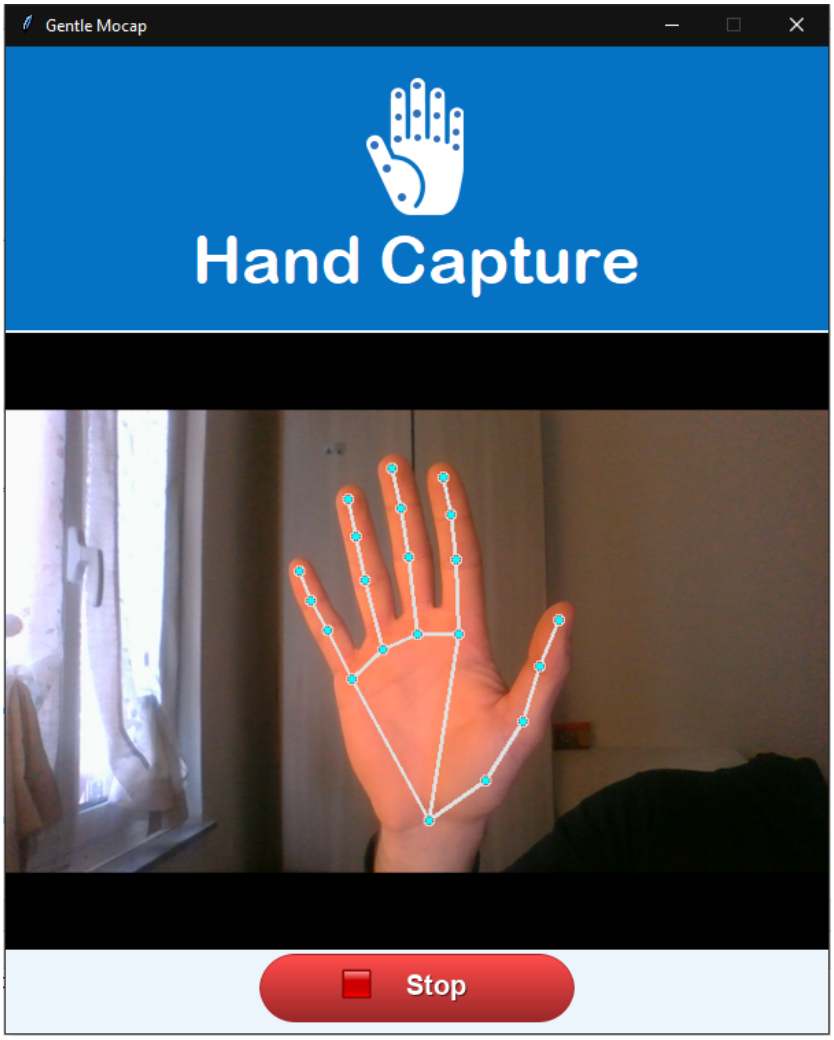


Figura 5.7: Hand recording [21 landmarks]

Il rilevamento e tracciamento della mano prevede l'utilizzo di 21 punti chiave che si posizionano lungo le dita e sul palmo. Nello specifico, ogni dito possiede 4 landmark che ne rilevano i movimenti nello spazio. Entrambi i lati della mano vengono riconosciuti dal tool, per cui la lettura avviene sia dalla parte del palmo e sia dalla parte del dorso. L'esecuzione di Hand Capture è utile al secondo tool in Blender per la successiva elaborazione delle gesture e trasferimento di quest'ultime sotto forma di animazioni sui modelli 3D.

Per ogni finestra di cattura è presente il button Stop, utilizzato per terminare la registrazione. Il salvataggio dei dati di cattura nel file denominato nella scheda precedente sarà automatico. Ogni landmark è memorizzato in coordinate cartesiane x,y,z.

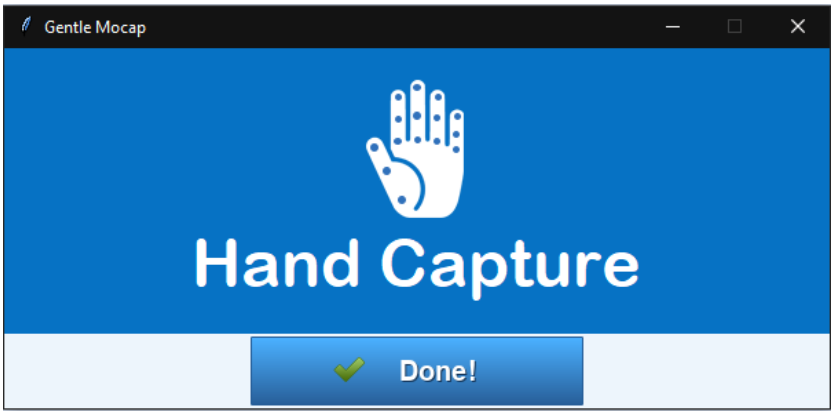


Figura 5.8: Hand capture done

Dopo aver premuto il bottone Stop con conseguente interruzione della registrazione, verrà visualizzato un avviso contenente informazioni relative all'esito positivo o negativo del Motion Capture. Nel caso di successo, si possono estrapolare i dati di Mocap dal file salvato, localizzato all'interno di una cartella specifica del tool denominata "Capture Files".

5.3 Struttura

Il processo di tracking e detection, all'interno del tool Gentle Mocap, viene eseguito tramite la programmazione di tre moduli differenti: Face Capture, Body Capture e Hand Capture. Ogni componente si dedica esclusivamente alla sezione del corpo specifica, rispettivamente al viso, al corpo e alle mani. Si tratta dei processi di cattura che avvengono in background rispetto all'interfaccia utente descritta precedentemente. Gli algoritmi di Motion Capture retrostanti alla libreria MediaPipe utilizzata permettono di riprendere i movimenti con un frame rate medio di 30 fps. Tenendo conto di test di simulazione di Gentle Mocap e del sovraccarico di movimenti nel caso peggiore, la frequenza dei fotogrammi stimata è:

- circa 20-24 fps nel caso l'attore esegua molte azioni in poco tempo. In questo caso, stiamo analizzando la situazione peggiore in cui tutto dipende da quanti landmark il software sta tracciando o rilevando: ad esempio, se le espressioni facciali cambiano rapidamente, l'analisi dei 468 punti di riferimento del viso porta ad un calo drastico degli fps. Se, invece, vengono considerate le articolazioni della mano o del corpo, la frequenza dei fotogrammi è più alta perché il numero dei landmark è inferiore.

- circa 30 fps per il frame rate medio. Tale valore viene ottenuto quando avviene uno spostamento dei punti chiave di cattura che cambiano posizione in modo

normale e non istantaneo. Il tool riesce, difatti, a rilevare i movimenti con maggiore fluidità, mantenendo gli fps ad un valore adeguato di ripresa.

- sopra i 30 fps nel caso migliore. Questo scenario si verifica in due condizioni:

  1. La prima, la più evidente, si concretizza quando la cattura non rileva nessuna parte del corpo. In questo caso, gli algoritmi di Mocap vanno in una sorta di "standby" poiché non spendono cicli macchina per eseguire istruzioni di cattura dei landmark, ma rimangono in attesa dell'imminente comparsa di una mano, del corpo o del viso dell'attore. Il frame rate raggiunge valori molto alti, anche sopra i 50 fps.
  2. La seconda situazione prevede che i punti di riferimento rimangano quasi completamente immobili. Il loro minimo spostamento consente agli algoritmi di non faticare nella stima della posizione rispetto al frame precedente. I fotogrammi al secondo possono raggiungere un range di 40-50 fps.

LIBRERIE

I processi algoritmici di gestione del Motion Capture sono forniti dal package:

- **MediaPipe:** è una libreria cross-platform che fornisce funzioni di machine learning per quanto riguarda l'ambito del computer vision. Il modulo è stato sviluppato da Google e permette di apprendere, rilevare e riconoscere i movimenti umani, gestendo questi dati in maniera efficiente e ottimizzata. Il package offre prestazioni real-time fondamentali per le esperienze live, grazie all'utilizzo di architetture leggere ad-hoc e all'accelerazione GPU in tutta la pipeline. Tra le soluzioni di Machine Learning offerte da MediaPipe, ai fini della tesi, ne sono state considerate solo tre:

  – *MediaPipe Face Mesh:* è una soluzione che sfrutta l'apprendimento automatico per ricostruire la geometria superficiale 3D del viso. E' in grado di stimare 468 punti di riferimento.

  – *MediaPipe Pose:* è un programma che utilizza il machine learning per tracciare e riconoscere la posa del corpo ad alta fedeltà. Il Motion Capture avviene tramite la stima di 33 landmark su tutto il corpo.

  – *MediaPipe Hands:* è un'efficace modalità di apprendimento delle mani e del tracking delle dita. Le articolazioni della mano sono valutate da 21 punti di riferimento.

5.3.1 Sistema di coordinate

Il sistema di coordinate utilizzato dal tool corrisponde ad un sistema in coordinate 3D. Le informazioni legate alle tre dimensioni sono elaborate numericamente sugli assi x, y e z. Ogni landmark viene memorizzato in MediaPipe come un contenitore che assume un nome diverso in base al tipo di cattura:

- **FACE_LANDMARK**, per il viso
- **POSE_LANDMARK**, per il corpo
- **HAND_LANDMARK**, per la mano

Il landmark contiene al suo interno quattro elementi essenziali:

- **x:** indica lo spostamento del landmark in orizzontale. E' un valore normalizzato tra [0.0, 1.0] rispetto alla dimensione in larghezza dell'immagine ripresa dalla telecamera.
- **y:** rappresenta la posizione in verticale del punto di riferimento. Si tratta di un valore normalizzato tra [0.0, 1.0] rispetto all'altezza del frame registrato.
- **z:** denota la profondità del landmark prendendo come riferimento un'origine diversa in base alla tipologia di Motion Capture:
  - Per il viso: il centro geometrico approssimativo della testa.
  - Per il corpo: il punto medio del bacino.
  - Per la mano: il centro geometrico approssimativo della mano.

  Il valore è normalizzato in un range tra [-1.0, 1.0], con una grandezza che rispetta una scala simile a quella di x. Dunque, si fa riferimento alla larghezza dell'immagine. In questo caso i valori di z possono essere negativi o positivi in base alla distanza dal centro del bacino, della mano o della testa. Un valore molto piccolo, vicino a -1, indica che il punto si trova ad una distanza ravvicinata alla telecamera.
- **visibility:** è la visibilità del punto che acquisisce un valore tra [0.0, 1.0] e viene sfruttata per stimare la probabilità che il landmark sia occluso o meno nel frame. Mentre un valore simile allo 0 indica un punto non visibile, un decimale che si avvicina all'1 simboleggia la completa percettibilità di tale punto.

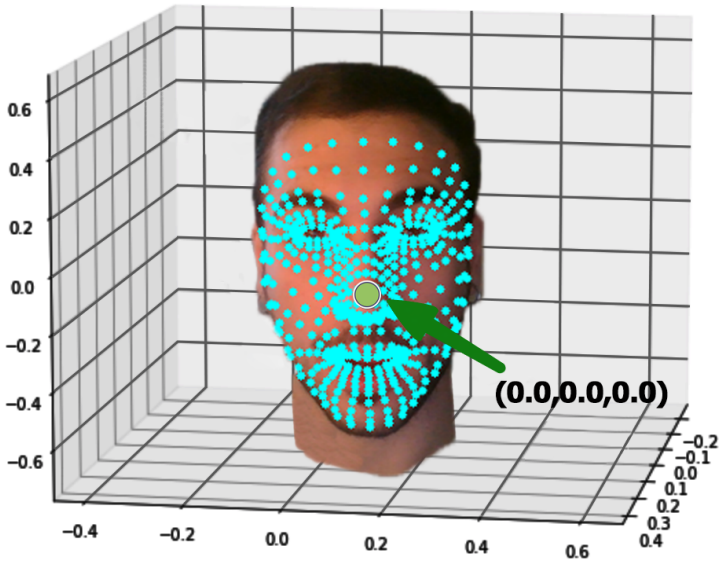


Figura 5.9: Coordinate 3D del viso

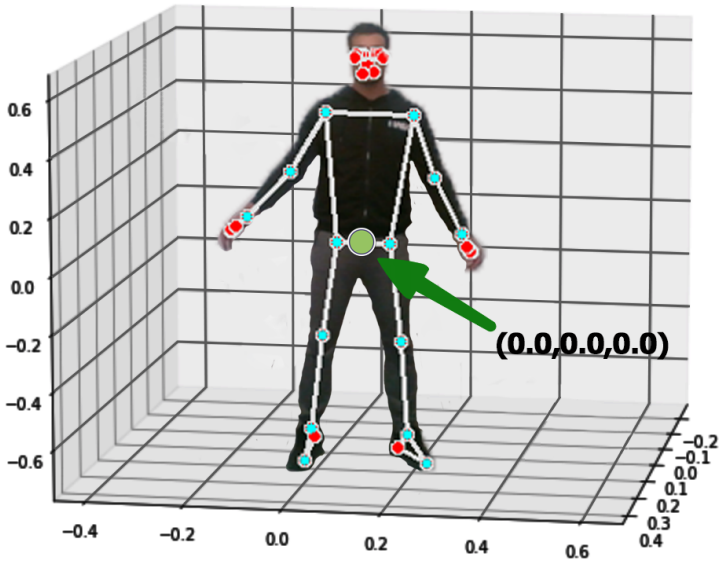


Figura 5.10: Coordinate 3D del corpo

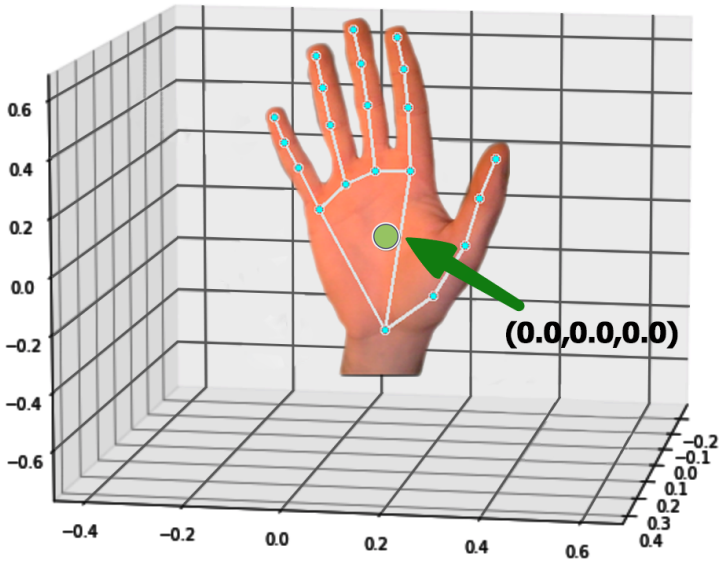


Figura 5.11: Coordinate 3D della mano

5.3.2 Funzionamento

Il funzionamento generale del programma consiste nel salvare la posizione dei landmark in coordinate 3D, ottenendo i valori di riferimento sull'asse x, y, z.
Ogni landmark viene identificato univocamente all'interno del tool da un **ID**. I punti vengono numerati a partire dal numero 0. L'ID permette di ciclare la lista di landmark del viso, corpo e mani restituita durante la cattura, riconoscendo in maniera precisa ogni singolo point.

I valori di cattura dei punti di riferimento corrispondono a numeri decimali normalizzati tra [-1.0, 1.0], i quali assumono un significato valido solo tramite una conversione in pixel dell'immagine. Lo scopo è ottenere, quindi, l'informazione di posizione del punto rispetto al pixel.

```
height, width, channel = frame.shape

x = int(landmark.x * width)
y = int(landmark.y * height)
z = int(landmark.z * width)
```

Figura 5.12: Conversione coordinate

Ogni decimale viene moltiplicato per un valore di riferimento. I valori x e z sono scalati rispetto alla larghezza (width) dell'immagine, mentre il valore y è moltiplicato per la lunghezza del frame (height).

Ecco un esempio di lista di landmark dopo la conversione in pixel:

```
ID Landmark: 0
x: 0.32045164704322815
y: 0.8363696336746216
z: 0.0

ID Landmark: 1
x: 0.39465877413749695
y: 0.8065536022186279
z: -0.027095848694443703

ID Landmark: 2
x: 0.46085914969444275
y: 0.7403775453567505
z: -0.03890252113342285

ID Landmark: 3
x: 0.5114427208900452
y: 0.6868624687194824
z: -0.051591161638498306

ID Landmark: 4
x: 0.5588606595993042
y: 0.6579241156578064
z: -0.06446700543165207
```

Figura 5.13: Valori normalizzati

```
ID Landmark: 0
x: 229
y: 402
z: 0

ID Landmark: 1
x: 280
y: 386
z: -17

ID Landmark: 2
x: 324
y: 352
z: -26

ID Landmark: 3
x: 356
y: 324
z: -36

ID Landmark: 4
x: 386
y: 306
z: -46
```

Figura 5.14: Valori in pixel

Al termine della cattura, vengono memorizzate le coordinate dei punti su un file di testo. Ad ogni frame si inserisce in un'unica stringa la lista di landmark che sono stati rilevati in quell'istante. Da un frame ad un altro, si nota come la stringa diventi più grande, in quanto vengono aggiunte altre liste di landmark. Terminata la registrazione dei movimenti, la mole di dati di cattura presente all'interno della stringa viene riversata in un file.txt scelto dall'utente. Il salvataggio dei dati in una singola stringa ed il trasferimento di questi ultimi sul file al termine del Motion Capture è una manovra essenziale, dovuta al fatto che la scrittura di un file durante l'esecuzione di algoritmi di Mocap richiede calcoli onerosi che potrebbero influire sulla velocità di cattura. Per questo motivo, la scrittura dei dati viene effettuata sul finale.

```
1  355,163,-116 301,161,-100 392,179,-112 262,177,-84 430,193,-178 223,187,-140 342,246,-8 311,245,8
   347,302,18 302,302,38 353,360,98 296,356,129 361,378,24 289,378,62
2  356,163,-113 301,161,-102 392,179,-107 263,177,-90 430,193,-166 224,187,-146 342,246,-5 311,245,5
   347,302,18 302,302,22 353,360,96 297,356,101 361,378,21 289,378,30
3  356,163,-111 301,161,-102 392,179,-106 264,177,-90 430,192,-164 224,187,-147 343,246,-4 311,245,4
   346,302,21 302,302,16 353,360,99 297,356,93 361,378,25 289,378,20
4  357,164,-113 301,162,-107 392,179,-106 264,177,-97 430,193,-166 225,187,-152 343,246,-3 311,245,3
   346,302,15 302,302,18 352,360,87 297,356,98 361,378,12 289,378,27
5  357,164,-116 301,162,-108 392,179,-111 264,177,-96 430,193,-169 225,187,-150 343,246,-3 311,245,3
   346,302,16 302,302,16 352,360,90 297,357,95 362,378,15 289,378,23
6  357,164,-116 302,162,-108 392,179,-110 264,177,-95 430,193,-169 225,187,-149 343,246,-3 311,245,3
   346,302,16 302,302,17 352,360,91 297,357,96 362,378,16 289,378,24
7  357,164,-117 302,162,-107 392,180,-112 264,177,-94 430,193,-171 225,187,-149 343,246,-3 311,245,3
   346,301,18 302,302,12 352,360,95 297,357,87 362,378,19 289,378,11
8  357,164,-117 302,162,-106 392,180,-112 264,177,-93 430,193,-171 225,187,-148 343,246,-2 311,245,2
   346,301,17 302,302,14 352,360,91 297,357,91 362,379,14 289,378,18
9  357,164,-117 302,162,-106 392,180,-112 264,177,-92 430,193,-173 225,187,-147 343,246,-2 311,245,2
   346,302,17 302,302,13 352,360,91 297,357,90 362,379,15 289,378,17
10 357,164,-117 302,162,-106 392,180,-112 265,177,-91 430,193,-173 225,187,-144 343,246,-2 311,245,2
   346,301,15 302,302,12 352,360,88 297,357,88 362,379,11 290,378,14
```

Figura 5.15: Esempio di coordinate nel file di testo [10 frame]

5.3.3 Face Capture

La sezione del tool, inerente alla cattura del viso, fa riferimento alla soluzione di Machine Learning "Face Mesh" fornita da MediaPipe. La superficie facciale dell'attore viene ricostruita con 468 landmark nello spazio 3D. Una densità maggiore dei punti è presente nelle aree più centrali e movimentate del viso come labbra, occhi, naso.

PROCRUSTES ANALYSYS

I dati geometrici facciali sono costituiti da primitive 3D: punti di riferimento, segmenti per la connessione dei punti e triangolazione della mesh. Le attività retrostanti al tracciamento del viso utilizzano un metodo statistico denominato "Procrustes Analysis" che viene eseguito sulla CPU con impatti minimi di memoria/-velocità. Il sistema si occupa dell'analisi di un insieme di forme multidimensionali: una forma o un set di forme vengono confrontate con altre, tentando di trovare uno stato di massima sovrapposizione. Questo stato ottimale viene stabilito attraverso la traslazione, la riflessione, la rotazione e il ridimensionamento delle matrici di coordinate delle forme in questione.

FACE MESH PIPELINE

La pipeline per la ricostruzione della mesh facciale si compone di due sistemi di rete neurale che lavorano in real-time:

- **Face detection model:** un rilevatore che lavora sull'immagine, con il compito di calcolare le posizioni dei volti. Quando il modello di riferimento non è

più in grado di identificare la presenza del viso, viene invocato nuovamente il rilevatore per rilocalizzare il viso.

- **Face landmark model:** consiste in un modello di riferimento 3D del viso che opera sulle posizioni rilevate, stimando la geometria facciale tramite regressione, ovvero un processo statistico che lavora su dataset per effettuare previsioni.

L'elaborazione dei dati è performante in quanto l'immagine viene ritagliata per concentrarsi solo sul volto dell'attore ed il crop del frame fa sì che i dati di sfondo e i movimenti esterni al viso non influiscano sui calcoli. Questo sistema consente agli algoritmi di registrazione di raggiungere la massima accuratezza nella previsione delle coordinate 3D dei landmark. All'interno della pipeline, i ritagli dell'immagine possono essere generati anche in base ai landmark del viso identificati nel frame precedente.

