Body Swap ed Embodiment tramite VR

Igor Iurevici[1], *Paola* Persico[1], e *Gianluca* Spiller[1]

[1] *Università Alma Mater Studiorum di Bologna*

Parole chiave: Realtà Virtuale, VR, Body Swap, Embodiment, Unity3D, IK, Avatar.

1 Introduzione

Le tecnologie VR offrono l'opportunità di coinvolgere psicologicamente l'utente che le sperimenta, e di suscitare forti reazioni emotive. L'obiettivo di questo progetto è l'applicazione delle tecnologie VR alle arti visive per creare un'esperienza immersiva, orientata alla sensibilizzazione sociale, da esporre in una mostra d'arte.

Il focus è la percezione dell'utente all'interno dell'ambiente nel quale si trova inserito, osservando la sua reazione ad un corpo difforme dal proprio.

Grazie all'immersività e al senso di Presenza che si va a creare grazie alle tecnologie VR, l'utente vive l'esperienza con naturalezza, e ciò rende le reazioni possibile oggetto di rappresentazione artistica.

In alcuni studi [1][2] viene evidenziato come l'esperienza di *embodiment* comporti, sulla base dell'avatar che ci rappresenta, una variazione del nostro bias razziale implicito. Questo potrebbe essere dovuto ad un'influenza a livello psicologico dell'utente, che si sentirebbe immedesimato nell'avatar e che modificherebbe la percezione di se stesso in funzione di questo.

Dopo aver quindi analizzato varia letteratura sullo stato dell'arte e le sperimentazioni che sono state fatte in materia [1][2], sono state pensate più esperienze in grado di coinvolgere gli utenti.

La prima esperienza prevede l'impersonificazione, tramite HMD, in un avatar di genere femminile, con una fase di esplorazione del nuovo corpo tramite uno specchio, ed una fase di accerchiamento che prevede un grande numero di agenti virtuali ostili.

La seconda esperienza prevede lo scambio dei corpi tra due soggetti differenti (un uomo e una donna). Questa seconda esperienza è stata implementata in due diversi modi. La prima implementazione prevede la scansione dei soggetti tramite scanner 3D e la condivisione di un ambiente virtuale tramite HMD. La seconda implementazione prevede l'utilizzo di visori dotati di webcam.

2 Implementazione

In questa sezione verranno discusse le soluzioni implementative relative al Body Swap e all'Embodiment.

2.1 Embodiment

Per quanto riguarda lo sviluppo dell'esperienza di Embodiment, è stata usata la piattaforma Unity3D.

Per rendere l'applicazione compatibile con diversi head-mounted display, è stato installato il package OpenXR 1.1.1, supportato da SteamVR. Lo standard OpenXR è compatibile soltanto con il nuovo Input System action-based, per cui è stato installato anche l'XR Interaction Toolkit 1.0.0-pre.4, che, oltre a strumenti per la gestione delle interazioni basati sulle azioni e non sui dispositivi, offre anche un utile simulatore per il testing (l'XR Device Simulator).

Il testing in laboratorio è stato effettuato con un HTC Vive Pro, collegato ad un computer Alienware (dotato di processore Intel Core i7-6850k, 32GB di RAM e due schede grafiche Nvidia GTX 1080) tramite cavo, e due controller Vive.

2.1.1 Ambiente

L'esperienza dell'utente è fortemente determinata dalle sue fasi. Per porre enfasi a ciascuna di esse e alla loro transizione, è stato sfruttato il concetto di Liminal Spaces [3]. Questo concetto si basa sull'idea che un certo tipo di spazio può trasmettere un senso disorientamento e ambiguità che intercorre tra delle fasi. Uno degli esempi più comuni di liminal spaces consiste in un corridoio molto lungo che fa da stato transitorio tra una destinazione e quella successiva. Pertanto l'ambiente costruito (Fig. 1) consiste in un corridoio di forma simil pentagonale nel quale l'avatar è posizionato nell'angolo con i lati più lunghi. L'architettura di tale ambiente è stata modellata usando ProBuilder [4], un tool di modellazione 3D presente in Unity. L'ambiente del corridoio è stato mantenuto minimale e simile a quello di un hotel, conforme all'esempio tipo di un Liminal Space. Di fronte all'avatar, sul muro esterno, è stato inserito uno specchio [5] usato per aumentare la consapevolezza dell'embodiment nel nuovo corpo. Così come lo specchio, anche i modelli dell'arredo dell'ambiente sono stati importati da assets gratuiti presenti sullo Unity Asset Store.