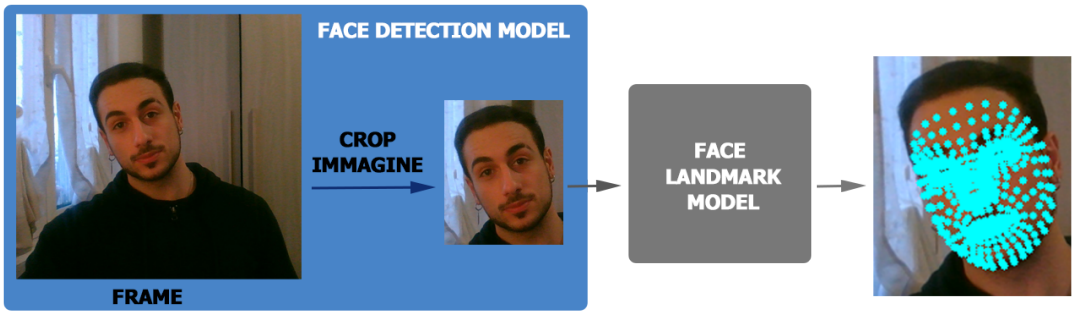


Figura 5.16: Architettura Face Mesh

CONFIGURAZIONE

Prima di iniziare la cattura del viso, è necessario configurare un Media Pipe Object specificando, tramite opzioni, come il processo di Motion Capture si debba comportare in presenza di volti. I parametri e i valori assegnati per l'inizializzazione sono elencati di seguito:

1. *Static_Image_Mode* [False]: se settato a False, opera un abbozzo di detection dei volti nei primi frame e, raggiunta un'alta confidenza, conserva il tracking localizzando i corrispettivi landmark del viso. Il sistema rappresenta la soluzione più vantaggiosa in quanto evita di invocare il rilevamento finché non si perde il tracciamento del viso. Se l'opzione è True, la detection opera su ogni singolo frame risultando onerosa a livello di calcolo.

2. *Max_Num_Faces* [1]: indica il massimo numero di volti da rilevare. Se il valore è 2, consente di catturare dati relativi a due volti.

3. *Refine_Landmarks* [False]: descrive la possibilità o meno di perfezionare i punti di riferimento intorno agli occhi e le labbra, aggiungendo landmark aggiuntivi attorno alle iridi.

4. *Min_Detection_Confidence* [0.5]: è il valore oltre il quale la confidenza del rilevamento è considerata riuscita. Se la confidence per il detect è maggiore di 0.5 si procede al tracking del volto.

5. *Min_Tracking_Confidence* [0.5]: è il valore minimo affinchè la confidenza di un tracciamento sia considerata riuscita. Se il tracking confidence è maggiore di 0.5, viene mantenuto il tracking, altrimenti si ritorna ad operare la detection. Il vantaggio è che non si usa sempre un modello di detection, ma viene conservato il tracking per ottimizzare la velocità di operazione.

LANDMARK DEL VISO

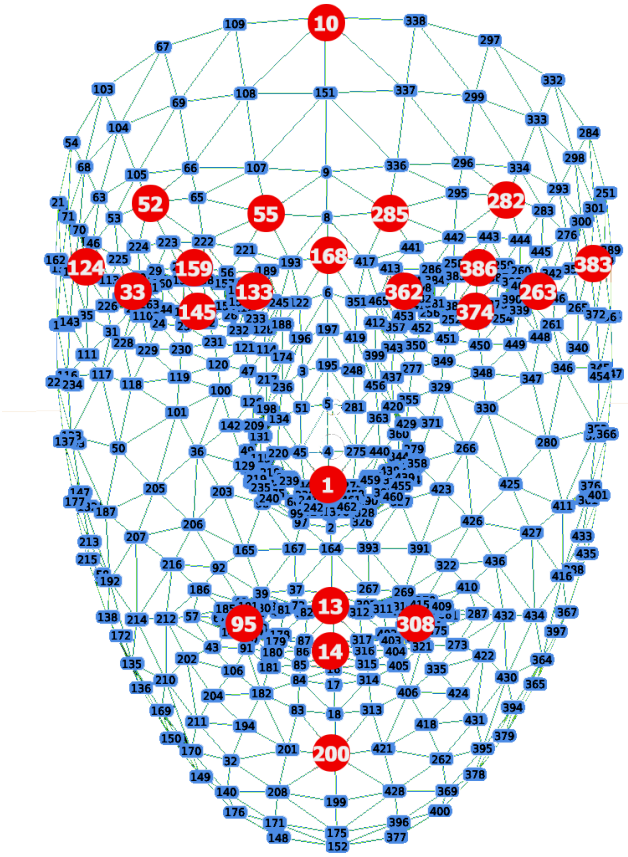


Figura 5.17: 468 Face Landmarks, utilizzati: 22

Nonostante i punti di riferimento analizzati durante il processo di cattura del viso siano 468, nel tool in Blender i landmark elaborati saranno molti di meno: al fine di animare i personaggi 3D e al termine della cattura, le coordinate estrapolate dal file di testo saranno solo 22, aventi id [1, 10, 124, 13, 133, 14, 145, 159, 168, 200, 263, 282, 285, 308, 33, 362, 374, 383, 386, 52, 55, 95]. La decisione di salvare nel file tutte le 468 coordinate, nonostante ne vengano utilizzate solo 22, pone comunque le basi per una prospettiva futura di miglioramento e aumento della precisione dell'add-on facciale realizzato in Blender.

5.3.4 Body Capture

Il Motion Capture del corpo si basa sulla soluzione di Machine Learning "Pose", messa a disposizione dal package MediaPipe. Il riconoscimento delle pose di un essere umano viene condotto tramite "BlazePose", un sistema di stima di 33 landmark collocati in punti chiave del corpo.

TOPOLOGIE A CONFRONTO: COCO vs BLAZE POSE

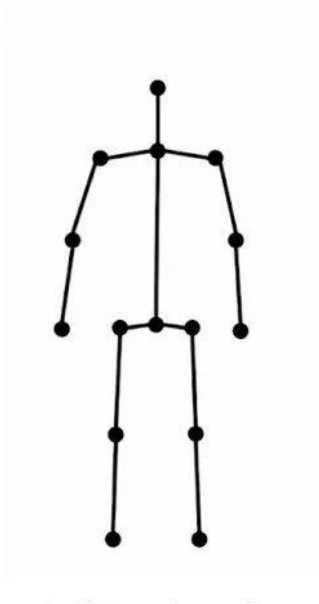


Figura 5.18: Topologia COCO

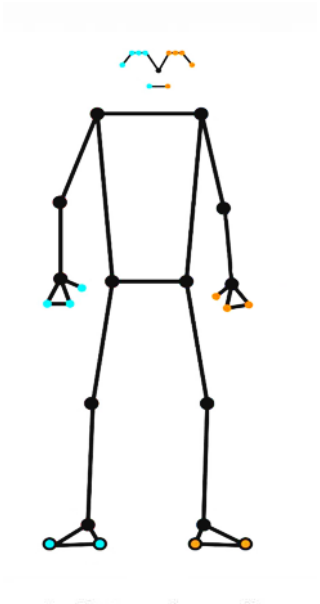


Figura 5.19: Topologia Blaze

Blaze Pose è una topologia moderna che utilizza 33 punti chiave del corpo umano, rispetto alla topologia standard attuale COCO che per la posa stima 17 punti di riferimento. COCO localizza landmark chiave solo fino alle caviglie, ai polsi e alla testa, senza fornire informazioni di posizionamento relative a mani, piedi o di punti facciali. Con l'avvento di BlazePose, sono stati aggiunti 16 riferimenti in

più rispetto alla topologia standard ed è stato alterato lo scheletro base del corpo. Questo sistema consente applicazioni pratiche efficaci per pose di fitness e danza. I nuovi punti chiave inclusi nella nuova topologia identificano il viso, le mani e i piedi e sono per la precisione:

- 11 punti per il viso: 4 per l'identificazione di un singolo occhio, 1 per il naso e 2 per la bocca.
- 6 punti per le mani: 3 per ogni singola mano, nella zona del palmo.
- 4 punti per i piedi: 2 punti per ogni piede, 1 per il tallone, 1 per la punta.

BODY POSE PIPELINE

La stima della posa avviene tramite una pipeline di Machine Learning che lavora su due fasi:

1. **Detection (o rilevamento):** questo step utilizza un rilevatore, con il quale la pipeline si occupa di individuare dapprima la persona/la regione di interesse (ROI) all'interno del frame di ripresa.
2. **Tracking (o tracciamento):** dopo la fase di detection, il tracker prevede tutti e 33 i landmark di posa del corpo umano dalla ROI.

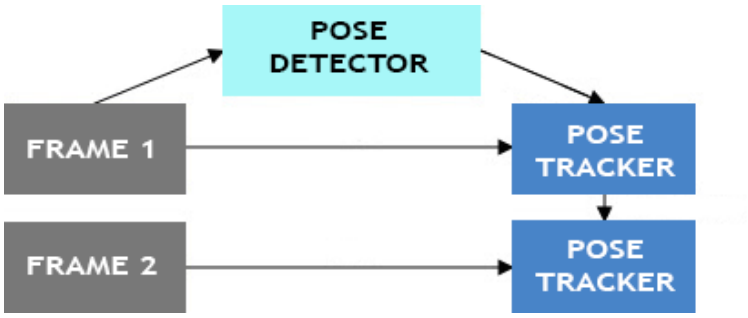


Figura 5.20: Pipeline Pose

Sono disponibili due modalità di stima che variano per precisione e performance di calcolo:

1. **Detection Mode:** il primo metodo consiste nell'effettuare ad ogni frame la detection, avendo come benefit la precisione dei dati di cattura, ma come grosso svantaggio la riduzione delle performance di calcolo.
2. **Detection/Tracking Mode:** il secondo metodo, quello consigliato, è di evitare quanto più possibile la fase di detection poichè onerosa, se non quando

si perde di vista il corpo umano durante la ripresa o la confidenza raggiunge valori bassi. Per ottimizzare il sistema, la Pose Pipeline propone una soluzione che permette di effettuare il tracking analizzando i fotogrammi precedenti, derivando e stimando la ROI dei punti chiave di posa.

CONFIGURAZIONE

Il processo di stima della posa umana richiede la costruzione iniziale di un Media Pipe Object, impostando il comportamento da seguire per la cattura dei movimenti. I parametri e i valori assegnati per l'inizializzazione sono elencati di seguito:

1. *Static_Image_Mode* [False]: se settato a False, il sistema di cattura utilizza il secondo metodo "Detection/Tracking Mode" descritto precedentemente. Dapprima si esegue la detection della persona analizzando i primi frame ed, una volta che la confidenza raggiunta è adeguata, conserva il tracking del corpo umano localizzando i corrispettivi landmark di posa.
   Se l'opzione è True, si utilizza il primo metodo "Detection Mode", che risulta però oneroso a livello di calcolo. La detection lavora su ogni singolo frame stimando la posa del soggetto in movimento.

2. *Model_Complexity* [1]: parametro che indica la complessità del modello che rileva le pose del corpo umano. E' rappresentato da un valore compreso tra [0,1,2], all'interno del quale un'alta complessità implica una maggiore latenza.

3. *Smooth_Landmarks* [True]: permette la possibilità di smussare, tramite filtri, i landmark del corpo, per evitare il jitter dei punti di riferimento.

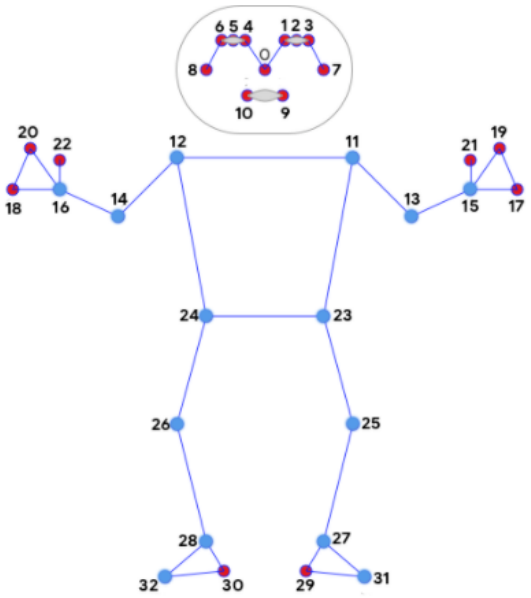
4. *Enable_Segmentation* [False]: se impostato a True, aggiunge una maschera di segmentazione che permette di suddividere i frame in segmenti da analizzare separatamente.

5. *Smooth_Segmentation* [True]: la soluzione filtra o meno le maschere di segmentazione su diverse immagini di input, con lo scopo di ridurre il jitter.

6. *Min_Detection_Confidence* [0.5]: è il valore oltre il quale la confidenza della detection risulta riuscita. Un valore di confidence detection maggiore di 0.5 indica il passaggio al tracking della posa umana.

7. *Min_Tracking_Confidence* [0.5]: viene rappresentato dal valore sotto al quale è necessario rieseguire la detection. Tuttavia, se il valore supera lo 0.5, allora la confidenza è abbastanza alta da poter mantenere il tracking dei landmark del corpo umano.

LANDMARK DEL CORPO



0. nose
1. left_eye_inner
2. left_eye
3. left_eye_outer
4. right_eye_inner
5. right_eye
6. right_eye_outer
7. left_ear
8. right_ear
9. mouth_left
10. mouth_right
11. left_shoulder
12. right_shoulder
13. left_elbow
14. right_elbow
15. left_wrist
16. right_wrist
17. left_pinky
18. right_pinky
19. left_index
20. right_index
21. left_thumb
22. right_thumb
23. left_hip
24. right_hip
25. left_knee
26. right_knee
27. left_ankle
28. right_ankle
29. left_heel
30. right_heel
31. left_foot_index
32. right_foot_index

Figura 5.21: 33 Body Landmarks, utilizzati: 14

Il corpo umano viene ricostruito mediante 33 punti chiave secondo la topologia BlazePose. I landmark e le relative coordinate che verranno memorizzate nel file di testo risultano essere 14, ma vengono scartate le coordinate relative alle mani, che saranno stimate nel software di modellazione Blender a partire dai landmark del polso. Inoltre, non sono ritenuti necessari i punti del viso, poichè saranno catturati in un secondo momento tramite la sezione dedicata Face Capture. Il tallone non viene preso in considerazione poiché nel tool di rigging del personaggio 3D è prevista la generazione della struttura del piede prendendo come riferimento la caviglia e la punta del piede. Nella figura in alto sono indicati in azzurro i punti di riferimento utili all'animazione del modello 3D.

5.3.5 Hand Capture

La mano viene catturata allo scopo di riconoscere i gesti nel secondo tool sviluppato in Blender. L'Hand Capture è reso possibile dalla soluzione di Machine Learning "Hands" presente in MediaPipe. Gli algoritmi di computer vision utilizzati permettono di riconoscere le articolazioni gestendo le occlusioni tra mani/dita. I landmark di riferimento per una singola mano sono 21 e permettono il tracciamento delle dita ad alta precisione.

HAND PIPELINE

Il riconoscimento delle articolazioni della mano prevede una pipeline che si compone di due modelli che lavorano insieme:

- **Palm Detection Model:** è un modello di rilevamento del palmo della mano. Il sistema cerca di apprendere la posizione del palmo all'interno del frame restituendo un bounding box, ovvero un riquadro che delimita la mano. Il crop dell'immagine, come avviene nel "Face Detection Model", ha il vantaggio di concentrare il lavoro degli algoritmi di Machine Learning solo sulla figura della mano ritagliata.
- **Hand Landmark Model:** è il modello che opera sul bounding box restituito dal "Palm Detection Model". Avere un'immagine della mano ritagliata riduce drasticamente il calcolo su dati di contorno che influirebbero sulla velocità di esecuzione dei processi di riconoscimento delle articolazioni. Lavorando sul crop dell'immagine, il modello dedica la previsione dei landmark su una zona specifica. Il sistema di Hand Detection si occupa di stimare i punti chiave della mano restituendo le relative coordinate 3D.

La Hand Pipeline permette, inoltre, di identificare un nuovo riquadro di delimitazione della mano osservando e analizzando i landmark catturati nei frame precedenti. Nell'evenienza in cui non venga più rilevata la presenza della mano, viene invocato il rilevamento della mano tramite il "Palm Detection Model" che va a rilocalizzare la mano.

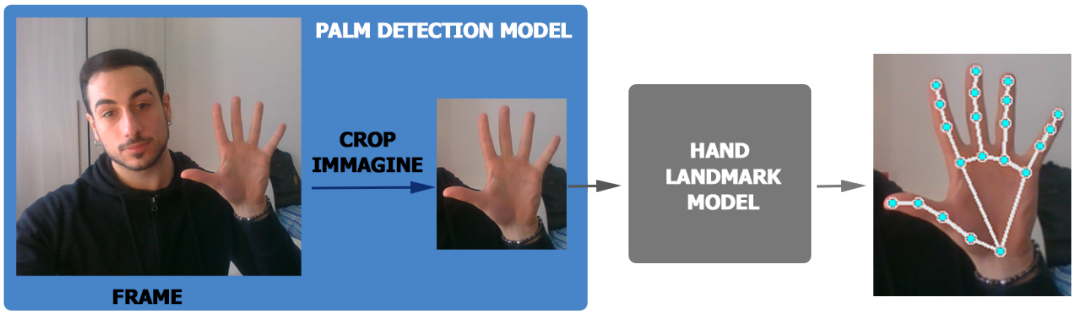


Figura 5.22: Architettura Hand

CONFIGURAZIONE

Per rilevare la posizione della mano e riconoscere i punti chiave delle articolazioni, è necessario configurare un Media Pipe Object, elencando al suo interno le indicazioni da rispettare per la cattura della mano. I parametri e i valori assegnati per l'inizializzazione sono elencati di seguito:

1. *Static_Image_Mode* [False]: se settato a False, il sistema di cattura esegue il modello "Hand Landmark Model" dopo il primo passaggio di rilevamento del palmo. Una volta riconosciuto il palmo con una confidenza alta, si passa al tracciamento dei landmark sulla mano.
   Se l'opzione è True, si utilizza il modello "Palm Detection Model" ad ogni frame. Nonostante la sua precisione, bisogna considerare lo svantaggio legato al dispendio di risorse computazionali.

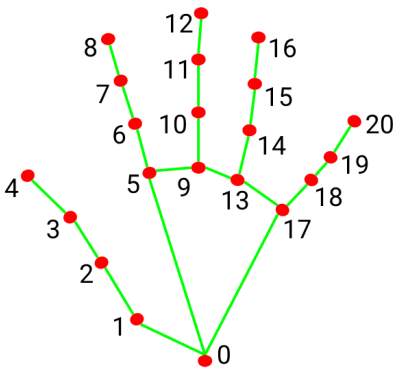
2. *Max_Num_Hands* [1]: indica il massimo numero di mani da rilevare. Se il valore è 2, il sistema consente di catturare due mani contemporaneamente. Ai fini dell'animazione dei personaggi 3D, è stato necessario catturare una singola mano.

3. *Model_Complexity* [1]: indica la complessità del modello "Hand Landmark Model" e corrisponde ad un valore compreso tra [0,1,2], dove l'incremento di accuratezza implica una maggiore latenza.

4. *Min_Detection_Confidence* [0.5]: è il valore oltre il quale la confidenza è ritenuta valida per passare alla fase di tracking della mano.

5. *Min_Tracking_Confidence* [0.5]: rappresenta il valore in corrispondenza del quale si può passare o meno dal tracking al detection e viceversa. Se la confidenza è maggiore di 0.5 si mantiene il tracciamento delle articolazioni. Se invece la confidenza scende sotto i 0.5 è necessario invocare il rilevamento della mano.

LANDMARK DELLA MANO



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Figura 5.23: 21 Hand Landmarks, utilizzati: 21

Le articolazioni della mano vengono stimate mediante 21 punti chiave. L'insieme dei 21 landmark sono salvati all'interno del file di testo che sarà poi analizzato

nell'add-on realizzato in Blender. In particolare, il secondo tool si occuperà dell'animazione di parti esterne degli insetti 3D all'interno del videogioco, come ali, antenne, mandibole e zampe. Per il controllo delle appendici degli insetti verranno analizzate le gesture della mano. L'argomento relativo al riconoscimento dei segni verrà affrontato nel capitolo successivo, che tratta il modo in cui l'add-on riconosca il movimento delle articolazioni per associarle a una gesture specifica.

Capitolo 6

Rigging e animazione personaggi

6.1 Addon in Blender

Il secondo tool di Gentle Mocap è l'add-on sviluppato in Blender e si occupa di elaborare i dati prodotti dal Motion Capture. Il primo applicativo restituisce, sottoforma di file di testo, le coordinate 3D dei landmark catturati ad ogni frame. L'add-on, invece, svolge la funzione di analizzare i punti presenti all'interno dei file, per convertirli in animazione 3D sui personaggi del videogioco. Il processo di conversione da dati di Mocap ad animazione segue un filo logico di applicazioni ottimali di tecniche di rigging, correzione del movimento dello scheletro, metodi di smooth del posizionamento dei landmark nello spazio. Il linguaggio di programmazione dell'add-on è Python che segue una stesura del codice derivante dalla libreria di Blender Bpy. Inoltre, l'intero programma è contenuto in un unico file Python. E' stato necessario rendere il tool compatibile con le versioni precedenti e aggiornate del software di modellazione. Pertanto, sono stati creati due file:

- **GentleMocap_Blender___2_79.py:** supporta la versione di Blender 2.79 rilasciata nel settembre 2017. Funziona su entrambe le versioni 2.79a e 2.79b.
- **GentleMocap_Blender_2_8_to_2_92.py:** supporta le versioni di Blender più recenti, dalla 2.80 rilasciata nel luglio 2019 alla 2.92 rilasciata nel febbraio 2021. Nello specifico, ingloba le versioni: 2.80, 2.81, 2.82, 2.83 LTS, 2.90, 2.91, 2.92.

Nella tesi verrà considerata la versione 2.79 come riferimento, tenendo conto che lo stesso approccio è utilizzato nelle versioni successive. La schematica di funzionamento è identica, ma è stato necessario adattare le versioni recenti di Blender a livello di codice poiché la libreria Bpy è stata aggiornata con il tempo.

LIBRERIE

La libreria utilizzata per la gestione dell'intero tool è:

- **Bpy:** è la libreria principale utilizzata da Blender per realizzare l'interfaccia utente e gestire tutte le componenti/moduli presenti nel software di modellazione. Questa libreria svolge un ruolo fondamentale per la lettura dei dati di Mocap e il trasferimento sull'armatura del personaggio, con conseguente animazione dello stesso.

INSTALLAZIONE

Per utilizzare l'add-on è necessario seguire la procedura di installazione che verrà rappresentata in quattro semplici step:

- *Impostazioni*
  Cliccare su file, poi su User Preferences. Si aprirà una nuova finestra con le preferenze utente.

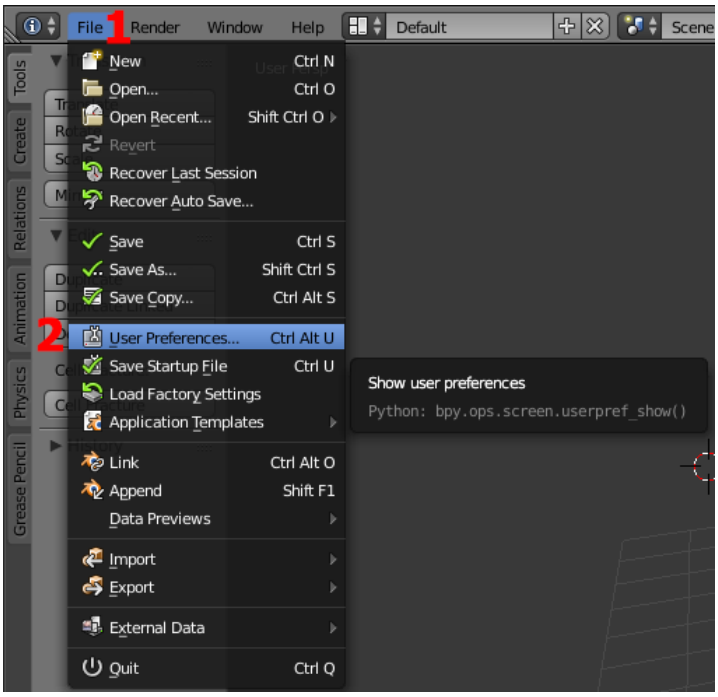


Figura 6.1: User Preferences

- *Ricerca*
  Considerare la sezione Add-ons. Cliccare "Install Add-on from File" per aprire la finestra di ricerca.

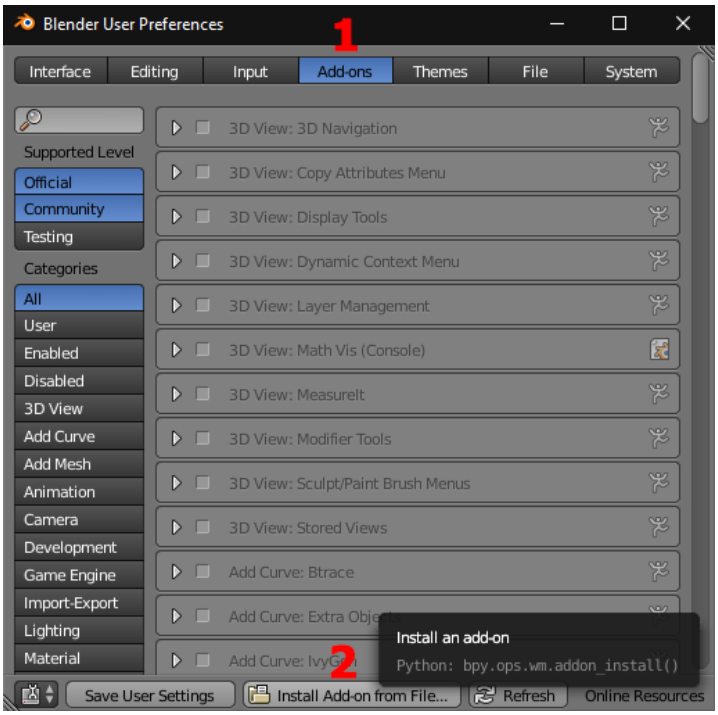


Figura 6.2: Search add-on

- *Installazione*
  Cercare il percorso del file e selezionare "Install Add-on from File" per installare l'add-on.

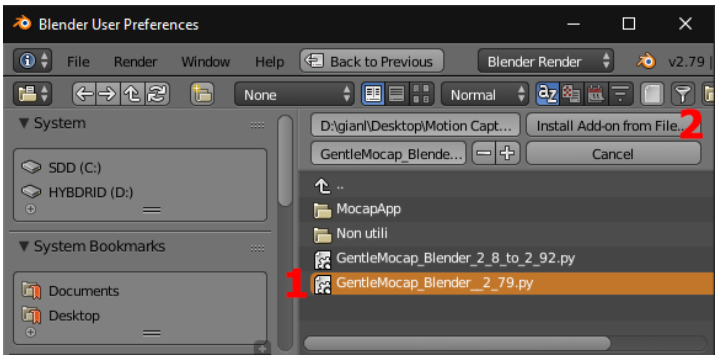


Figura 6.3: Install Add-on from File

- *Abilitazione*
  Per abilitare l'add-on, è necessario cercarlo nella barra di ricerca per poi spuntare l'opzione "Tools: Gentle Mocap Addon".

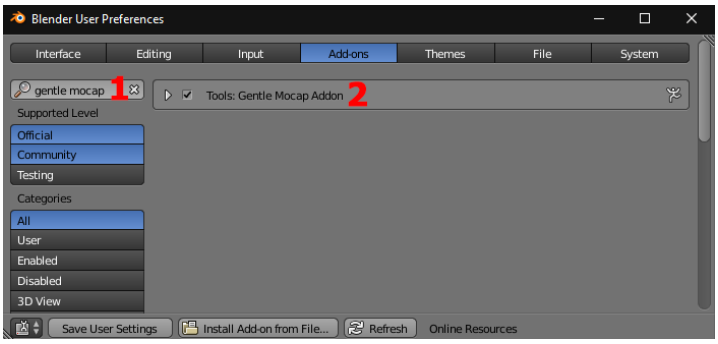


Figura 6.4: Enable Add-on

Terminata l'installazione, il tool verrà visualizzato automaticamente nella schermata principale di Blender.

6.2 Interfaccia utente

La GUI dell'add-on è stata realizzata tramite elementi interattivi forniti dalla libreria Bpy. L'interfaccia risulta basilare, di facile utilizzo e ridimensionabile per avere la completa visione di ciò che sta accadendo mentre si utilizza il tool. Per rispettare la colorazione standard di Blender, sono state utilizzate poche tonalità di colore, infatti prevalgono il nero, il bianco e il grigio. La cromaticità varia per la prima e seconda versione dell'add-on, in base alla tinta che il software di modellazione utilizza per una certa versione. Visto che il modulo Bpy non mette a disposizione molti elementi per personalizzare la GUI, l'interfaccia ha richiesto uno studio ad-hoc per non risultare piatta e con pochi dettagli. Per rendere la GUI ricca di particolari, sono state inserite delle piccole icone, che, posizionate accanto ad un testo, rappresentano un simbolo corrispondente a ciò che descrive il testo. Questo permette all'utente di orientarsi facilmente, associando un mezzo visivo all'azione che vorrebbe compiere sulla GUI. La struttura dell'interfaccia è organizzata per mezzo di una griglia verticale. Per facilitare la ricerca di un task ben preciso, l'interfaccia è suddivisa in sezioni, ciascuna delle quali è identificata con un titolo di descrizione e un'icona. La GUI è suddivisa in diverse finestre di interazione elencate di seguito.

6.2.1 Finestre di Interazione

1. **Menu principale**

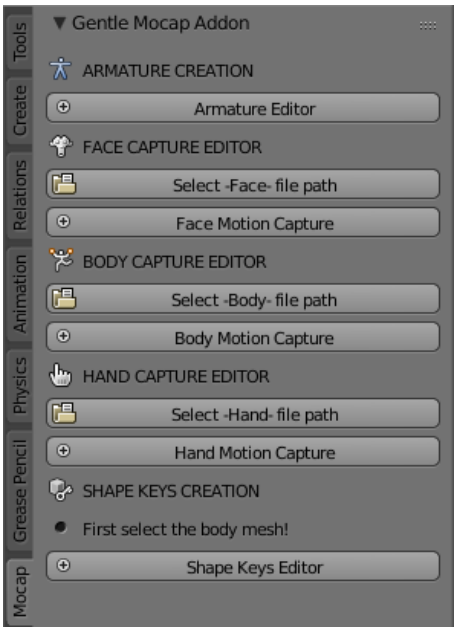


Figura 6.5: Menu for Blender version 2.79

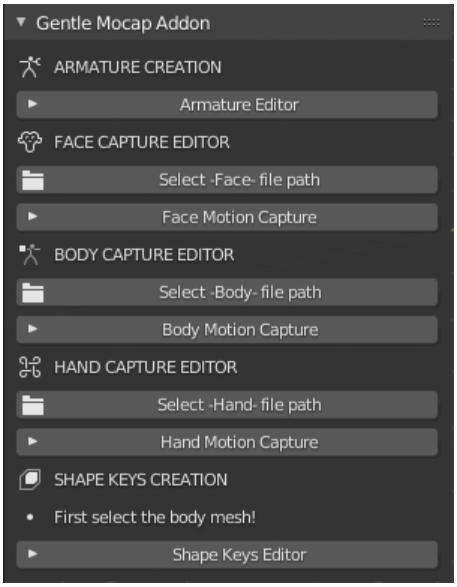


Figura 6.6: Menu for Blender version from 2.80 to 2.92

In figura vi sono le due versioni dell'interfaccia del menu principale. La colorazione varia in base alla tinta standard utilizzata in una specifica versione di Blender. Le icone risultano colorate nella versione 2.79, mentre nelle altre versioni sono completamente bianche. Il menu principale è ripartito in 5 sezioni schierate verticalmente:

- **Armature Creation:** è lo step iniziale di creazione dell'armatura che accoglierà il personaggio 3D.
- **Face Capture Editor:** permette di caricare il file di cattura del viso per convertire i dati in espressioni facciali.
- **Body Capture Editor:** consente di fare l'upload del file relativo ai movimenti del corpo e ha il compito di analizzare le coordinate 3D al suo interno per muovere adeguatamente lo scheletro del personaggio.
- **Hand Capture Editor:** richiede non solo il caricamento del file contenente i landmark acquisiti durante la registrazione della mano, ma analizza anche questi dati per associare un gesto ad un comportamento specifico sul personaggio.

- **Shape Keys Creation:** gestisce la creazione delle shape key per deformare ulteriormente la mesh del modello 3D.

2. **Armature Editor**

Figura 6.7: Armature Editor

L'editor dell'armatura consente di creare lo scheletro iniziale e impostarne la dimensione tramite il parametro:

(a) *Ridimensionamento armatura*

3. **Face Capture Editor**

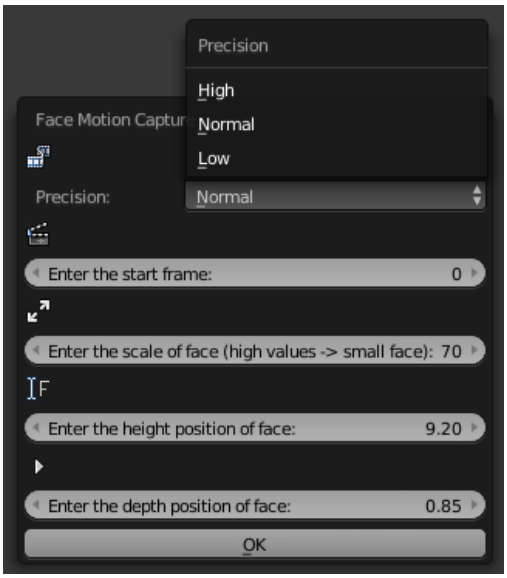


Figura 6.8: Face Capture Editor

L'editor di gestione del viso permette di aggiungere ossa facciali allo scheletro per visualizzarne il movimento. In background i dati del volto vengono salvati in variabili che saranno utilizzate dallo "Shape Keys Creation" per alterare la mesh del viso. I parameteri che si possono impostare sono:

(a) *Precisione*

(b) *Frame di partenza*

(c) *Ridimensionamento viso*

(d) *Posizione di altezza del viso*

(e) *Posizione di profondità del viso*

4. **Body Capture Editor**

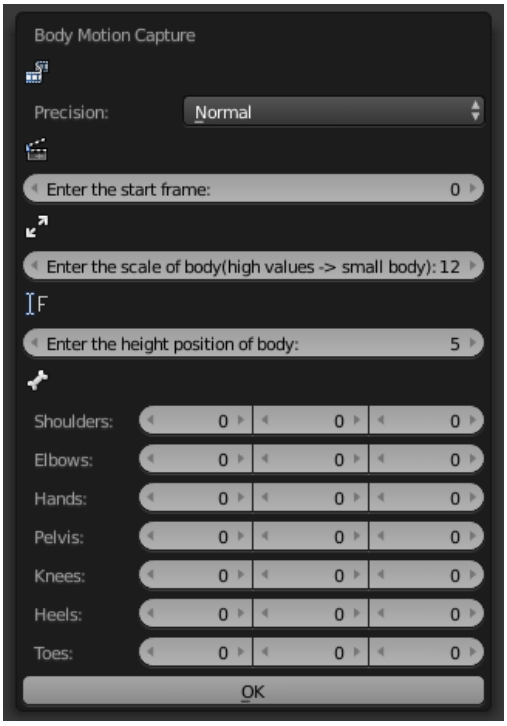


Figura 6.9: Body Capture Editor

L'editor relativo al movimento del corpo consente di adattare lo scheletro del personaggio alla propria conformazione fisica. In seguito, trasforma i dati di Motion Capture in animazione dell'ossatura di braccia, gambe e busto. I parametri modificabili sono:

(a) *Precisione*

(b) *Frame di partenza*

(c) *Ridimensionamento corpo*

(d) *Posizione di altezza del corpo*

(e) *Settaggi di adattamento dello scheletro*

5. **Hand Capture Editor**

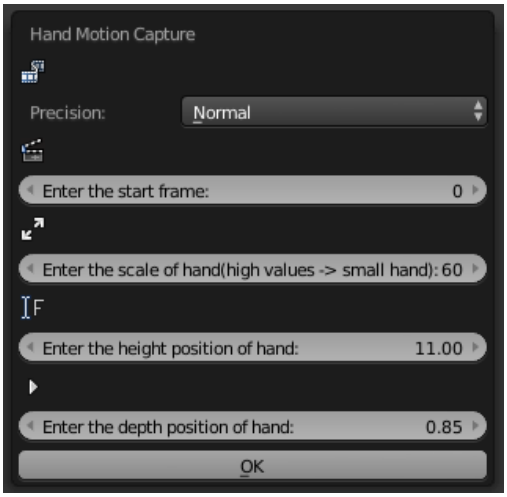


Figura 6.10: Hand Capture Editor

L'editor di analisi delle gesture delle mani si occupa del riconoscimento di gesti che verranno convertiti in animazione limitata a parti specifiche del personaggio. I parametri di settaggio sono:

(a) *Precisione*

(b) *Frame di partenza*

(c) *Ridimensionamento mano*

(d) *Posizione di altezza della mano*

(e) *Posizione di profondità della mano*

6. **Shape Keys Creation**

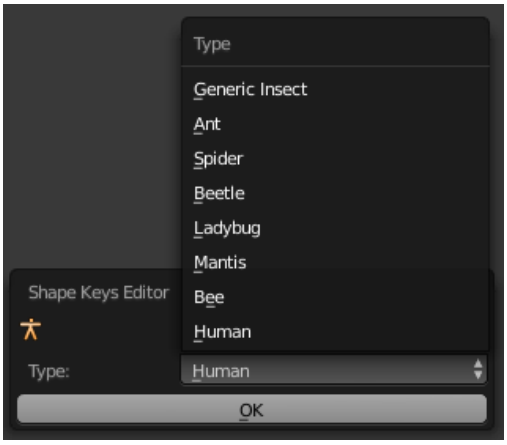


Figura 6.11: Shape Keys Editor

L'editor di gestione delle shape keys garantisce l'alterazione della mesh del viso e altre zone del corpo del personaggio. I valori delle Shape Key variano frame per frame in base ai dati di Motion Capture e modificano la mesh. Il parametro da settare è:

(a) *Tipo di mesh*

6.2.2 Legenda

(a) *Ridimensionamento armatura:* è il valore di scala dell'armatura che può essere settato in un range che va da [0,10].

(b) *Precisione:* indica la precisione con cui vengono elaborati i dati, ovvero ogni quanti frame è necessario leggere le coordinate per poi convertirle in animazione. Ci sono tre impostazioni da scegliere, che indicano a che frame di distanza considerare i landmark registrati dalla sessione di Motion Capture:

- *Low:* ogni 6 frame (Bassa precisione).
- *Medium:* ogni 3 frame (Media precisione).
- *High:* ogni frame (Alta precisione).

(c) *Frame di partenza:* rappresenta il primo frame di interesse da cui far partire la lettura dei dati di Mocap.

(d) *Ridimensionamento viso:* : è un valore di scala dello scheletro del viso.

(e) *Ridimensionamento corpo:* è un valore di scala dello scheletro del corpo.

(f) *Ridimensionamento mano:* è un valore di scala della rappresentazione della mano di riferimento.

(g) *Posizione di altezza del viso:* è la posizione dello scheletro della testa sull'asse verticale.

(h) *Posizione di altezza del corpo:* è la posizione dello scheletro del corpo rispetto all'asse verticale.

(i) *Posizione di altezza della mano:* è la posizione della mano di riferimento.

(j) *Posizione di profondità del viso:* è la profondità a cui viene posizionato lo scheletro della testa.

(k) *Posizione di profondità della mano:* è la profondità a cui viene posizionata la mano di riferimento .

(l) *Settaggi di adattamento dello scheletro:* sono i valori di posizionamento nello spazio 3D delle spalle, gomiti, mani, bacino, ginocchia, talloni e punta del piede. Permette la personalizzazione completa dello scheletro per adattarlo alla struttura fisica del personaggio.

(m) *Tipo di mesh:* indica il tipo di modello sul quale si vuole lavorare. In base all'impostazione scelta, vengono create Shape Key specifiche per la tipologia del personaggio. Le selezioni disponibili sono:

- *Umano*
- *Insetto Generico*
- *Formica*
- *Ragno*
- *Scarabeo*
- *Coccinella*
- *Mantide*
- *Ape*

6.3 Struttura

Il codice sorgente dell'add-on di Gentle Mocap è contenuto in un unico file Python. La struttura del programma si suddivide in vari moduli, ciascuno dei quali esegue una funzione specifica. All'interno della libreria Bpy questi moduli prendono il nome di:

- **Operator:** per l'esecuzione di codice relativo all'analisi dei dati di cattura e trasferimento sul modello 3D sotto forma di animazione.
- **Panel:** per la visualizzazione di un pannello per l'interfaccia utente.

OPERATOR

Un operator è una classe derivante da "bpy.types.Operator". Ha delle proprietà, una funzione di esecuzione (*execute*) e, facoltativamente, una funzione di chiamata (*invoke*). Un operator corrisponde ad un button e, per tale motivo, può essere registrato per farlo apparire nella GUI. Quando si interagisce col button, viene

richiamata la sua funzione di execute. Nello specifico, le funzioni presenti all'interno di un operator sono tre e sono invocate nel seguente ordine:

1. **Invoke:** è la funzione di chiamata che provoca l'apertura di una nuova finestra. Quando si interagisce nel menu principale e si preme un button, viene invocato l'operator corrispondente che si apre come popup.

2. **Draw:** è la funzione che mette su display le proprietà dell'operator. Queste proprietà possono essere modificate dall'utente tramite interfaccia e automaticamente vengono salvate in variabili all'interno dello script dell'operator. Le proprietà in questione sono quelle presenti nella *Legenda*.

3. **Execute:** rappresenta il corpo di esecuzione dell'operator ed ha la funzione più importante, ovvero svolge il calcolo sui dati di Mocap, sul rigging e sull'animazione. Contiene lo script che permette di trasformare delle semplici coordinate 3D in animazione del personaggio, applicando tecniche di ottimizzazione per rendere fluido il movimento del modello.

PANEL

Un panel è una classe derivante da "bpy.types.Panel". Possiede delle proprietà e una funzione di **draw** che permette di personalizzare la UI del pannello. Il panel è utilizzato per la visualizzazione del menu principale e, tramite button, permette di invocare le finestre di interazione, gestite dagli operator per l'elaborazione dei dati di Motion Capture. Un pannello può essere scalato per mostrarne il contenuto interno o compresso per nasconderne gli elementi.

6.3.1 Sistema di coordinate

Il sistema di coordinate 3D assume un concetto diverso all'interno del primo tool di Gentle Mocap rispetto a questo secondo tool di rigging e animazione dei personaggi. La differenza riguarda gli assi y e z, che vengono considerati in maniera differente in entrambi i tool. Il sistema di coordinate utilizzato nel tool di Motion Capture è diverso da quello nell'ambiente del software di modellazione.

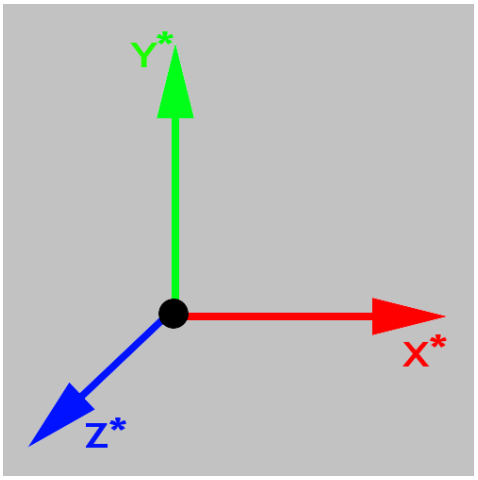


Figura 6.12: Coordinate del primo tool

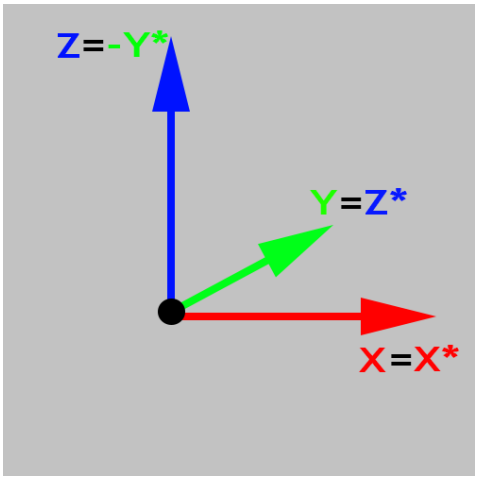


Figura 6.13: Coordinate del secondo tool

Tenendo conto del fattore di conversione dell'asse verticale e dell'asse di profondità, i landmark verranno analizzati all'interno dell'add-on rispettando queste specifiche. Durante la lettura di un file di cattura si passa da (x*,y*,z*) nello spazio 3D del primo tool alla conversione di (x=x*,z=-y*,y=z*) nella scena 3D di Blender.