2.1.2 Agenti virtuali

I modelli 3D degli agenti virtuali sono stati creati utilizzando il software open-source MakeHuman [6], che offre un'interfaccia intuitiva per la personalizzazione di modelli umanoidi. A ogni modello è stata aggiunta un'armatura

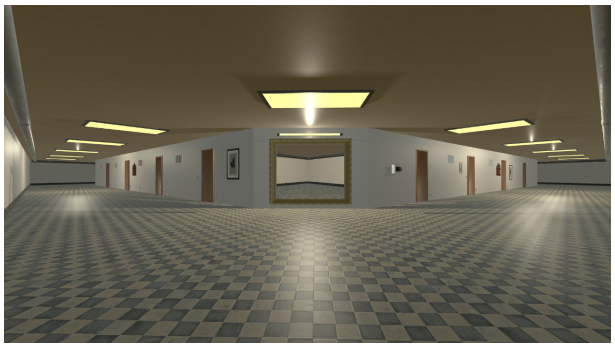

Figure 1. Ambiente modellato in base al concetto di Liminal Spaces.

Figure 2. Fase di accerchiamento dell'avatar.

compatibile con Unity usando un plugin [7], oltre alla dentatura e alla lingua necessari per l'animazione facciale.

In seguito, i modelli sono stati esportati nel formato di scambio MHX2 in modo da poter essere importati in Blender [8] previa installazione di un plugin apposito [9].

Nel menu di importazione è stata selezionata l'opzione Face Shapes per permettere a Blender di creare delle Shape Keys (corrispondenti alle Blendshapes di Unity) per facilitare la creazione delle animazioni facciali attraverso la morph target animation.

I modelli in Blender sono stati poi esportati in formato FBX per essere importati in Unity. In alcuni casi, gli shader dei materiali dei capelli e delle ciglia sono stati modificati da Standard Opaque a Standard Fade per una migliore resa.

L'armatura di ogni modello è stata impostata come Humanoid, ed è stata cambiata la configurazione dell'osso della mascella (da "jaw" a "tongue_base").

Per l'animazione facciale invece, è stato usato il tool Animation di Unity basato sull'interpolazione tra keyframe, deformando la mesh attraverso le blendshape (in particolare quelle relative alla bocca e alle sopracciglia).

Ad ogni avatar è associato un animator formato dallo stato "Walking" e "Idle", per ciascuno dei quali sono state usate due animazioni diverse per dare più credibilità alla scena. Dato che lo Unity Asset Store dispone di una ricca collezione di animazioni, abbiamo importato i due asset (gratuiti) più adatti all'esperienza del progetto [10] [11].

Nel momento in cui un agente entra in scena, questo inizia a camminare liberamente in uno spazio predeterminato con l'obiettivo di raggiungere uno dei tre target (GameObjects invisibili) scelti in maniera casuale. Tale comportamento è reso possibile grazie a NavMesh [12], una struttura dati astratta che grazie ad algoritmi di ricerca permette la riproduzione di movimenti motori pseudospontanei in un'area specifica. Lo script Spawner si occupa di istanziare agenti nella scena in maniera incrementale fino al raggiungimento di un numero prestabilito. Ogni agente è caratterizzata da cinque fasi scriptate e condizionate da alcuni timer. Le fasi sono:

1. Camminare liberamente lungo l'ambiente;
2. Guardare l'avatar;
3. Movimento labiale;
4. Fermarsi davanti all'avatar;
5. Andare via dall'avatar.

All'entrata in scena, lo stato di "Walking" viene attivato automaticamente e la velocità di movimento viene influenzata da una variabile random inclusa in un piccolo range. Trascorso il tempo relativo a ciascun timer vengono eseguite le seguenti operazioni:

- `timeToLook`: il `LookAt` target dell'animator viene impostato alla camera dell'avatar.
- `timeToTalk`: viene attivata la transizione condizionale per passare allo stato associato all'animazione "Talk";
- `timeToRaid`: il target del NavMeshAgent diventa quello di fronte all'avatar. Quando viene raggiunto, viene attivata la transizione condizionale per passare allo stato "Idle" e si termina il movimento (Fig. 2);
- `timeToReturn`: un target in un punto non visibile dall'avatar viene assegnato, e lo stato "Walking" insieme al movimento viene riattivato.

Dati i requisiti, è necessario che una piccola parte degli agenti (maggioranza di genere femminile) non passino mai alla fase di accerchiamento. Pertanto quando un agente viene istanziato, si effettua un controllo del suo genere in base al nome del GameObject (dato da quello del Prefab), e in base ad esso viene assegnata una probabilità di non passare alla fase successiva (uomini: 20%, donne: 80%).

2.1.3 Avatar

L'avatar dell'utente è stato creato con la stessa procedura degli agenti virtuali (ad eccezione del rig, generato tramite il sito web Mixamo [13]). Tuttavia, poiché i movimenti dell'avatar dovevano seguire quelli dell'utente, l'animazione ha richiesto dei passaggi aggiuntivi.