6.3.2 Armature Editor

E' il primo operator e il primo step da eseguire prima di ogni possibile computazione legata al Motion Capture e si occupa della generazione dell'armatura, ovvero dello scheletro osseo che compone il personaggio. La base iniziale di scheletro preesistente in Blender verrà sottoposta a modifica per comporre una struttura ossea adatta non solo al rigging dei personaggi, ma anche conforme ad ospitare i dati di Motion Capture. La base di partenza è il **Basic Human Meta-rig**, messo a disposizione dall'add-on *Rigify* e già disponibile in Blender, ma bisogna abilitarla nelle User Preferences del software. Di seguito, si analizzerà la differenza di struttura tra lo scheletro di Basic Human Meta-rig e quello modificato per l'utilizzo nell'add-on, a cui si associa per semplicità il nome **Gentle Meta-rig**.

DA BASIC HUMAN META-RIG A GENTLE META-RIG

Il Basic Human Meta-rig messo a disposizione da Rigify è una struttura parziale di scheletro umano in quanto non possiede alcuna caratteristica ossea facciale. Le differenze tra le due figure rappresentate di seguito è l'eliminazione di alcune ossa. In particolare, sono state rimosse le bone dei talloni, della punta del piede e del bacino.

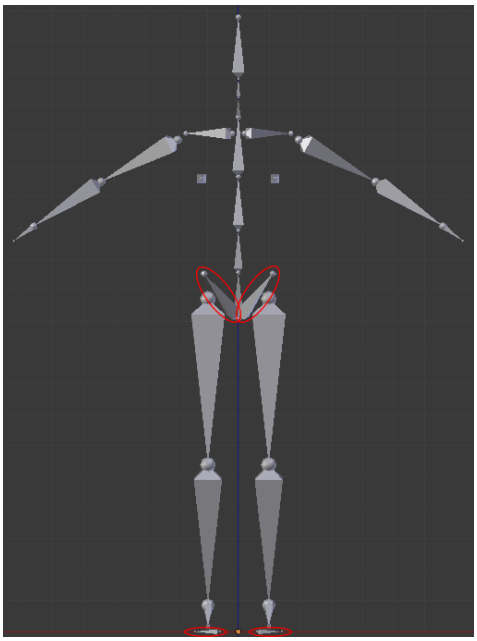


Figura 6.14: Basic Human Meta-rig

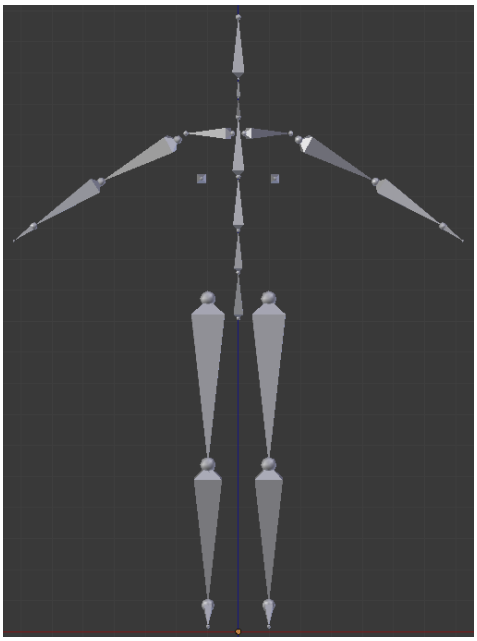


Figura 6.15: Gentle Meta-rig

Le differenze che emergono dall'analisi dei due scheletri sono:

- *Bacino:* le ossa del bacino nel Basic Human Meta-rig hanno lo scopo di collegare l'ultima sezione della spina dorsale alla parte superiore del femore. Questo legame non è stato ritenuto necessario perché, come si vedrà nella sezione *Bone Constraints*, il bacino verrà bilanciato tramite modificatori associati al femore. Per questo motivo, i pelvis dello scheletro sono stati eliminati.

- *Talloni e punta del piede:* la decisione di rimozione di queste parti è dovuta ad una conformità delle coordinate di Motion Capture con la struttura del piede. Nel primo tool di Gentle Mocap, i punti registrati nel piede sono due, la caviglia e la punta del piede. Per adattare questi due punti allo scheletro del Basic Human Meta-rig, è stata preferita la scelta di eliminare le bone citate. L'obbiettivo ultimo corrisponde ad una riproduzione del collegamento "caviglia-punta del piede" del Motion Capture quanto più fedele possibile

alla struttura ossea del piede del personaggio in Blender. Per ovviare alla problematica relativa alla stesura del piede degli insetti che è diversa da quella di un essere umano, si ricorre all'adattamento del movimento compiuto dal personaggio rispetto a quello catturato dal programma: il piede, parte del modello 3D, ripercorre l'azione della corrispettiva parte del corpo dell'attore che è stata registrata.

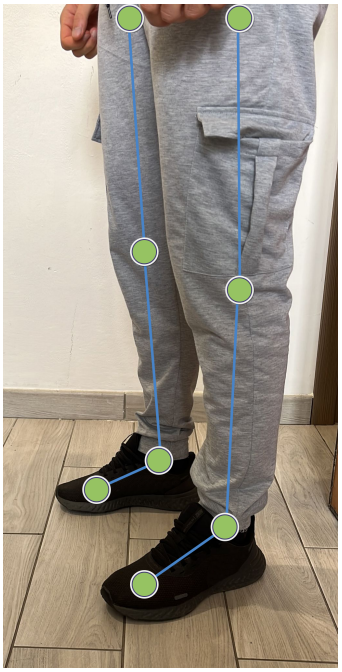


Figura 6.16: Motion Capture del piede

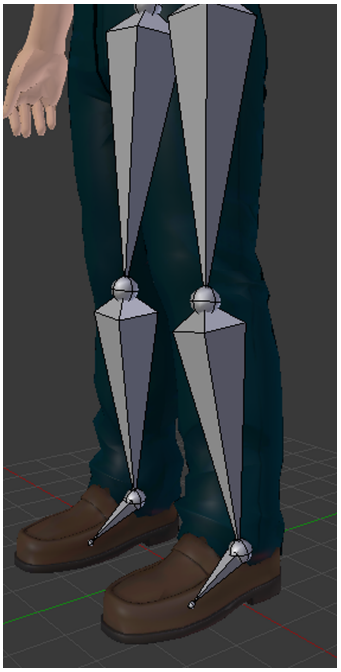


Figura 6.17: Rig del piede di un umano

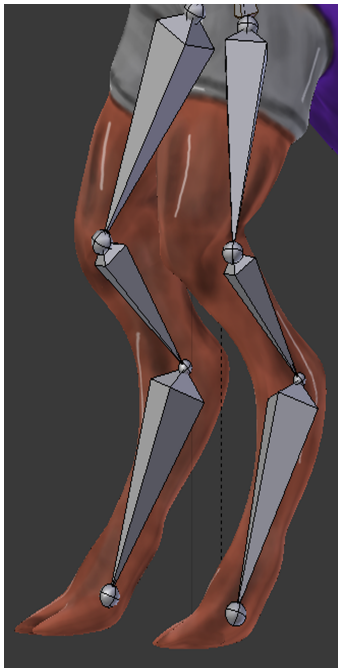


Figura 6.18: Rig del piede di un insetto

RENAMING OSSATURA PER LA COMPATIBILITA' CON UNREAL

Il nome delle ossa appartenenti allo scheletro generato dall'Armature Editor subiscono un renaming per mantenere la compatibilità con la nomenclatura di Unreal Engine. Nel motore di gioco si fa riferimento a un personaggio standard che è il **UE4_Mannequin_Skeleton**, in riferimento al quale le ossa sono state ridefinite per facilitarne l'importazione in Unreal. Questo permette l'adattamento automatico dello scheletro Gentle Meta-rig all'UE4_Mannequin_Skeleton, garantito dalla corrispondenza dei nomi delle ossa. Unreal Engine 4 fornisce kit gratuiti di animazioni per il Mannequin Skeleton come ad esempio camminata, corsa, ecc.. Grazie al renaming e alla similarità dei due scheletri, si possono utilizzare queste animazioni anche sui personaggi che utilizzano il Gentle Meta-Rig come scheletro. Si tratta, quindi, di un'aggiunta nel caso si vogliano sfruttare le animazioni

dell'UE4_Mannequin_Skeleton senza crearne nuove. Da notare come non siano state utilizzate animazioni fornite da Unreal nello sviluppo del videogioco, ma ogni animazione del personaggio è stata creata da zero tramite Motion Capture.

```
bones["spine.006"].name="head"
bones["spine.005"].name="neck_02"
bones["spine.004"].name="neck_01"
bones["spine.003"].name="spine_03"
bones["spine.002"].name="spine_02"
bones["spine.001"].name="spine_01"
bones["spine"].name="Pelvis"
bones["shoulder.R"].name="clavicle_R"
bones["upper_arm.R"].name="UpperArm_R"
bones["forearm.R"].name="lowerarm_r"
bones["hand.R"].name="Hand_R"
bones["shoulder.L"].name="clavicle_L"
bones["upper_arm.L"].name="UpperArm_L"
bones["forearm.L"].name="lowerarm_l"
bones["hand.L"].name="Hand_L"
bones["thigh.R"].name="Thigh_R"
bones["shin.R"].name="calf_r"
bones["foot.R"].name="Foot_R"
bones["thigh.L"].name="Thigh_L"
bones["shin.L"].name="calf_l"
bones["foot.L"].name="Foot_L"
```

Figura 6.19: Script di renaming delle ossa

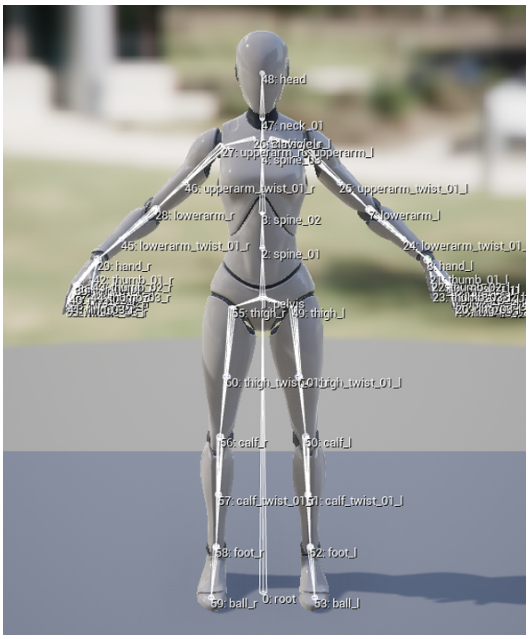


Figura 6.20: UE4 Female Mannequin Skeleton

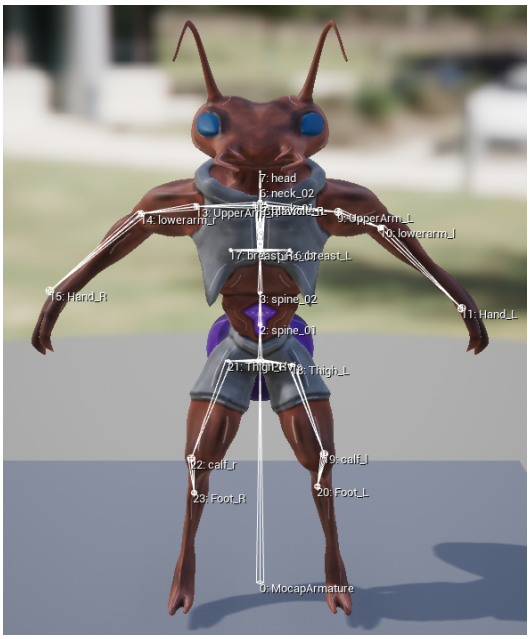
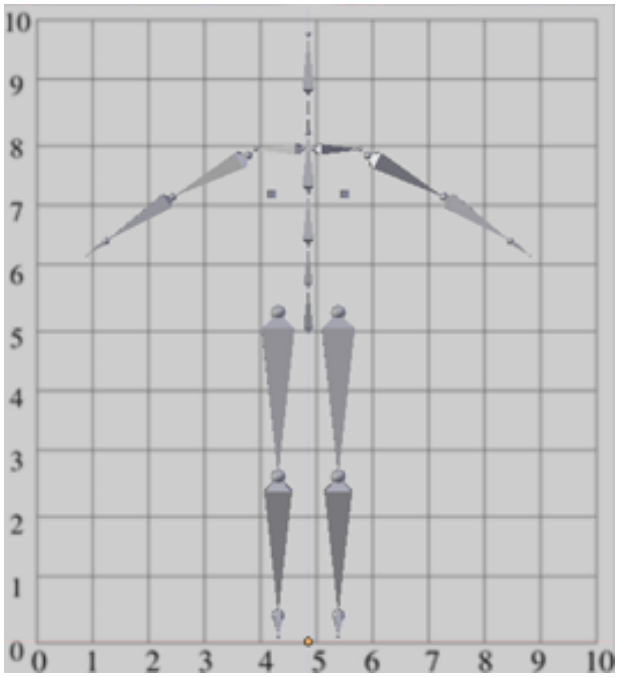


Figura 6.21: Gentle Meta-rig della formica

RIFERIMENTI DELLO SCHELETRO

Quando viene generata l'armatura, come riferimento in background, compaiono due immagini affiancate. Per visualizzarle è necessario sfruttare la vista ortogonale frontale. Le immagini sono:



- **Griglia di scala:** è la griglia che indica la grandezza dell'armatura in riferimento ad una scala che va da [0,10]. Lo scheletro generato apparirà davanti all'immagine per poter permettere al rigger l'impostazione della giusta dimensione.

Figura 6.22: Griglia di scala

- **Struttura del viso:** rappresenta la gerarchia dei punti facciali, che, in totale, sono 22 in quanto non vengono utilizzati tutti e 468 i riferimenti. Ad ogni landmark/bone del viso viene associato un numero che lo identifica. Il rigger o animator potrà ricavare vantaggio da questa legenda per poter accedere alla struttura del viso e dei nomi delle Shape Key.

Figura 6.23: Struttura del viso

6.3.3 Face Capture Editor

L'editor che si occupa del viso svolge il ruolo di analizzatore dei dati provenienti dal primo tool di cattura del volto. La sua funzione è convertire le coordinate facciali presenti all'interno del file di testo in animazione. Nel caso del Face Capture Editor, lo scopo non è quello di andare direttamente ad alterare la mesh facciale (in quanto questa caratteristica esecutiva è affidata allo "Shape Keys Creation"), bensì si occupa di preparare i dati del volto e memorizzarli in variabili, cosicché l'editor delle Shape Key possa generare le espressioni facciali tramite la lettura di tali informazioni. Inoltre, l'editor del viso permette di visualizzare con un'anteprima le ossa del volto che sono state in movimento durante la cattura. Di seguito, sono analizzati i comportamenti e le funzioni svolte dal Face Capture Editor.

CREAZIONE DELLO SCHELETRO FACCIALE

Le bone del viso che vengono generate sono, in totale, 22. Dei 468 landmark, ne vengono estrapolati soltanto una minima parte. L'utilizzo di pochi punti di riferimento non influisce sulla qualità dell' espressione facciale in quanto vengono sfruttati i 22 landmark che occupano le posizioni coinvolte maggiormente nel movimento, come la bocca e gli occhi. Le bone vengono imparentate all'osso centrale della testa in modo che lo spostamento di esso provochi un eguale traslazione alle ossa facciali.

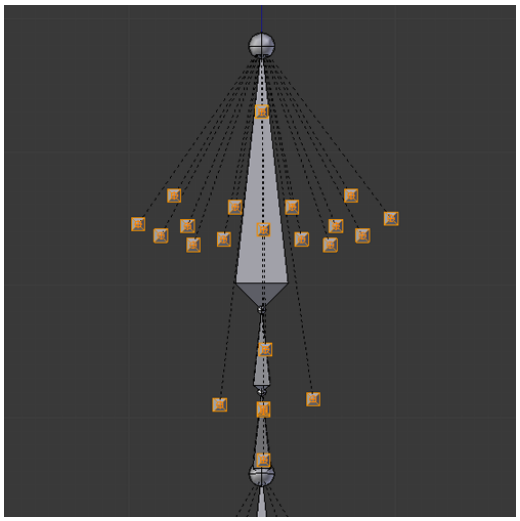


Figura 6.24: Ossatura del viso, vista frontale

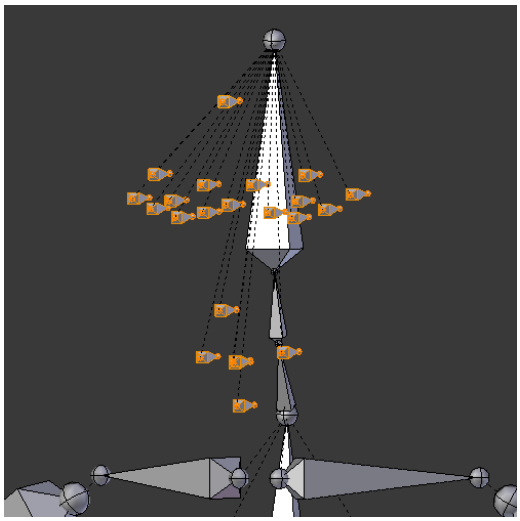


Figura 6.25: Ossatura del viso, vista prospettica

MOVIMENTO DEL VISO IN BASE AI LANDMARK

Dopo aver creato le ossa facciali, si passa alla lettura delle coordinate 3D all'interno del file di testo della cattura del volto. Il file viene scansionato riga per riga, dove ogni riga corrisponde ad un frame. Si tiene conto della precisione di analisi impostata dall'utente e del frame di partenza deciso. Gli altri parametri di offset verticale e dimensione del viso, vengono mantenuti e applicati sui landmark scansionati. A questo punto, si legge il file tramite un ciclo for. In particolare:

- *Primo ciclo:* il frame di partenza impostato dall'utente viene preso come riferimento per il posizionamento iniziale delle ossa facciali. In Edit Mode, si collocano le bone del volto stabilendo la struttura ossea della testa.
- *Secondo ciclo in poi:* a partire dal secondo frame, sono inseriti i Keyframe, ovvero i frame chiave che memorizzano il movimento di un osso facciale in quel preciso istante.
- *Termine del ciclo:* al termine del ciclo, ogni bone del volto ha una sua posizione nello spazio 3D, che cambia frame per frame. Per riprodurre il risultato, basta semplicemente effettuare il Play dell'animazione.

RICERCA DEL MASSIMO E DEL MINIMO VALORE DELLE ESPRESSIONI FACCIALI

Durante la lettura da file dei dati di cattura, viene svolta in parallelo una ricerca di valori massimi e minimi per ogni landmark facciale, ciascuno dei quali ha una sua posizione di massima e minima estensione. Si procede con una ricerca di questi valori per ottenere una corrispondenza tra la posizione del punto e il range di valori [0,1] assegnabili ad una shape key facciale. Ogni associazione (coordinata del punto, valore shape key) assume un significato diverso per i vari muscoli facciali:

- *Fronte:*
  - Minimo (fronte aggrottata): posizione verticale più bassa -> Valore Shape Key: 0
  - Massimo (fronte distesa): posizione verticale più alta -> Valore Shape Key :1
- *Sopracciglia:*
  - Minimo (sopracciglia abbassate): posizione verticale più bassa -> Valore Shape Key: 0
  - Massimo (sopracciglia alzate): posizione verticale più alta -> Valore Shape Key :1

- *Occhi:*
  - Palpebra superiore:
    - Minimo (quando la palpebra è chiusa): posizione verticale più bassa -> Valore Shape Key: 0
    - Massimo (quando la palbebra è aperta): posizione verticale più alta -> Valore Shape Key: 1
  - Palpebra inferiore:
    - Minimo (quando la palbebra è chiusa): posizione verticale più alta -> Valore Shape Key: 0
    - Massimo (quando la palbebra è aperta): posizione verticale più bassa -> Valore Shape Key: 1
- *Punta del naso:*
  - Minimo (punta del naso abbassata): posizione verticale più bassa -> Valore Shape Key: 0
  - Massimo (punta del naso sollevata): posizione verticale più alta -> Valore Shape Key: 1
- *Bocca:*
  - Labbro superiore:
    - Minimo (quando il labbro è chiuso): posizione verticale più bassa -> Valore Shape Key: 0
    - Massimo (quando il labbro è aperto): posizione verticale più alta -> Valore Shape Key: 1
  - Labbro inferiore:
    - Minimo (quando il labbro è chiuso): posizione verticale più alta -> Valore Shape Key: 0
    - Massimo (quando il labbro è aperto): posizione verticale più bassa -> Valore Shape Key: 1
  - Fine della bocca a destra:
    - Minimo (quando la fine della bocca è verso l'interno): posizione orizzontale maggiore -> Valore Shape Key: 0
    - Massimo (quando la fine della bocca si distende): posizione orizzontale minore -> Valore Shape Key: 1
  - Fine della bocca a sinistra:

* Minimo (quando la fine della bocca è verso l'interno): posizione orizzontale minore -> Valore Shape Key: 0
* Massimo (quando la fine della bocca si distende): posizione orizzontale maggiore -> Valore Shape Key: 1

DRIVER DI ROTAZIONE DELLA TESTA

Un driver di rotazione viene associato all'osso della testa sull'asse verticale di rotazione. In particolare, il bone della testa ruota di un certo valore in base alla traslazione sull'asse orizzontale del landmark n.10 della fronte (vedere *Struttura del viso*). Inoltre, è stato scelto come riferimento il punto della fronte in quanto occupa una posizione centrale del viso. Se tale punto trasla a destra o sinistra, di conseguenza la testa ruoterà per seguirne il movimento.

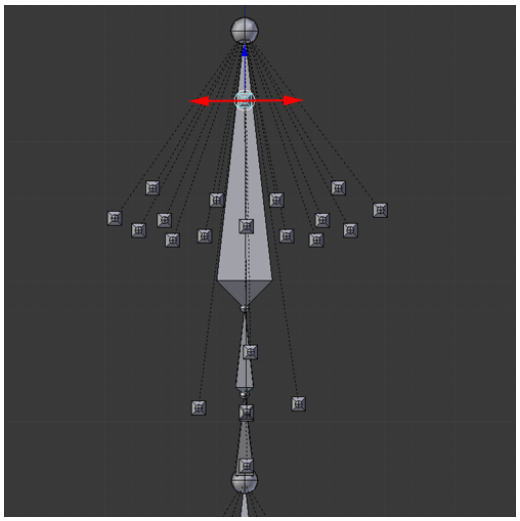


Figura 6.26: Rotazione della testa nulla

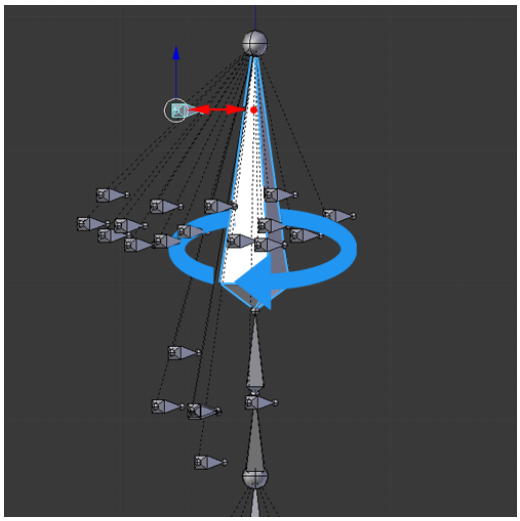


Figura 6.27: Rotazione della testa in base al punto della fronte

SMOOTH DEI MOVIMENTI

Per smussare e rendere più fluidi i movimenti delle ossa facciali, si applica il metodo "Smooth Keys", tecnica che agisce su F-Curve presenti all'interno del Graph Editor di Blender. Una F-Curve facciale è una funzione curva che rappresenta graficamente le posizioni delle varie ossa del volto frame per frame. Vengono applicati degli algoritmi che vanno a smussare i punti sulla curva, tenendo conto sia della distanza tra ogni frame, sia del valore lineare medio della curva. Il vantaggio è l'eliminazione del tremolio causato dalla cattura tramite Motion Capture.

6.3.4 Body Capture Editor

L'editor del corpo ha il compito di convertire le coordinate di cattura derivanti dal primo tool di Motion Capture in animazione dello scheletro. L'operator si occupa di impostare i vincoli delle ossa, in quanto esse devono seguire i movimenti dei landmark con delle regole specifiche. Tramite la lettura del file di cattura, i dati di Motion Capture vengono applicati alla struttura dello scheletro che si muove in base al valore delle coordinate all'interno dello spazio 3D.

BONE CONSTRAINTS

Alle ossa dello scheletro Gentle Meta-rig vengono assegnati dei vincoli, i cosiddetti "Bone Constraints" che, in Blender, possono essere utilizzati per controllare i gradi di libertà delle bone, vincolando e limitando i movimenti. Si possono anche adoperare delle costrizioni per fare in modo che un osso dell'armatura segua un altro oggetto/osso. Inoltre, si dispone della funzione di cinematica inversa (IK) che permette il raggiungimento di una posa desiderata agendo sulle estremità di uno scheletro articolato. I vincoli utilizzati sullo scheletro del corpo sono:

- **IK:** è l'acronimo di Inverse Kinematics. Si tratta della cinematica inversa che, agendo su uno scheletro articolato e flessibile, permette di raggiungere una determinata posa secondo parametri di calcolo sulle angolazioni e sul movimento più adeguato.

- **Copy Location:** è un vincolo di trasformazione che copia la locazione di un oggetto/osso di riferimento. In questo caso un certo osso è vincolato nel replicare la stessa posizione del landmark di cattura ad esso associato. E' necessario specificare il target di riferimento da seguire.

- **Damped Track:** è un vincolo di tracking che obbliga l'asse di un osso al puntamento di un target specifico. Il constraint utilizza una tipologia di rotazione denominata Swing, riferita ad un asse verso una specifica direzione. In questo caso, si predilige il percorso di rotazione più breve possibile attorno tale asse. Il vincolo è sfruttato per allineare le ossa, puntando e seguendo i landmark di cattura del corpo.

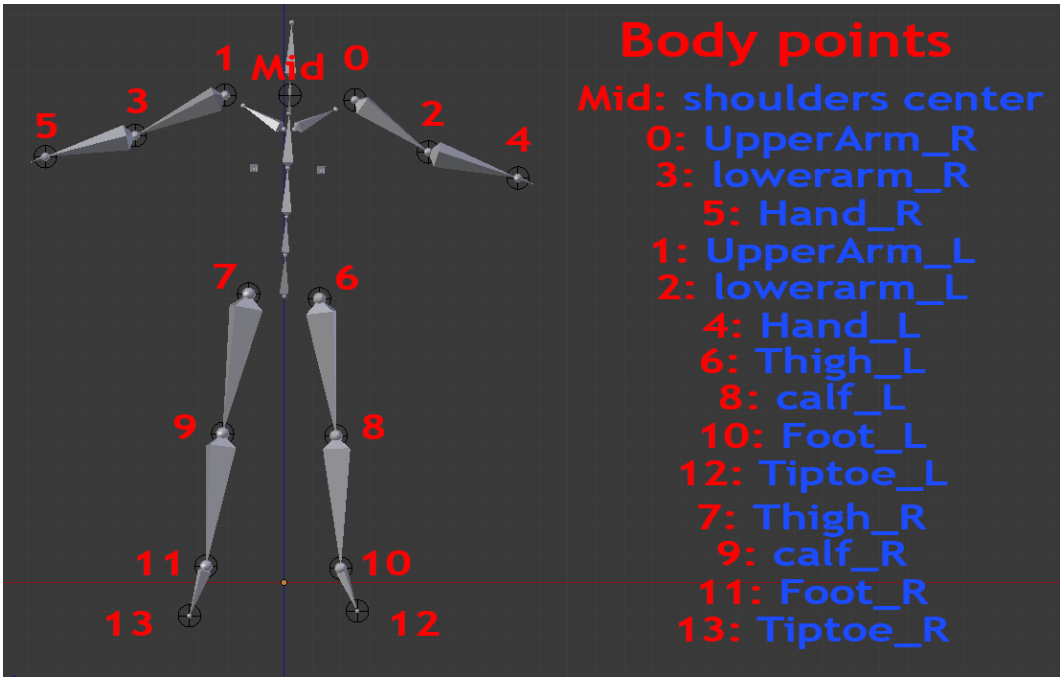


Figura 6.28: Struttura del corpo

Le ossa del Gentle Meta-rig a cui viene impostato un vincolo sono descritte di seguito:

- **Testa:**
  - *IK:* cinematica inversa con punto di partenza posizionato nello scheletro della testa e con una catena articolata, avente una lunghezza di 7 bone. L'estremità finale di questa catena è il lombare, la sezione terminale della spina dorsale.
- **Braccio sinistro:**
  - *Clavicola:*
    - *Damped Track:* tracking della clavicola in direzione del punto 0 del corpo, ovvero la spalla.
  - *Braccio superiore:*
    - *Copy Location:* la parte iniziale dell'osso del braccio superiore è caratterizzata dal vincolo di condividere la stessa posizione del punto 0 dello scheletro.
    - *Damped Track:* tracking del braccio superiore in direzione del punto 2 del corpo, ovvero il gomito.

  - *Avambraccio:*
    - *Damped Track:* tracking dell'avambraccio in direzione del punto 4 del corpo, ovvero il polso della mano.

- **Braccio destro:**
  - *Clavicola:*
    - *Damped Track:* tracking della clavicola in direzione del punto 1 del corpo, ovvero la spalla.
  - *Braccio superiore:*
    - *Copy Location:* : la parte iniziale dell'osso del braccio superiore è caratterizzata dal vincolo di condividere la stessa posizione del punto 1 dello scheletro.
    - *Damped Track:* tracking del braccio superiore in direzione del punto 3 del corpo, ovvero il gomito.
  - *Avambraccio:*
    - *Damped Track:* tracking dell'avambraccio in direzione del punto 5 del corpo, ovvero il polso della mano.

- **Bacino:**
  si esegue un'operazione di *bilanciamento del bacino*: l'obiettivo è conservare la centralità del bacino (osso Pelvis) rispetto al punto della coscia destra (osso Thigh_R) e della coscia sinistra (osso Thigh_L). Adottando questa soluzione, la zona lombare rimane al centro delle gambe evitando spostamenti bruschi non realistici. La centralità dell'osso Pelvis ha delle ripercussione anche sull'intera colonna vertebrale collegata ad esso: per far sì che la sezione lombare sia sempre nel punto medio rispetto alla parte superiore delle gambe, si vincola l'osso Pelvis ad essere influenzato al 50% dalla posizione dell'osso Thigh_L e al 50% dalla posizione dell'osso Thigh_R.

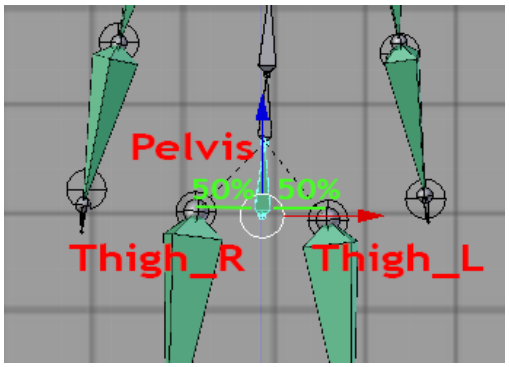


Figura 6.29: Bilanciamento del bacino

  - *Copy Location coscia destra / coscia sinistra:* si vincola la posizione ad essere influenzata dal 50% dalla parte iniziale dell'osso della coscia destra e dall'altro 50% dalla parte iniziale dell'osso della coscia sinistra.
  - *Damped Track:* tracking dell'osso centrale del bacino in direzione del punto Mid del corpo, ovvero il punto centrale del petto.

- **Gamba sinistra:**

- *Coscia:*

  - *Copy Location:* : la parte iniziale dell'osso della coscia è caratterizzata dal vincolo di condividere la stessa posizione del punto 6 dello scheletro.
  - *Damped Track:* tracking della coscia in direzione del punto 8 del corpo, ovvero il ginocchio.

- *Polpaccio:*

  - *Damped Track:* tracking del polpaccio in direzione del punto 10 del corpo, ovvero la caviglia.

- *Piede:*

  - *Damped Track:* tracking del piede in direzione del punto 12 del corpo, ovvero la punta del piede.

- **Gamba destra:**

- *Coscia:*

  - *Copy Location:* la parte iniziale dell'osso della coscia è caratterizzata dal vincolo di condividere la stessa posizione del punto 7 dello scheletro
  - *Damped Track:* tracking della coscia in direzione del punto 9 del corpo, ovvero il ginocchio.

- *Polpaccio:*

  - *Damped Track:* tracking del polpaccio in direzione del punto 11 del corpo, ovvero la caviglia.

- *Piede:*

  - *Damped Track:* tracking del piede in direzione del punto 13 del corpo, ovvero la punta del piede.

MOVIMENTO DEL CORPO IN BASE AI LANDMARK

Il file di cattura del corpo derivante dal primo tool di Motion Capture viene letto per estrapolare le coordinate 3D dei landmark, le quali vengono convertite in riferimento allo spazio 3D di Blender e utilizzate per muovere lo scheletro del corpo. L'utente, inizialmente, decide il frame di partenza da cui scansionare i landmark e la precisione di replica dei movimenti. Inoltre, durante l'elaborazione dei dati, si terrà conto dei settaggi di adattamento dello scheletro che sono stati scelti. La lettura del file avviene tramite un ciclo for secondo le seguenti modalità:

- *Primo ciclo:* il primo vero ciclo, degno di nota, si ha quando il puntatore della riga nel file corrisponde al frame di partenza impostato dall'utente. In tale frame, vengono scansionate le prime 14 coordinate (empty object) del corpo per poi collocarle, in Edit Mode, nello spazio, in modo tale che fungano da riferimento per le ossa. La motivazione di quanto detto risiede nel fatto che le bone dovranno seguire i movimenti degli empty object tramite i vincoli descritti precedentemente di Copy Location e Damped Track. Con la lettura dei dati appartenenti al primo frame, viene generato anche un quindicesimo empty object che è il "Mid", ovvero un punto centrale del petto, generato calcolando il punto medio tra spalla destra e sinistra. "Mid" rappresenta un importante riferimento all'osso centrale del bacino. Inoltre, parallelamente alla creazione di questi empty, viene generata in edit mode la forma starter dello scheletro del corpo.

- *Secondo ciclo in poi:* dal secondo frame in poi, gli empty object vengono dapprima fissati nello spazio tramite Keyframe ed in seguito vengono assegnati dei frame chiave solo agli empty e non alle bone, in quanto le ossa puntano e copiano la locazione di questi punti di riferimento in modo automatico tramite constraint. Il valore di posizionamento del "Mid" point viene aggiornato continuamente prendendo il valore medio tra spalla sinistra e destra.

- *Termine del ciclo:* terminata la lettura delle coordinate da file, le ossa hanno un loro movimento vincolato dai bone constraint. A questo punto, si può assistere al movimento dell'intero scheletro corporeo eseguendo il Play dell'animazione.

SMOOTH DEI MOVIMENTI

Per eliminare il tremolio derivante dalla registrazione dei movimenti tramite il processo di Motion Capture, si applica la tecnica di "Smooth Keys" sulla F-Curve. Questa funzione curva visualizzabile nel Graph Editor è composta da vari punti che occupano delle posizioni specifiche del corpo nei vari frame. Con il metodo di smoothing, si rende il movimento dello scheletro corporeo più stabile e fluido.

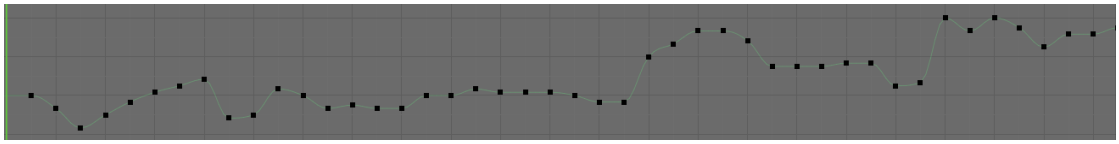


Figura 6.30: F-Curve senza smoothing

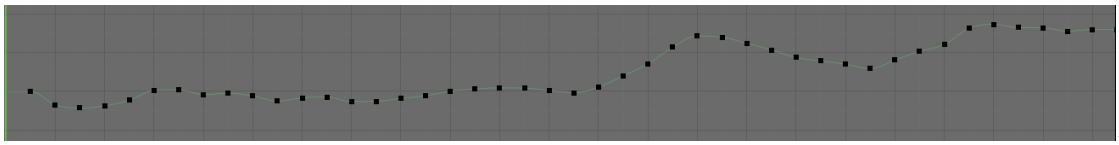


Figura 6.31: F-Curve con smoothing

MOVIMENTI DEL CORPO OTTENUTI SUGLI INSETTI DEL VIDEOGIOCO

Si propongono, di seguito, alcune immagini di posa degli insetti:

Figura 6.32: Posa Formica

Figura 6.33: Posa Ape

Figura 6.34: Posa Scarabeo

Figura 6.35: Posa Ragno

Figura 6.36: Posa Mantide

Figura 6.37: Posa Coccinella

6.3.5 Hand Capture Editor

L'editor di gestione della mano si occupa di dare un feedback visivo del movimento delle articolazioni al rigger e all'animator. Non si tratta di costruire e muovere le mani dello scheletro, in quanto sono state considerate fisse ai fini del videogioco; sono bensì analizzati i gesti delle mani che saranno utilizzati all'interno dello Shape Keys Creator. In questo modo sarà possibile muovere componenti tipiche di un insetto che non sono comuni con il corpo umano, come le ali, le antenne, le mandibole o le zampe. Questo operator di Hand Capture, oltre a fornire una rappresentazione

visiva della mano tramite punti (empty object), svolge il ruolo fondamentale nel salvataggio delle coordinate 3D in array per renderle poi disponibili all'editor di Shape Key. L'Hand Capture Editor svolge il ruolo di preparatore dei dati in vista relativi alla creazione delle Shape Key per l'alterazione di appendici dell'insetto.

RAPPRESENTAZIONE VISUALE DELLA MANO

La mano catturata nel primo tool deve essere replicata all'interno di Blender. I movimenti della mano non necessitano della creazione di ossa, in quanto si vuole fornire solo una rappresentazione visiva di come le articolazioni cambino la loro posizione nello spazio. Per questo motivo, sono creati 21 empty object per la mano, che seguono il movimento dei landmark corrispondenti letti da file.

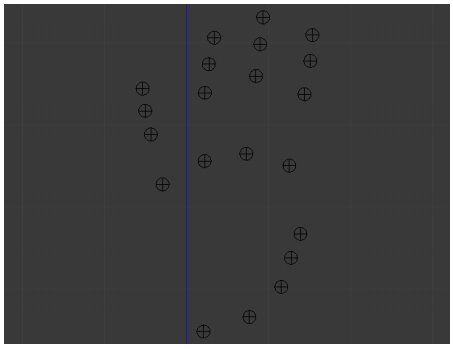


Figura 6.38: Empty object della mano

MOVIMENTO DELLA MANO IN BASE AI LANDMARK

In seguito alla creazione dei 21 empty object per la mano, è necessario animarli in base al movimento delle articolazioni registrate durante la sessione di Motion Capture. Si scansiona quindi il file, frame per frame, estrapolando le coordinate 3D e convertendole in movimento nella rappresentazione visiva della mano. Durante la lettura dei landmark si tiene conto dei settaggi stabiliti dall'utente, come il frame di partenza, l'altezza a cui la mano deve essere rappresentata e la relativa profondità. Si legge il file tramite un ciclo for. In particolare:

- *Primo ciclo:* il primo frame, impostato dall'utente, serve a settare il punto più centrale della mano come riferimento per l'intera mano. Questo significa che la mano si muoverà rispettando l'offset del punto considerato come centro. L'origine considerata è il punto più basso del palmo, in quanto risulta essere

la metà che divide in due la mano.

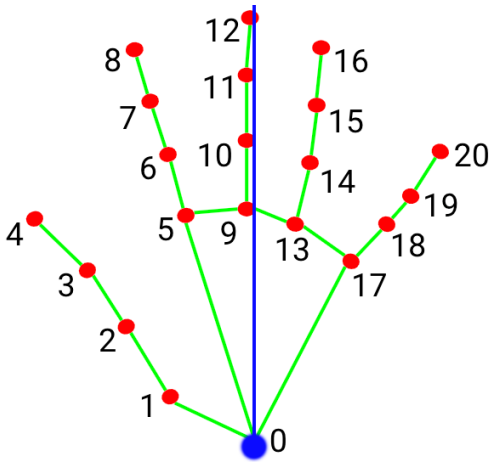


Figura 6.39: Centro della mano considerato

- *Secondo ciclo in poi:* dal secondo frame, le coordinate prelevate dal file di cattura sono convertite nello spazio 3D di Blender. In seguito, saranno inserite in array e verranno utilizzate dallo Shape Key Creator per l'alterazione della mesh di parti esterne dell'insetto. Sono inseriti inoltre i Keyframe in base al valore della coordinata associata, in modo tale da impostare e salvare la posizione di un determinato empty object in quel preciso frame.

- *Termine del ciclo:* terminato il ciclo, il movimento della mano è stato memorizzato e può essere riprodotto visivamente eseguendo il Play dell'animazione.

SMOOTH DEI MOVIMENTI

Il movimento degli empty object che forniscono una rappresentazione visiva della mano subisce un miglioramento. Lo spostamento dei punti viene infatti pulito da bruschi cambi di posizione che si verificano a causa del tremolio derivante dalla cattura della mano durante la sessione di Motion Capture. Grazie alla tecnica di "Smooth Keys" i punti di riferimento delle articolazioni si muovono con maggiore fluidità.

6.3.6 Shape Keys Creator

Le shape key sono salvataggi di forme chiave che corrispondono ad un'alterazione della mesh di un modello 3D. La forma memorizzata è uno stato della mesh che cambia in base ad un valore intero che in questo caso è stato considerato tra [0,1]. Quando lo stato è a 0, la mesh è nella sua forma originale, quando lo stato è a 1 la mesh assume la forma impostata dal modellatore. Variando lo stato tra 0 e 1, la

mesh subisce delle modifiche per raggiungere la shape desiderata. Lo Shape Key Creator utilizza questa metodologia per animare:

- **Espressioni facciali:** si permettere la gestione e la replica delle espressioni facciali derivanti dal Motion Capture del viso. Nel caso degli insetti, è compito dell'animatore associare le shape key ad una specifica espressione sul muso dell'insetto.

- **Parti esterne dell'insetto:** è consentita la modifica di parti corporee esterne dei personaggi del videogioco in base alla cattura dei gesti delle mani.

L'utente che utilizza Gentle Mocap, può impostare nell'editor la tipologia di mesh sulla quale si vogliono aggiungere le shape key del viso e/o dell'insetto. I tipi di superficie 3D accettabili sono sette:

1. **Human:** è una mesh umana che utilizza delle shape key per le espressioni facciali.

2. **Bee:** è la mesh dell'ape che, oltre alle shape key del viso utilizza "Wing_Flap" per la gestione del battito alare.

3. **Mantis:** è la mesh della mantide che, oltre alle shape key del viso utilizza "Wing_Flap" per la gestione del battito alare.

4. **Ladybug:** è la mesh della coccinella che, oltre alle shape key del viso utilizza "Wing_Flap" per la gestione del battito alare.

5. **Beetle:** è la mesh dello scarabeo che, oltre alle shape key del viso utilizza "Mandible_Horns" per la gestione dell'allungamento e chiusura delle mandibole.

6. **Spider:** è la mesh del ragno che, oltre alle shape key del viso utilizza "Paws1", "Paws2", "Paws3" e "Paws4" per la gestione del movimento delle 8 zampe nel retro del corpo.

7. **Ant:** è la mesh della formica che, oltre alle shape key del viso utilizza "Antenna1", "Antenna2" per la gestione dell'allungamento delle due antenne sulla testa.

8. **Generic insect:** è la combinazione delle precedenti Shape Key per generare un generico insetto di cui si possono alterare le ali, le mandibole, le zampe e le antenne.

SHAPE KEY PER LE ESPRESSIONI FACCIALI

L'animazione delle espressioni facciali fa riferimento alla mesh di un viso umano o del volto di un insetto. Le shape key generate dallo script assumono, frame per frame, degli stati ben precisi nell'intervallo [0,1] in base al movimento associato ad un determinato punto del viso.
Come visto precedentemente nel "Face Capture Editor", sono stati ricercati i massimi e i minimi valori delle espressioni facciali analizzando l'intera durata della cattura. Utilizzando questi valori come riferimento, il software è in grado di comprendere se un punto del viso è nella sua massima estensione oppure è rilassato. Prendiamo per esempio la palpebra superiore dell'occhio: se la palpebra è rilassata avrà il valore minimo 0, se la palpebra è aperta avrà il valore massimo 1. Lo stesso ragionamento è valido per tutte le altre shape key.
Una volta create le shape key facciali, l'animator, selezionandone una ed entrando in Edit Mode, può alterare la mesh della zona del viso desiderata e decidere che forma deve acquisire quando la shape key è nello stato 1.

- **Elenco shape key per il viso**
  Le shape key messe a disposizione dal tool per la gestione del viso sono in totale 22, in numero uguale ai landmark considerati per la creazione delle ossa facciali. Esse hanno dei nomi specifici in riferimento alla zona del viso che si vuole controllare:

  1. *Basis:* forma base del viso, senza alcuna manipolazione.
  2. *eyebrowR_r:* parte finale del sopracciglio destro.
  3. *eyebrowR_up:* parte centrale del sopracciglio destro.
  4. *eyebrowR_l:* parte interna del sopracciglio destro.
  5. *eyeR_l:* angolo esterno dell'occhio destro.
  6. *eyeR_r:* angolo interno dell'occhio destro.
  7. *eyeR_up:* palpebra superiore dell'occhio destro.
  8. *eyeR_down:* palpebra inferiore dell'occhio destro.
  9. *eyebrowL_r:* parte interna del sopracciglio sinistro.
  10. *eyebrowL_up:* parte centrale del sopracciglio sinistro.
  11. *eyebrowL_l:* parte finale del sopracciglio sinistro.
  12. *eyeL_l:* angolo interno dell'occhio sinistro.
  13. *eyeL_r:* angolo esterno dell'occhio sinistro.
  14. *eyeL_up:* palpebra superiore dell'occhio sinistro.
  15. *eyeL_down:* palpebra inferiore dell'occhio sinistro.

16. *eyeMiddle:* parte centrale tra i due occhi.
17. *mouthR:* angolo destro della bocca.
18. *mouthL:* angolo sinistro della bocca.
19. *mouthUp:* labbro superiore della bocca.
20. *mouthDown:* labbro inferiore della bocca.
21. *forehead:* fronte.
22. *chin:* mento.

- **Driver per la gestione delle shape key**
  Il valore delle shape key nel range [0,1] è gestito da un driver di traslazione che, in base al movimento dei punti facciali, è in grado di calcolare uno stato tra 0 e 1 per le shape key del viso. Un driver contiene al suo interno un'espressione matematica che converte il movimento di traslazione di un landmark del viso in un valore associabile alla shape key interessata.
  A questo scopo, è stata creata una funzione **create_driver_face** che riceve in ingresso cinque parametri:

  – *shape_key:* è la shape key di una zona del viso a cui si vuole assegnare frame per frame uno stato nel range [0,1] in base al modo in cui il punto facciale in quella zona si muove nel tempo.
  – *target_name:* è il punto facciale preso come riferimento per calcolare il valore della shape key. Se, ad esempio, si prende come target_name il riferimento a "FacePoint 13" (vedere *Struttura del viso*), allora ci si aspetta di modificare una shape key "mouthUp" passata come parametro. Questo implica che, in base a come si muove il Facepoint del labbro superiore, cambia di conseguenza la posizione del labbro tramite lo stato della shape key assegnata.
  – *transform_type:* indica quale asse di traslazione tener conto per il movimento del punto di riferimento del viso "target_name". Se si passa come parametro "Z" (l'asse verticale) allora il calcolo della shape key dipende dal modo in cui si muove in verticale il Facepoint "target_name". E' vantaggioso considerare la traslazione sull'asse Z per la chiusura/apertura degli occhi e della bocca, come anche per l'inarcatura delle sopracciglia. Se invece si passa come "transform_type" il valore "X" allora sarà considerato il movimento orizzontale come riferimento al punto facciale. La traslazione sull'asse X è utile nei casi in cui gli angoli della bocca o degli occhi si estendano verso l'esterno oppure si comprimano verso l'interno.
  – *transform_space:* indica se considerare lo spazio globale o locale di Blender. In questo caso, è stato passato il parametro "World_Space" come transform_space.

– *invert_result:* è un valore booleano che se impostato a "False" permette di considerare il movimento invertito del punto facciale di riferimento lungo l'asse "transform_type". Se il parametro passato è "True" allora viene considerato il movimento normale senza invertire l'asse di traslazione.

Il calcolo del valore della shape key è dato da un'espressione matematica che tiene conto del massimo e del minimo delle espressioni facciali calcolati nel "Face Capture Editor" e della posizione del punto di riferimento del viso.

```
1 diff=abs(max_face_value[target_name]-min_face_value[target_name])

2 if invert_result==True:
      drv.expression = '((%f - %s)/ %f)' % (max_face_value[target_name],var.name,diff)
  else:
      drv.expression = '1-((%f - %s)/ %f)' % (max_face_value[target_name],var.name,diff)
```

Figura 6.40: Espressione matematica del driver del viso

1. **Calcolo della differenza (diff):** viene calcolato il valore assoluto della differenza tra la posizione massima raggiunta dal punto di riferimento sull'asse considerato e la sua posizione minima. Questa differenza tra massimi e minimi è utile poiché rappresenta un confronto col range [0,1] che contiene i massimi e i minimi valori delle shape key.
2. **Espressione del driver:** l'espressione matematica del driver assume due formule diverse in base alla scelta di considerare l'asse di riferimento invertito o meno:
   – Se si considera l'asse invertito:

   $$expression = (max\_face\_value[target\_name] - var)/diff$$

   – Se si considera l'asse originale:

   $$expression = 1 - (max\_face\_value[target\_name] - var)/diff$$

   L'espressione matematica restituisce un valore compreso tra [0,1] in base al movimento di un determinato punto facciale di riferimento. Questo parametro del driver monitora lo stato della shape key modificandone il valore frame per frame. Sarà, poi, l'utente ad associare a quella shape key la forma desiderata per visualizzarne i risultati.

- **Espressioni facciali sui personaggi ottenute tramite Shape Key**
  Di seguito sono rappresentate alcune espressioni facciali ottenute per ogni insetto:

  – **Ape**

Figura 6.41: Espressione 1 ape:
-Antenne abbassate
-Muso abbassato

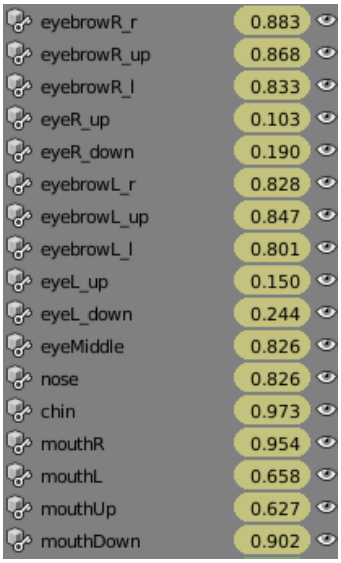


Figura 6.42: Shape key 1 ape

Figura 6.43: Espressione 2 ape:
-Antenne alzate
-Muso sollevato

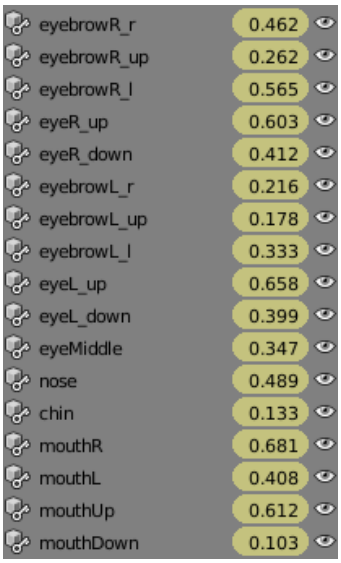


Figura 6.44: Shape key 2 ape

– **Mantide**

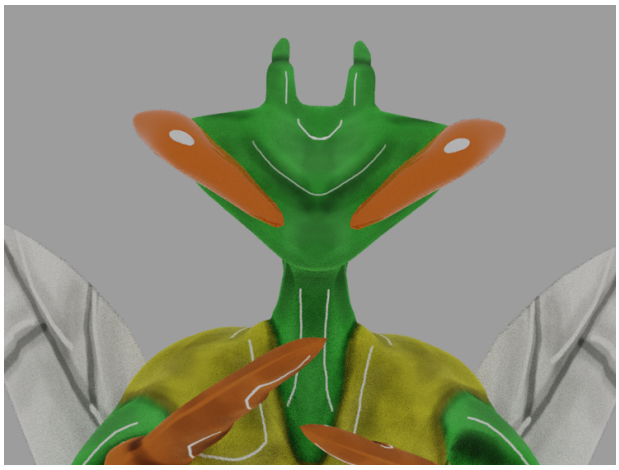


Figura 6.45: Espressione 1 mantide:
-Pupille compresse
-Antenne sollevate
-Muso abbassato

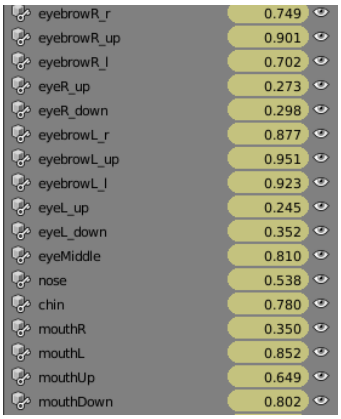


Figura 6.46: Shape key 1 mantide

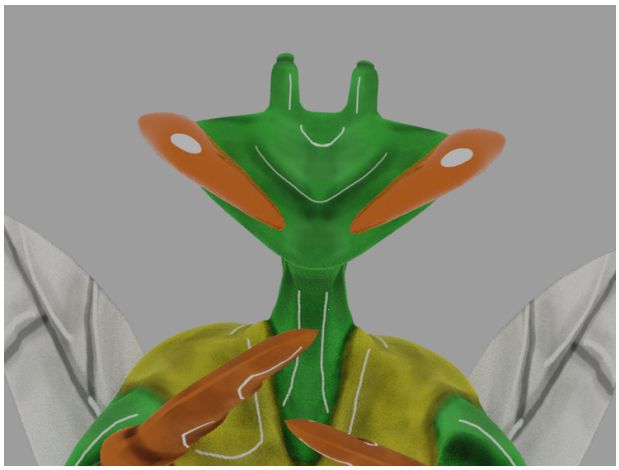


Figura 6.47: Espressione 2 mantide:
-Pupille dilatate
-Antenne abbassate
-Muso rialzato

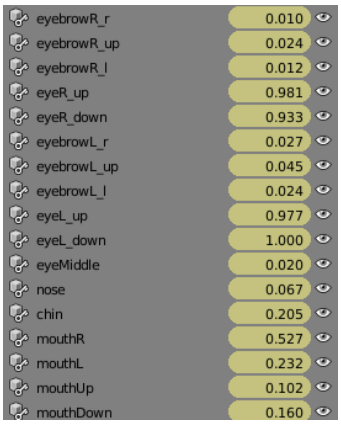


Figura 6.48: Shape key 2 mantide

– **Coccinella**

Figura 6.49: Espressione 1 coccinella: -Mandibole ravvicinate

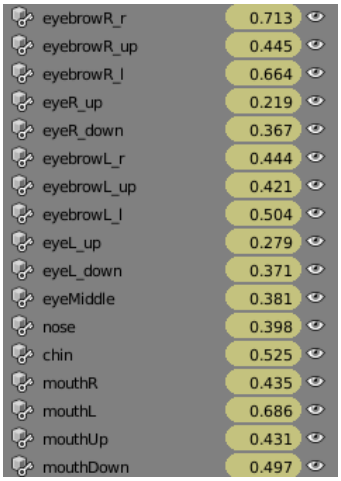


Figura 6.50: Shape key 1 coccinella

Figura 6.51: Espressione 2 coccinella: -Mandibole aperte

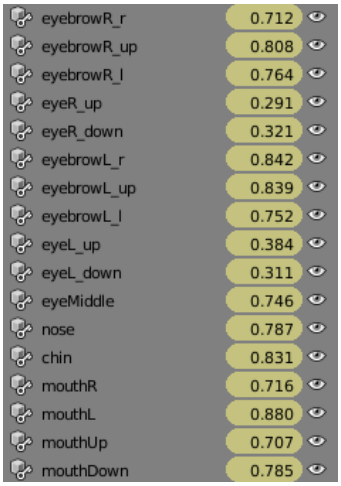


Figura 6.52: Shape key 2 coccinella

– **Scarabeo**

Figura 6.53: Espressione 1 scarabeo: -Bocca chiusa

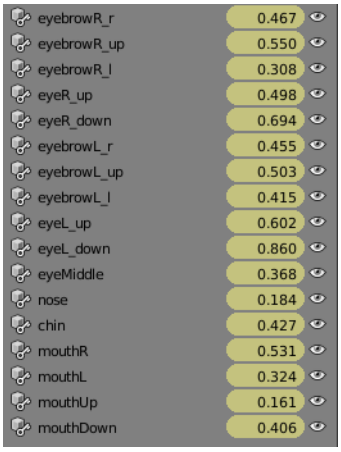


Figura 6.54: Shape key 1 scarabeo

Figura 6.55: Espressione 2 scarabeo: -Bocca aperta

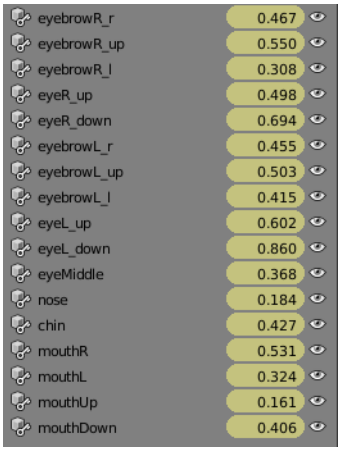


Figura 6.56: Shape key 2 scarabeo

– **Ragno**

Figura 6.57: Espressione 1 ragno:
-Mandibole ravvicinate e chiuse

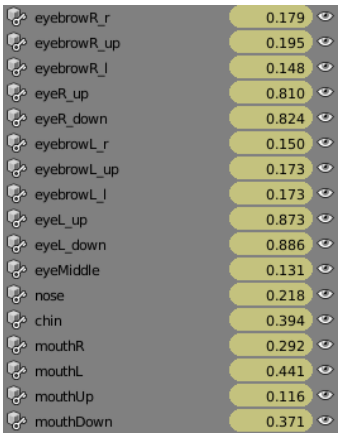


Figura 6.58: Shape key 1 ragno

Figura 6.59: Espressione 2 ragno:
-Mandibole distanziate e allungate

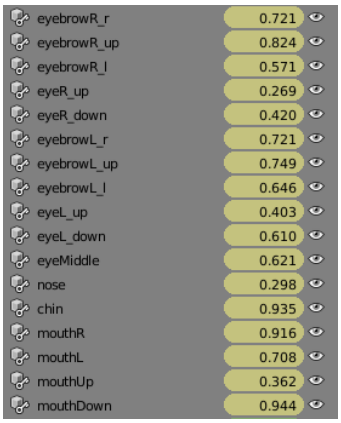


Figura 6.60: Shape key 2 ragno

– **Formica**

Figura 6.61: Espressione 1 formica:
-Mandibole chiuse

Figura 6.62: Shape key
1 formica

Figura 6.63: Espressione 2 formica:
-Mandibole aperte

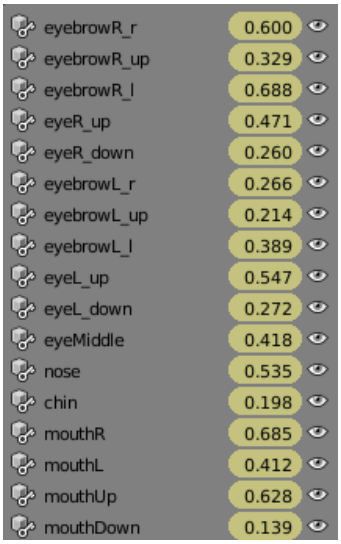


Figura 6.64: Shape key
2 formica

SHAPE KEY PER LE PARTI ESTERNE DELL'INSETTO

- **Ape, mantide e coccinella: Shape Key per il battito alare**
  Il controllo del battito alare dei personaggi ape, mantide e coccinella è gestito dalla chiusura e apertura della mano. In particolare, vengono considerati i movimenti delle quattro dita, cioè indice, medio, anulare e mignolo, escludendo il pollice. Quindi la mano viene considerata chiusa quando tutte e quattro le dita sono piegate verso l'interno del palmo, mentre la mano è riconosciuta come aperta quando le quattro dita sono rivolte verso l'esterno. Il movimento articolato è esaminato sull'asse Z verticale di Blender, in modo che vengano rilevate le dita quando sono verso l'alto (apertura della mano) o verso il basso (chiusura della mano).
  La Shape Key che controlla il battito alare è **Wing_Flap**, che può assumere un valore di stato:

  - **0:** se la mano è chiusa.
  - **1:** se la mano è aperta.
  - **Tra 0 e 1:** se la mano non è completamente aperta o chiusa.

  - **Funzione per la gestione della chiusura e apertura della mano**
    Per quanto riguarda il battito alare, è stata definita una funzione che si occupa di analizzare i punti in movimento della mano e convertire le loro posizioni in un valore [0,1] per la shape key.
    La funzione definita è BEE_MANTIS_LADYBUG_hand_closed_opened che riceve come parametro di ingresso la shape key "Wing_Flap" a cui assegnare un valore di stato ad ogni frame in base alla mano (se aperta o chiusa). Ogni singolo dito della mano viene analizzato, nonostante abbia compiuto solo un parziale movimento di chiusura o di apertura verso l'alto. Si può, quindi, dedurre che ogni dito influenza il valore della shape key e si può concludere che:

    * solo quando tutte le dita sono completamente distese la mano si può considerare aperta.
    * solo quando tutte le dita sono completamente piegate si può considerare la mano chiusa.

```
value=1
#ogni dito abbassato toglie -0.25 al value massimo (1) della mano aperta
#ogni dito chiuso a metà toglie -0.125 al value massimo (1) della mano aperta

#[i_initial + n]-> offset + numero del landmark della mano          [2]-> asse z

#INDICE
if list_hand_values[i_initial+8][2] < list_hand_values[i_initial+6][2]: #chiusura completa
    value-=0.25
else:
    if list_hand_values[i_initial+8][2] < list_hand_values[i_initial+7][2]:  #metà chiusura
        value-=0.125
#MEDIO
if list_hand_values[i_initial+12][2] < list_hand_values[i_initial+10][2]: #chiusura completa
    value-=0.25
else:
    if list_hand_values[i_initial+12][2] < list_hand_values[i_initial+11][2]:  #metà chiusura
        value-=0.125
#ANULARE
if list_hand_values[i_initial+16][2] < list_hand_values[i_initial+14][2]: #chiusura completa
    value-=0.25
else:
    if list_hand_values[i_initial+16][2] < list_hand_values[i_initial+15][2]: #metà chiusura
        value-=0.125
#MIGNOLO
if list_hand_values[i_initial+20][2] < list_hand_values[i_initial+18][2]: #chiusura completa
    value-=0.25
else:
    if list_hand_values[i_initial+20][2] < list_hand_values[i_initial+19][2]: #metà chiusura
        value-=0.125
```

Figura 6.65: Script di calcolo della shape key per il battito alare

Lo script indica come vengono esaminate le coordinate di movimento dei punti delle dita ad ogni singolo frame, in corrispondenza del quale vengono analizzati i 22 landmark della mano salvati in un array "list_hand_values". L'indice "i_initial" fa da offset per posizionare il puntatore dell'array alle coordinate del frame successivo ad ogni ciclo della funzione. Per ogni dito viene considerata la punta finale, l'inizio e la fine della seconda falange. Si parte considerando il valore della shape key a 1, che è il valore massimo e si sottraggono dei valori specifici 0.25 e 0.125. Ogni dito ha un peso di sottrazione che influenza la chiusura totale della mano fino al raggiungimento del valore 0:

* -0.25: se è completamente piegato, ovvero la punta del dito ha una posizione più bassa rispetto al punto di inizio della seconda falange.
* -0.125: se è piegato a metà, quando la punta del dito ha una posizione più bassa della fine della seconda falange ma più alta dell'inizio della seconda falange.

– **Risultati ottenuti**

Figura 6.66: Ali distese

Figura 6.67: Battito ali

Figura 6.68: Ali corte

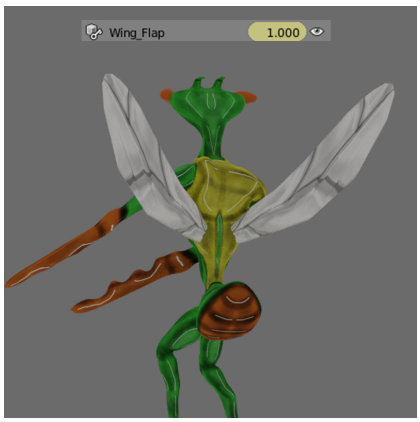


Figura 6.69: Ali allungate

Figura 6.70: Ali richiuse

Figura 6.71: Ali aperte

- **Scarabeo: Shape Key per le mandibole**
  La gestione delle mandibole dello scarabeo è controllata dal gesto "pinza", coordinato dal pollice e indice. Le forme alterate delle due zanne sono provocate dalla distanza che c'è tra il pollice e l'indice. Il movimento è valutato sull'asse X e Z di Blender, in quanto, per calcolare la distanza tra due punti, è necessario un sistema cartesiano con asse verticale e asse orizzontale.
  La shape key che controlla l'allungamento e la chiusura delle mandibole è **Mandible_Horns** che può assumere un valore di stato:

  - **0:** se il pollice e indice sono a contatto.
  - **1:** se il pollice e indice sono alla massima distanza di non contatto.
  - **Tra 0 e 1:** in base alla distanza indice-pollice.

  - **Funzione per la gestione della distanza pollice-indice**
    E' stata realizzata una funzione che converte la distanza tra pollice e indice in un valore [0,1] per la shape key di controllo delle mandibole dello scarabeo.
    La funzione prende il nome di BEETLE_clamp_hand e accetta in ingresso il parametro "Mandible_Horns" come shape key. Il valore di stato della shape key sarà aggiornato ad ogni frame in base al gesto "pinza" registrato durante la cattura della mano. Il calcolo di stato della shape key necessita di due passaggi fondamentali:

    1. **Calcolo della massima distanza pollice-indice**
       Vengono analizzati i dati di cattura ad ogni frame e si trova la distanza massima che c'è stata tra pollice e indice durante tutto il tempo di registrazione della mano. Una volta che il valore massimo viene considerato come stato '1' nella shape key, si procede con un'associazione tra massima distanza delle due dita e massimo valore della forma chiave. Il valore di distanza massimo è memorizzato in una variabile "max_distance", utilizzato dal secondo blocco di codice presentato di seguito.
    2. **Conversione distanza pollice-indice in valore della shape key**

```
value=1
#[i_initial + n]-> offset + numero del landmark della mano     [0]-> asse x    [2]->asse z

# CALCOLO DELLA DISTANZA POLLICE-INDICE
x1,z1=list_hand_values[i_initial+4][0],list_hand_values[i_initial+4][2] #PUNTA POLLICE
x2,z2=list_hand_values[i_initial+8][0],list_hand_values[i_initial+8][2] #PUNTA INDICE

distance=math.hypot(x2-x1,z2-z1)

#CALCOLO DEL VALORE DELLA SHAPE KEY
value=distance/max_distance
```

Figura 6.72: Script di calcolo della shape key per le mandibole

Lo script analizza ad ogni frame le posizioni dei landmark della punta del pollice e dell'indice sull'asse orizzontale X e sull'asse verticale Z. Viene calcolata la distanza di contatto pollice-indice per mezzo di una funzione matematica math.hypoth messa a disposizione dalla libreria Math. In seguito, si esegue una rapporto tra la distanza in quel frame e la distanza massima calcolata nel punto precedente, in modo da ottenere un valore compreso tra 0 e 1, necessario per comprendere se è avvenuto o meno il contatto fra le dita.

– **Risultati ottenuti**

Figura 6.73: Zanne corte

Figura 6.74: Zanne lunghe

- **Ragno: Shape Key per le zampe**
  Il ragno modellato in Blender presenta, sul dorso, otto zampe movibili: quattro zampe sono disposte sul lato sinistro e le altre quattro sono disposte sul lato destro. Per il controllo di queste articolazioni, che dipendono dal conteggio delle dita alzate in una mano, sono stati considerati l'indice, il dito medio, l'anulare e il mignolo. Il movimento di ciascun componente appena citato influenza una coppia di zampe (la coppia avrà una zampa destra ed una zampa sinistra). Pure essendo un'elaborazione simile alla gestione del battito alare, sarà caratterizzata da una maggiore specificità poiché in questo caso non si considera l'insieme della dita, bensì un singolo dito che ha effetto su una sola forma chiave, provocando il controllo di una coppia di zampe . Per considerare il conteggio di un dito si prende come riferimento l'asse Z verticale: quando un dito è completamente abbassato non viene conteggiato, mentre quando è completamente disteso verso l'alto va a influenzare la mesh di due zampe del ragno.

  Le shape key che controllano le zampe posteriori del ragno sono in totale quattro ed il controllo viene effettuato dall'alto verso il basso tramite:

1. **Paws1:** gestisce le due zampe superiori.
2. **Paws2:** gestisce le due zampe intermedie superiori.
3. **Paws3:** gestisce le due zampe intermedie inferiori.
4. **Paws4:** gestisce le due zampe inferiori.

Ogni singola shape key può assumere un valore di stato:

- **0:** se il dito è abbassato.
- **0.5:** se il dito non è completamente abbassato.
- **1:** se il dito è alzato.

- **Funzione per la gestione del conteggio delle dita alzate**
  L'analisi dei movimenti delle dita escluso il pollice ha come scopo la trasformazione delle coordinate 3D di questi punti in un valore adatto alle shape key delle zampe in un valore tra [0,1].
  Questo compito è svolto dalla funzione SPIDER_finger_counter che richiede in ingresso le quattro shape key "Paws1", "Paws2", "Paws3" e "Paws4". Le forme chiave assumono un valore di stato ad ogni frame, in base a quale movimento è stato svolto dalle dita (se alzate o abbassate).

```
#si controllano in totale 8 zampe in modo simmetrico (left,right)
value_1=1 #valore dell'indice-> controlla 2 zampe in alto
value_2=1 #valore del medio -> controlla 2 zampe intermedie in alto
value_3=1 #valore dell'anulare -> controlla 2 zampe intermedie in basso
value_4=1 # valore del mignolo-> controlla 2 zampe in basso

#[i_initial + n]-> offset + numero del landmark della mano          [2]-> asse z

#INDICE
if list_hand_values[i_initial+8][2] < list_hand_values[i_initial+6][2]: #alzato
    value_1-=1
else:
    if list_hand_values[i_initial+8][2] < list_hand_values[i_initial+7][2]:  #metà abbassato
        value_1-=0.5
#MEDIO
if list_hand_values[i_initial+12][2] < list_hand_values[i_initial+10][2]: #MEDIO
    value_2-=1
else:
    if list_hand_values[i_initial+12][2] < list_hand_values[i_initial+11][2]:  #metà abbassato
        value_2-=0.5
#ANULARE
if list_hand_values[i_initial+16][2] < list_hand_values[i_initial+14][2]: #alzato
    value_3-=1
else:
    if list_hand_values[i_initial+16][2] < list_hand_values[i_initial+15][2]: #metà abbassato
        value_3-=0.5
#MIGNOLO
if list_hand_values[i_initial+20][2] < list_hand_values[i_initial+18][2]: #alzato
    value_4-=1
else:
    if list_hand_values[i_initial+20][2] < list_hand_values[i_initial+19][2]: #metà abbassato
        value_4-=0.5
```

Figura 6.75: Script di calcolo delle shape key per le zampe

Lo script, che si occupa di calcolare i valori di stato delle shape key appartenenti alle zampe del ragno considera le articolazioni in movimento sull'indice, medio, anulare e mignolo. Ogni dito va a influenzare le variabili "value_1", "value_2", "value_3", "value_4" che saranno poi associate alle shape key "Paws1","Paws2","Paws3","Paws4" per l'alterazione della mesh delle zampe frame per frame. All'inizio il valore della shape key di un singolo dito viene considerato al massimo e in base alla posizione del dito si sottraggono i valori 0.5 o 1. In particolare, il dito alzato o meno influenza il peso di sottrazione secondo i valori:

* -1: se il dito è completamente abbassato, ovvero la punta del dito ha una posizione più bassa rispetto al punto di inizio della seconda falange.
* -0.5: se il dito è mezzo abbassato, si verifica quando la punta del dito ha una posizione più bassa della fine della seconda falange ma più alta dell'inizio della seconda falange.

– **Risultati ottenuti**

Figura 6.76: Zampe ragno 1

Figura 6.77: Zampe ragno 2

- **Formica: Shape Key per le antenne**
  L'allungamento delle antenne della formica è gestito mediante la gesture "virgolette": le antenne vengono considerate a riposo quando indice e medio sono abbassati, mentre quando entrambe le dita vengono alzate causano l'allungamento delle antenne in direzione frontale alla testa. Il movimento delle due dita viene analizzato sull'asse verticale Z e la gestione delle shape key utilizza una tecnica simile alle zampe del ragno, anche se, in questo caso, si tiene conto solo di due dita. L'indice va a influenzare l'antenna sinistra, mentre il dito medio permette di alterare la forma dell'antenna destra. Le shape key che controllano le antenne sono due, **Antenna1** e **Antenna2**, rispettivamente

influenzate da indice e medio. Le due forme chiave possono assumere come valori di stato:

- **0:** se il dito è abbassato.
- **0.5:** se il dito non è completamente alzato.
- **1:** se il dito è alzato.

- **Funzione per la gestione di conteggio di indice e medio**
  La creazione della funzione ANT_air_quotes_finger permette di riconoscere quando l'indice e il medio della mano sono alzati o meno. A seconda della posizione in cui esse si trovano, si calcola un valore di stato nel range [0,1] da assegnare alle shape key di gestione delle antenne. Tale funzione riceve in ingresso le due shape key "Antenna1" e "Antenna2", che assumeranno valori specifici ad ogni frame di animazione in base alla stesura del dito corrispondente. Le articolazioni di un singolo dito vengono monitorate per capire se siano rivolte verso il palmo o verso l'alto.

```
#si controllano 2 antenne della formica (left,right)
value_1=1 #indice-> controlla antenna sinistra
value_2=1 #medio -> controlla antenna destra

#ogni dito abbassato toglie -1 al value massimo (1) del dito alzato
#ogni dito abbassato a metà toglie -0.5 al value massimo (1) del dito alzato

#[i_initial + n]-> offset + numero del landmark della mano          [2]-> asse z

#INDICE
if list_hand_values[i_initial+8][2] < list_hand_values[i_initial+6][2]: #completamente abbassato
    value_1-=1
else:
    if list_hand_values[i_initial+8][2] < list_hand_values[i_initial+7][2]:  #metà chiusura
        value_1-=0.5
#MEDIO
if list_hand_values[i_initial+12][2] < list_hand_values[i_initial+10][2]: #completamente abbassato
    value_2-=1
else:
    if list_hand_values[i_initial+12][2] < list_hand_values[i_initial+11][2]:  #metà chiusura
        value_2-=0.5
```

Figura 6.78: Script di calcolo delle shape key per le antenne

Lo script presente all'interno della funzione assegna dei valori alle shape key delle antenne, considerando la posizione dei punti associati alla punta finale, all'inizio e alla fine della seconda falange di un dito. Inizialmente, il valore della shape key di un'antenna viene considerata a 1, quindi il dito si considera completamente alzato. In seguito, si analizza ad ogni singolo frame lo stato di movimento dell'indice e del medio sottraendo peso al valore iniziale quanto più il dito è abbassato. Ogni dito influenza quindi l'allungamento e il riposo di un'antenna, sottraendo peso al valore della shape key, fino al raggiungimento del valore 0. I pesi sottratti sono:

  - -1: se il dito è completamente abbassato, ovvero la punta del dito ha una posizione più bassa rispetto al punto di inizio della seconda falange.
  - -0.5: se il dito non è del tutto alzato, quando la punta del dito ha una posizione più bassa della fine della seconda falange ma più alta dell'inizio della seconda falange.

– **Risultati ottenuti**

Figura 6.79: Antenne formica 1

Figura 6.80: Antenne formica 2

6.3.7 Skinning

Dopo aver completato l'animazione facciale e del corpo, avendo anche gestito i dati della mano, è necessario mettere in atto il processo di Skinning. Si tratta dell'imparentamento tra lo scheletro e la mesh del personaggio 3D che deve essere deformato dal movimento delle ossa. La procedura di collegamento scheletro-mesh legata ai personaggi del videogioco ha portato buoni risultati con la metodologia Automatic Weights. Vengono associati dei pesi automatici alle ossa dell'armatura che hanno un'influenza di deformazione sui vertici adiacenti alla zona dell'osso. Questa configurazione nel caso degli insetti 3D ha apportato risultati di deformazioni ottimi senza la necessità di modificare manualmente l'influenza di ogni bone sulla mesh. Nonostante si tratti di un metodo rapido e automatico, è buona norma controllare tramite Weight Paint se effettivamente uno specifico osso vada a intaccare i vertici giusti del personaggio.

Capitolo 7

Gestione personaggi su Unreal

Una volta pronti i personaggi 3D modellati e animati tramite Gentle Mocap, possono essere importati all'interno del progetto del videogioco multiplayer. Il motore di gioco utilizzato è Unreal Engine 4. I personaggi animati sono stati studiati appositamente per garantire un loro adattamento alla maggior parte dei motori di gioco, ma poichè la nomenclatura dello scheletro corrisponde a quella standard di Unreal Engine 4, è più semplice adattare il personaggio generato in questo motore di gioco. Nei paragrafi successivi verrà spiegato come gli insetti e le loro animazioni associate verranno gestiti in Unreal. Nello specifico, verranno analizzati quattro punti fondamentali:

1. **Export e import dei personaggi**
   Come esportare correttamente il modello animato da Blender e importarlo nel progetto del prodotto videoludico in Unreal Engine 4.

2. **Modifiche ai parametri delle animazioni**
   Le animazioni importate in Unreal possono subire modifiche dei parametri per rendere il risultato dell'animazione più convincente.

3. **Animation Blend Space**
   Generazione del sistema di gestione delle animazioni e delle loro combinazioni.

4. **VFX**
   Ad ogni insetto del videogioco è associato un effetto speciale che accompagna l'esecuzione di un'abilità specifica.

5. **Scripting**
   Scripting tramite Blueprint per l'innesto in gioco delle animazioni di abilità dell'insetto e l'avvio degli effetti speciali in real-time.

7.1 Export e import

Per esportare da Blender il personaggio animato tramite Gentle Mocap e importarlo in Unreal Engine è necessario rispettare dei settaggi. Queste impostazioni sono utili affinché il modello venga importato interamente assieme alla sua mesh, texture, animazione e shape key.

EXPORT DA BLENDER

Quando il personaggio 3D è completo di mesh, texture, animazioni e shape key associate, si può procedere all'esportazione completa del modello 3D. I passaggi da seguire per un export corretto sono:

1. **Selezione della mesh e dell'armatura**
   Nella viewport di Blender è necessario selezionare la mesh del personaggio e lo scheletro generato da Gentle Mocap che può essere cercato nella hierarchy con il nome "Mocap Armature".

2. **Esportazione del modello come fbx**
   Il personaggio deve essere esportato come FBX, un formato molto comune per i modelli 3D che viene supportato da molti software. Dalla topbar di Blender bisogna navigare nel percorso "File->Export-> FBX(.fbx)".

3. **Main settings**

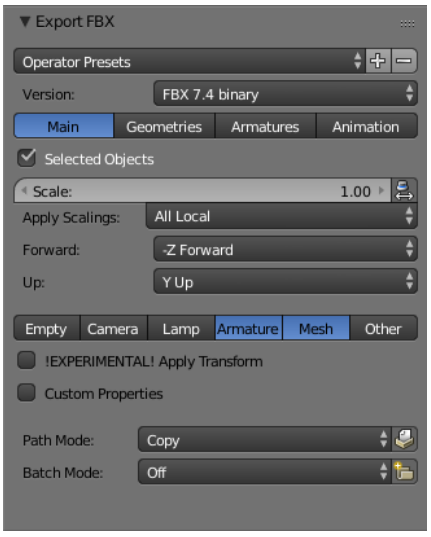
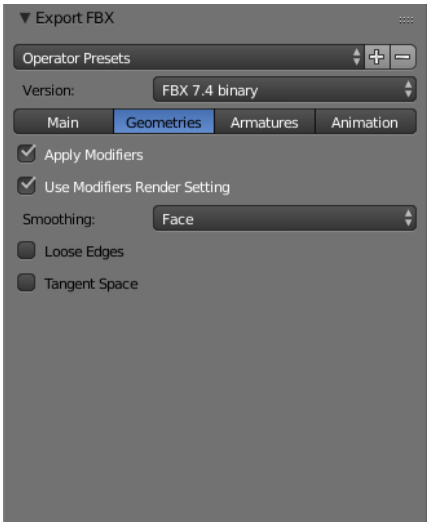


Figura 7.1: Main Settings

I parametri fondamentali da impostare nella sezione Main sono:

- *Selected objects:* da spuntare per considerare la mesh e l'armatura selezionate.
- *Armature-Mesh:* nella barra orizzontale selezionare Armature e Mesh per esportare solo quella tipologia di oggetti.
- *PathMode:* impostare su Copy e selezionare l'icona per inglobare le texture del modello all'interno del file fbx che sarà generato.

4. **Geometries settings**

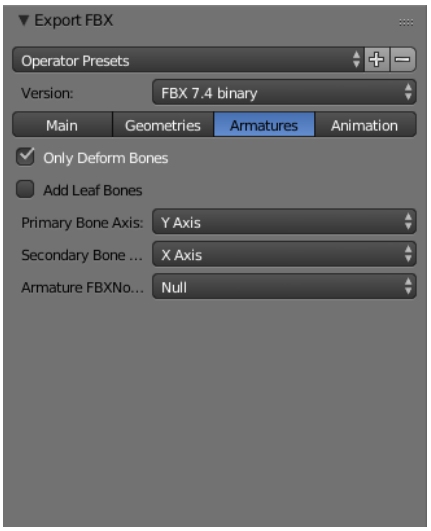


La variabile da impostare nella sezione Geometries è:

- *Smoothing:* selezionare la metodologia Face per lo smooth del modello.

Figura 7.2: Geometries Settings

5. **Armatures settings**

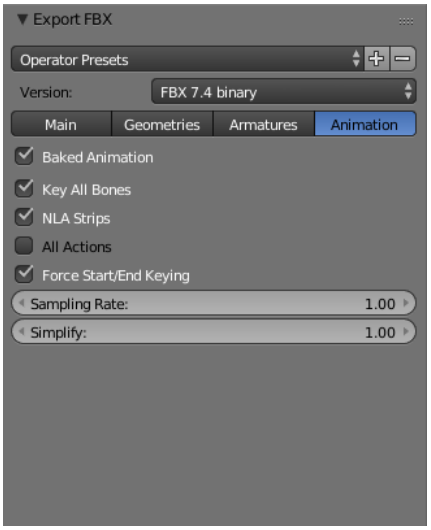


I parametri da cambiare nella sezione Armatures sono:

- *Only Deform Bones:* da spuntare per considerare solo le ossa dello scheletro che deformano la mesh.
- *Add Leaf Bones:* da deselezionare per non generare ossa terminali a quelle dello scheletro.

Figura 7.3: Armatures Settings

6. **Animation settings**



La variabile da modificare nella sezione Animation è:

- *All actions:* : da deselezionare per non considerare tutte le action singole che vanno a comporre l'animazione finale, ma permette di esportare l'animazione completa di cui si è fatto il bake.

Figura 7.4: Animation Settings

IMPORT IN UNREAL

Il file in formato fbx del modello 3d animato può essere ora importato in Unreal secondo le impostazioni visualizzate nella figura di seguito. Il menu di import di Unreal è suddiviso in varie sezioni inerenti alla mesh, all'animazione e al materiale del modello. E' necessario avere i seguenti settings di importazione:

- **Skeletal Mesh:** per importare lo scheletro associato alla mesh del personaggio.
- **Import Mesh:** per importare la mesh del modello 3D.
- **Import Morph Targets:** per consentire ad Unreal di importare delle shape key.
- **Import Animations:** per importare le animazioni che muovono il personaggio.
- **Create New Materials:** per creare e assegnare i material al modello.

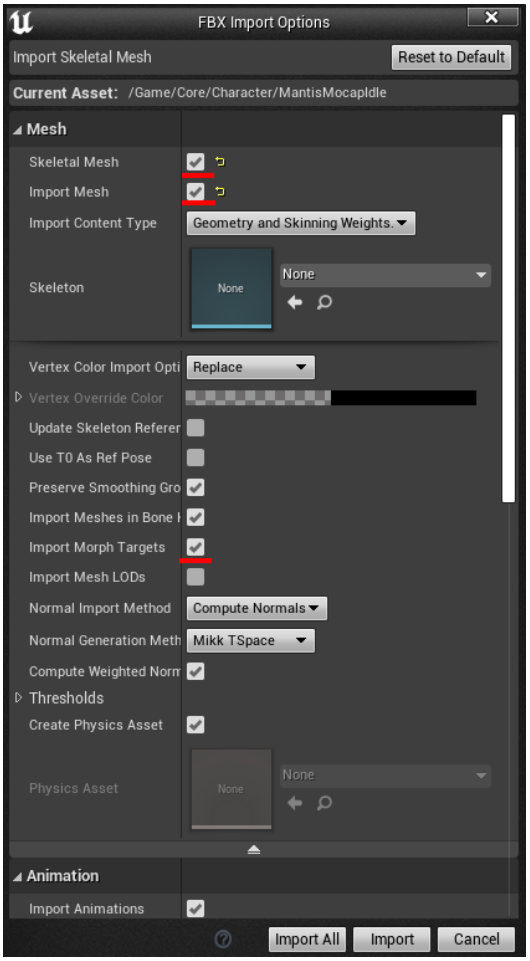


Figura 7.5: Import in Unreal prima pagina

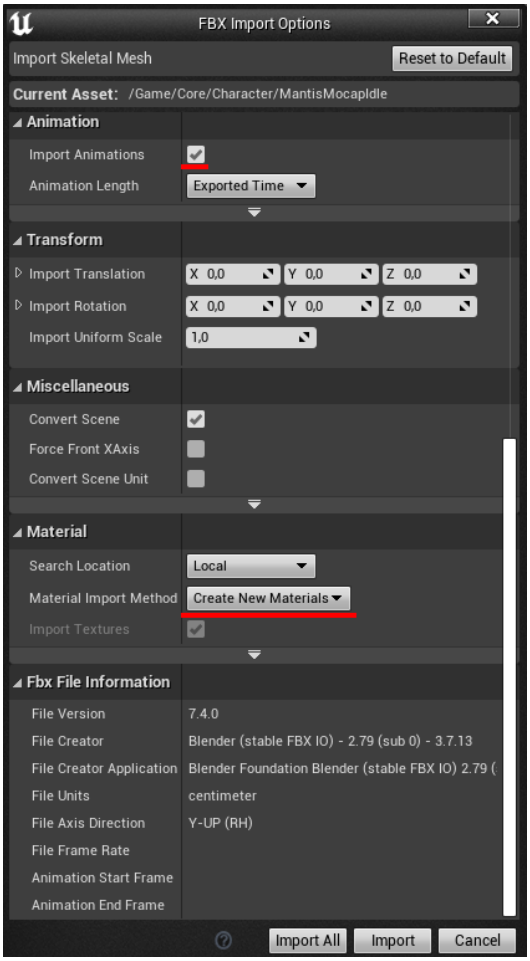


Figura 7.6: Import in Unreal, seconda pagina

7.2 Modifiche ai parametri delle animazioni

Le animazioni importate in Unreal Engine possono essere modificate dall'animator per assicurare un loro adattamento allo stile del prodotto videoludico che si vuole realizzare. Il videogioco della tesi richiede l'esecuzione di abilità da parte degli insetti, caratterizzate da animazioni rapide, quasi istantanee, in quanto il gameplay deve risultare frenetico e dinamico. Nel caso di Bugball è stato necessario aumentare o rallentare la velocità delle animazioni di un personaggio. Se durante la sessione di Motion Capture l'attore ha compiuto dei movimenti lenti, l'animazione fornita da Gentle Mocap sarà alla stessa velocità.
Per modificare l'animazione, senza la necessità di ricrearne una nuova da capo,

l'animator può impostare il parametro "Rate Scale" in Unreal, che aumenta o diminuisce la velocità dei movimenti.
Il motore di gioco fornisce anche la possibilità di correggere il movimento dello scheletro frame per frame. Questo caso è utile in quanto in Bugball sono previsti lanci del pallone da gioco e si potrebbe modificare il posizionamento del braccio per gestire meglio la presa e il tiro.

7.3 Animation Blend Space

BLEND SPACE 1D

Rappresenta uno spazio monodimensionale di blending, ovvero di fusione delle animazioni. Si tratta di un grafico 1D che possiede i valori di input lungo un asse e le animazioni da combinare sui vari punti del grafico. L'animazione combinata viene calcolata unendo le animazioni presenti sul grafico in base alla loro posizione rispetto ai valori di input correnti sull'asse di riferimento.
Il Blend Space 1D è stato utilizzato per combinare le animazioni di:

- **Idle:** animazione del personaggio a riposo.
- **Walk:** animazione di camminata del personaggio.
- **Run:** animazione di corsa del personaggio.

Figura 7.7: Blend Space 1D dei personaggi del videogioco

L'asse di riferimento è Speed ed indica che i valori in input che andranno a cambiare le animazioni in riproduzione sono dati dalla velocità corrente del personaggio. La velocità può assumere valori che vanno da 0 a 375, come indicato sull'asse orizzontale. Le animazioni sono cambiate in base alla velocità di input come elencato di seguito:

- *Speed 0:* animazione di Idle.
- *Speed 93 circa:* animazione di Walk.
- *Speed 375:* animazione di Run.

- *Speed tra 0 e 93:* mix tra animazione di Idle e Walk.
- *Speed tra 93 e 375:* mix tra animazione Walk e Run.

SISTEMA DI LOCOMOTION

Rappresenta il grafo che controlla il passaggio di stato tra le animazioni, quindi indica tramite diramazioni quale animazione il personaggio deve riprodurre in base a una determinata condizione. Il locomotion dei personaggi del videogioco richiede un'unica diramazione dall'Entry point in quanto è necessario gestire solo il passaggio da stato di riposo a camminata e corsa che è controllato singolarmente dal Blend Space 1D. Il Blend Space 1D riceve in ingresso la velocità (variabile Speed) e cambia l'animazione in base ad essa.

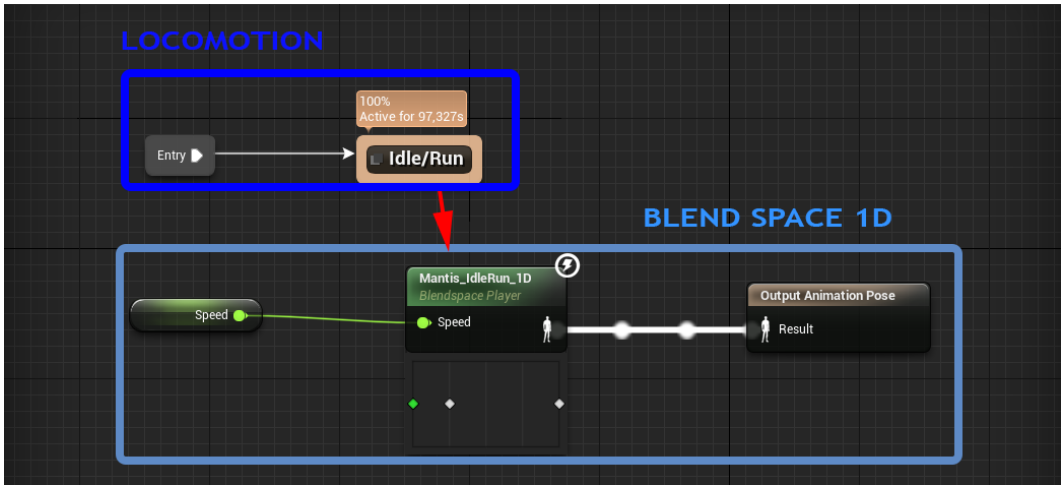


Figura 7.8: Locomotion e Blend Space 1D

ANIMATION BLEND SPACE

L'Animation Blend Space è lo spazio di gestione delle animazioni tramite scripting visuale con Blueprint, che si suddivide in due tipologie di grafici:

- **Event Graph:** per la gestione di eventi e variabili.
- **Anim Graph:** per il controllo delle animazioni.

Per analizzare l'Animation Blend Space da un punto di vista delle animazioni, bisogna focalizzarsi sulla composizione dell'Anim Graph, che utilizza al suo interno il Locomotion descritto precedentemente.

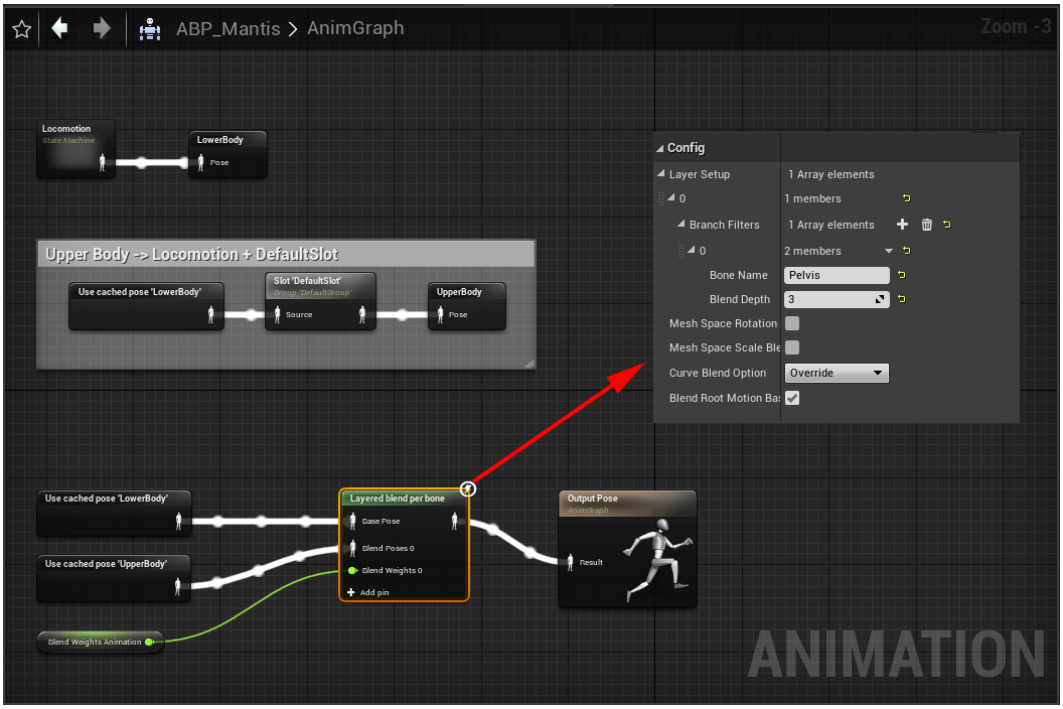


Figura 7.9: Anim Graph

Il sistema riceve in ingresso il Locomotion, ovvero il movimento a riposo, di camminata o corsa effettuato dal personaggio in base alla velocità. Lo scheletro animato del personaggio viene poi suddiviso in due parti:

- *Parte superiore del corpo:* dal bacino in su.
- *Parte inferiore del corpo:* dal bacino in giù.

Questa divisione del corpo in due parti è importante in quanto, se il personaggio è in camminata/corsa e decide di lanciare un'abilità, allora le gambe continuano a eseguire l'animazione di camminata, mentre la parte superiore del corpo attua l'animazione di abilità. La suddivisione dello scheletro animato è gestita dall'oggetto "Layered Blend per bone". Il processo si occupa della fusione proveniente dalla parte superiore e inferiore del corpo. Nelle impostazioni di configurazione, il Layer Setup possiede come selezione l'osso Pelvis per dividere le zone animate del corpo in due parti. Il "Layered Blend per bone" riceve in ingresso le due pose, una inerente alla metà superiore del corpo e una associata alla parte inferiore del corpo. Il valore di "Blend depth" dell'osso Pelvis è stato settato a 3 per ottenere una combinazione delle animazioni quasi completa. In generale "Blend Depth" indica uno stato tra 0 e 10 che influenza la fusione delle animazioni tra le due parti dello scheletro. Avendo in ingresso le due pose "Base Pose" e "Blend Poses 0", il blend depth agisce secondo

questo schema:

Figura 7.10: Blend Depth

Il terzo parametro di "Layered Blend per bone" è "Blend Weights Animation" che permette di cambiare il peso della combinazione tra le due pose ricevute in ingresso. Questa variabile viene cambiata dai game programmer in base a come si vuole che le animazioni siano fuse tra loro:

- **Blend Weights Animation=1:** se si vuole mantenere la combinazione delle animazioni di Abilità e Idle/Walk/Run per le due parti del corpo.

- **Blend Weights Animation=2:** se si vuole riprodurre solo l'animazione di abilità dell'insetto escludendo le animazioni di Idle/Walk/Run. Si bloccano quindi le animazioni base per avviare solo quelle delle abilità.

7.4 VFX

I personaggi del videogioco sono dotati di abilità le cui animazioni sono accompagnate dall'aggiunta di VFX, gli effetti speciali. I visual effect rendono l'animazione degli insetti più ricca di dettagli, visualizzando varie forme particellari per le skillshot, attacchi ad area o cure e potenziamenti degli alleati.

SOCKET

Unreal Engine consente di creare socket nel suo set di strumenti di animazione. I socket sono oggetti vuoti all'interno dello scheletro che hanno come genitore un osso specifico o la root dello scheletro e vengono utilizzati per attaccare un oggetto al socket in una determinata posizione, rotazione e con una specifica dimensione. Nel videogioco l'oggetto che si attacca al socket della mano è il pallone da gioco. Ma il socket indica anche un punto di riferimento da cui far partire i VFX.

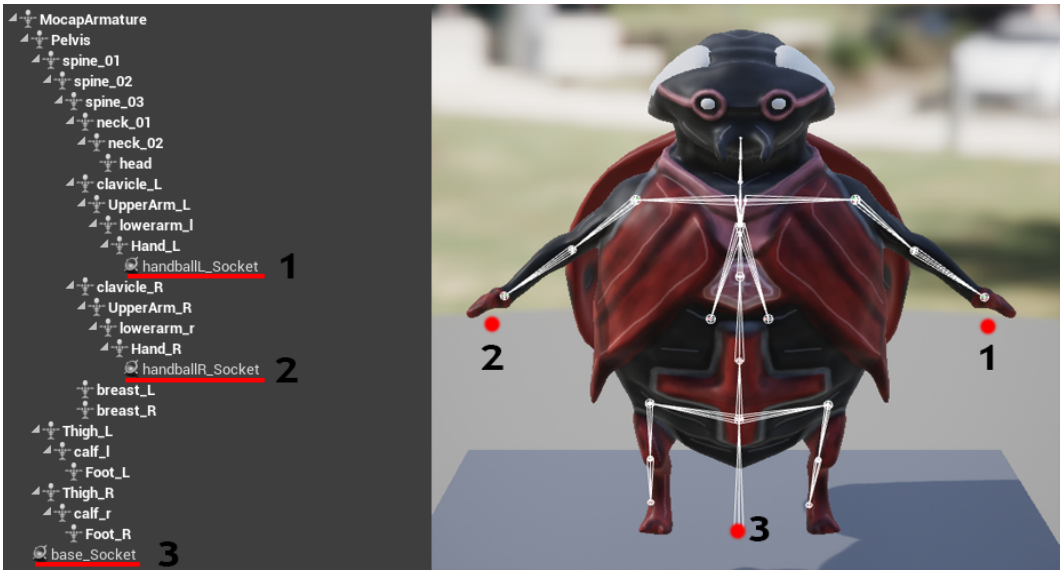


Figura 7.11: Socket della coccinella

Considerando la gerarchia dello scheletro associato alla coccinella, sono presenti tre socket: uno per la mano sinistra, uno per la mano destra e uno per la base dell'insetto. Questi punti di riferimento sono utilizzati per gestire i visual effect associati alle animazioni di tiro, di stun e di abilità della coccinella. Gli altri cinque insetti possono avere anche loro un numero diverso o uguale di socket in base a dove l'effetto speciale deva essere visualizzato assieme all'animazione.

EVENT GRAPH

L'Event Graph è utilizzato nell'Animation Blend Space per la gestione degli eventi associati all'esecuzione dei VFX nel momento in cui viene scatenata un'animazione specifica. Un effetto speciale può essere azionato alla ricezione di un AnimNotify proveniente dall'animazione, che ad un frame specifico notifica l'event graph che può far partire il VFX.

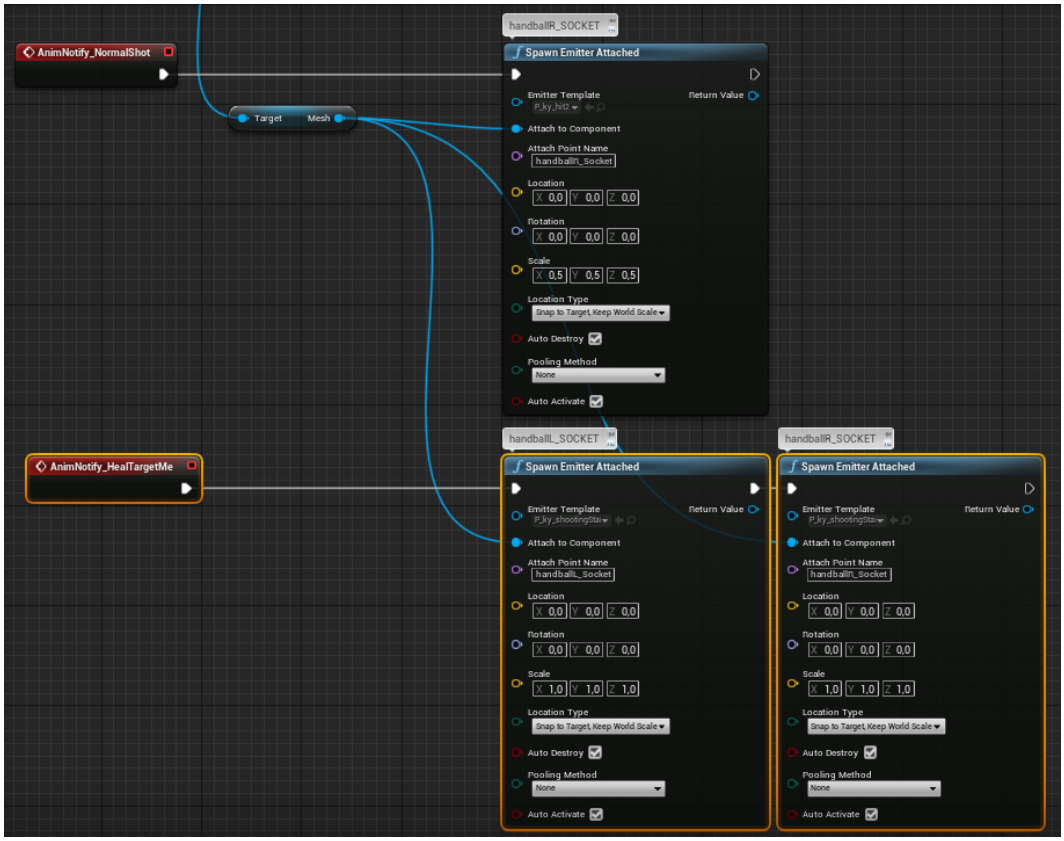


Figura 7.12: Azionamento dei VFX della coccinella

Come si può vedere in figura, nell'esempio della coccinella, quando arriva una notifica inerente all'animazione di tiro normale "AnimNotify_NormalShot" viene generato e visualizzato l'effetto speciale passato come Emitter Template sul socket della mano destra "handballR_Socket". Quando viene invece ricevuta la notifica dell'animazione di cura "AnimNotify_HealTargetMe" vengono azionati due effetti speciali: uno per il socket della mano destra e uno per il socket della mano sinistra. Lo stesso ragionamento per la generazione dei VFX in un preciso istante dell'animazione è stato seguito anche per gli altri personaggi del videogioco.

7.5 Scripting

In aiuto ai game programmer, è stata eseguita una parte di scripting visiva in Blueprint per rendere disponibili le animazioni legate al tiro normale, alla raccolta della palla e alle abilità associate all'insetto. Invece, le altre animazioni di Idle, Walk e Run sono automaticamente gestite nell'Animation Blend Space in base al parametro di velocità del personaggio. Il game programmer ha la necessità

di avere a disposizione le animazioni del personaggio in quanto è nella logica del player decidere quando un certo input implica l'avvio di un'animazione specifica, ad esempio di abilità.

ANIMATION MONTAGE

Per rendere più semplice l'utilizzo e l'importazione delle animazioni del personaggio, è stato ideato l'utilizzo di un array di Montage. Un montage in Unreal è un oggetto specifico che contiene l'animazione e abilita la gestione e il controllo di animazioni all'interno di Blueprint o tramite codice. L'array di Montage è una variabile chiamata "Ability Anim Montage Array", dove il game programmer può caricare una serie di animazioni associate all'insetto. L'obiettivo è prelevare dal vettore l'animazione interessata e azionarla.
Come visto in precedenza nell'Animation Blend Space, il programmatore deve impostare una variabile "Blend Weights Animation" che consente di cambiare il peso della combinazione tra la posa del montage e la posa base del personaggio.

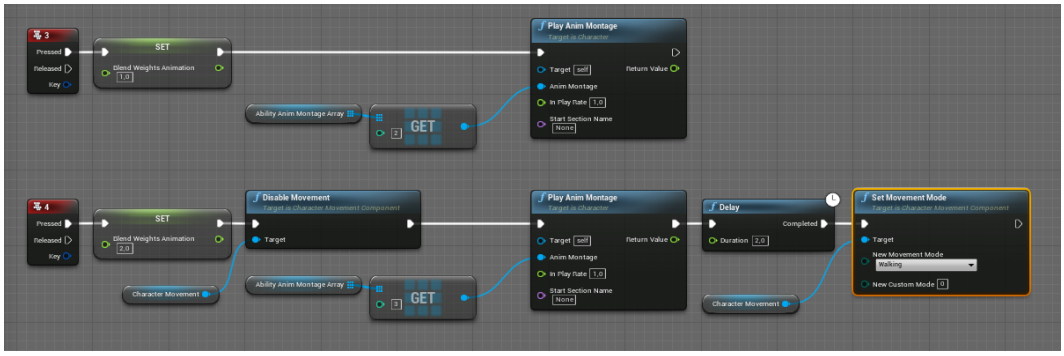


Figura 7.13: Scripting e Animation Montage

Analizzando il primo script, in corrispondenza della ricezione dell'input viene impostato a 1 il peso della fusione tra animazione del Montage e posa base del personaggio. Questo indica che le due pose andranno a combinarsi tra loro. In seguito, viene estratto dall'array il Montage nella posizione 2 e ne viene fatto il play.
Il secondo script associa al "Blend Weights Animation" un valore 2. Tale valore implica che il personaggio esegua solo l'animazione del Montage, mentre la posa base di idle, camminata o corsa, verrà bloccata. Con il comando "Disable Movement" si ferma il personaggio sul posto per poi far partire l'animazione estraendo dal vettore il Montage in posizione 3. Al termine del Montage e a seguito di un certo tempo di delay, il personaggio può ritornare a camminare.

Capitolo 8

Conclusioni

Il presente lavoro ha cercato di illustrare un nuovo software di Motion Capture full body e a costi molto bassi che consenta di animare personaggi 3D dalla forma umanoide.

La novità introdotta da Gentle Mocap è la capacità del sistema di cattura di adattarsi ad una singola telecamera, anche di scarsa qualità, in grado di registrare i movimenti dell'attore. Il costo del tool è completamente gratuito e l'unica spesa richiesta è legata alla manodopera dei rigger e degli animator che l'azienda videoludica deve sostenere. Lo sviluppo di Gentle Mocap ha il grande vantaggio di abbattere la necessità di utilizzo di strumentazioni fisiche come tute dotate di sensori inerziali o telecamere in alta definizione che filmino il movimento dell'attore. Il processo di cattura e di riconoscimento delle varie parti del corpo, associato alla conseguente generazione dello scheletro e dell'animazione del personaggio, sono entrambi svolti via software.

Adoperando Gentle Mocap, anche un utente inesperto potrebbe cimentarsi nella creazione di animazioni precise e fluide. L'utente potrebbe essere egli stesso l'attore e ricoprire anche il ruolo di rigger e animator. Il tool risulta così intuitivo e semplice che anche un principiante sarebbe in grado di utilizzare il software con facilità.

Un altro beneficio legato al tool innovativo è la semplificazione delle tempistiche legate alle sessioni di Motion Capture, alla realizzazione della struttura ossea del modello 3D e alla conversione dei dati di cattura in animazione. La preparazione delle telecamere per la registrazione dell'attore e l'atto di indossare la tuta implica forza lavoro e un allungamento delle tempistiche prima di iniziare il Motion Capture. Con Gentle Mocap, non esiste nessuna procedura da eseguire in precedenza, ma l'utente deve solo posizionarsi davanti alla telecamera ad un'adeguata distanza. Inoltre, il risparmio inerente al concetto di tempo è collegato al metodo automatico

di creazione dello scheletro e trasferimento dei dati di Mocap sul modello 3D da animare.

In pochi semplici passi, il risultato finale è molto simile ad un Motion Capture tradizionale, con movimenti fluidi e realistici che rispettano con fedeltà le pose messe in atto dall'attore. Ma nonostante i processi automatici siano allettanti, richiedono comunque una manodopera aggiuntiva da parte dei rigger e degli animator. Potrebbero verificarsi errori impredicibili che vanno ad influire sul risultato finale dell'animazione, il che è un evento indesiderato per un personaggio da importare in un prodotto videoludico. Per tale problema è necessaria una figura professionale che manipoli e corregga questi errori, ma si tratta di modifiche minime in quanto Gentle Mocap fa gran parte del lavoro.

Perchè scegliere Gentle Mocap? Perchè è un processo automatico che gratuitamente, senza nessuna strumentazione fisica superflua e in pochi minuti permette di creare ottime animazioni sui personaggi 3D umanoidi tramite la tecnica moderna di Motion Capture.

Bibliografia

Bibliografia Classica

[1] Brian Windsor Midori Kitagawa. *MoCap for Artists: Workflow and Techniques for Motion Capture*. Focal Press, 2008.

[2] Ricardo Tobon. *The Mocap Book: A Practical Guide to the Art of Motion Capture*. Foris Force, 2010.

Sitografia

[3] *Wikipedia*. URL: https://it.wikipedia.org/wiki/Motion_capture.

[4] *Blender*. URL: https://docs.blender.org/.

[5] *Unreal Engine*. URL: https://docs.unrealengine.com/4.27/en-US/.

[6] *MediaPipe*. URL: https://google.github.io/mediapipe/.

[7] *Google AI Blog*. URL: https://ai.googleblog.com/2020/08/on-device-real-time-body-pose-tracking.html.

[8] *Motion Scan*. URL: https://lanoire.fandom.com/wiki/MotionScan.

[9] *Everyeye*. URL: https://www.everyeye.it/articoli/speciale-il-passaggio-2d-3d-videogiochi-1032.html.

[10] *Geeze Zone*. URL: https://geezezone.com/threads/facts-about-the-video-games-transition-from-2d-to-3d.1146/.

[11] *Plural Sight*. URL: https://www.pluralsight.com/blog/film-games/80s-now-evolution-animation-video-games.

[12] *GPEM*. URL: https://gpemmocap.wordpress.com/2014/10/02/levoluzione-del-motion-capture-nel-cinema/.