Come noto, l'HTC Vive Pro permette - grazie alla tecnologia Room-scale basata su infrarossi emessi da due base station - un tracking 6DoF della testa dell'utente; i controller, inoltre, permettono di tracciare il movimento delle mani. Per ottenere un full-body avatar a partire solo dalle informazioni sulla posizione e sulla rotazione delle estremità superiori dell'utente (ottenute inserendo nella

scena un XR Rig Room-scale), è stato quindi necessario ricorrere alla Cinematica Inversa (Inverse Kinematics, IK).

Per creare i vincoli dell'armatura, è stato dapprima installato il package Animation Rigging in modo da poter aggiungere la componente Rig Builder, oltre ad un Bone Renderer per visualizzare le ossa dell'avatar. Per le braccia, sono stati creati dei Two Bone IK Constraint i cui "target" erano i palmi delle mani e i cui "hint" erano in corrispondenza dei gomiti; per la testa invece, è stato creato un Multi Parent Constraint.

Tramite lo script VRRig sono stati allineati i target con le posizioni e le rotazioni reali delle mani e della testa dell'utente, ed è stata inserita una condizione per cui, se l'utente ruota le braccia di più di una certa soglia - 20 gradi - rispetto al torso, ruota anche il suo corpo nella stessa direzione della testa (è stata usata l'interpolazione lineare - Lerp - per rendere meno brusco il movimento). Per rendere visibili le braccia, è stato ridotto il Clipping Plane Near a 0.01.

Poiché la rotazione delle braccia risultava poco naturale, è stato testato in seguito per gli arti superiori un approccio differente, basato sul sistema Mecanim di Unity, che ha dato risultati più soddisfacenti.

Per quanto riguarda invece il movimento dei piedi, poiché la navigazione dell'utente nella scena non è tra i requisiti, non è stata applicata la Cinematica Inversa, ma sono state usate prevalentemente delle animazioni generiche, integrate attraverso l'Animator di Unity. Il Base Layer che si occupa dell'animazione delle gambe (e non interferisce con il resto del corpo grazie ad un'Avatar Mask) ha uno stato di Idle, con un'animazione del respiro umano, ed un Blend Tree di tipo 2D Simple Directional che fonde quattro animazioni: Walk e Walk Backwards (scaricate da Mixamo), Right Turn e Left Turn (presenti nel package Photon [14]). La fusione avviene attraverso due parametri, DirectionX e DirectionY, che vengono assegnati nello script VRAnimatorController sulla base della velocità dell'headset.

Un altro aspetto di cui ci si è occupati è l'accovacciamento, gestito tramite lo script VRFootIK, ed in particolare tramite la funzione di callback OnAnimatorIK. In quest'ultima funzione, viene attivata la Cinematica Inversa - e quindi il piegamento delle ginocchia - quando il Raycast che parte dai piedi interseca il pavimento. Per evitare l'accovacciamento nel caso in cui l'utente semplicemente guarda verso il basso, è stata inserita una condizione basata su un altro Raycast che parte dalla testa dell'utente controllando che la distanza dal pavimento sia superiore ad una certa soglia. Per entrambi i Raycast, è stata usata una Layer Mask in modo che venissero ignorati tutti i collider eccetto il Mesh collider del pavimento.

Infine, la gestione delle collisioni è stata effettuata applicando ad una replica dell'avatar due trigger Capsule collider e due Kinematic Rigidbody alle mani, e due Capsule collider sul torso, in modo da sospendere il movimento delle mani in caso di collisione (salvando le loro ultime posizioni nei GameObject Left Hand Target e Right Hand Target). Per ottimizzare il rendering, sono stati eliminati tutti i materiali dalla replica.

2.1.4 Interfaccia e calibrazione

Come interfaccia utente, è stato creato un menu sotto forma di canvas con un pulsante che permette di iniziare l'esperienza.

Una volta premuto il pulsante con il controller, viene attivato l'avatar e lo Spawner. All'attivazione dell'avatar, viene aggiornata la posizione del Player Offset dell'XR Rig per tenere conto della differenza tra la stima dell'altezza della testa dell'utente e l'altezza della testa dell'avatar. Per la stima della lunghezza delle braccia, invece, viene utilizzata l'euristica per la quale quest'ultima è circa pari alla metà della statura. L'offset delle braccia viene poi utilizzato come tracking offset nella funzione OnAnimatorIK.

2.1.5 Ottimizzazione

Per migliorare la performance dell'applicazione, ed in particolare aumentare gli FPS, sono state adottate le seguenti tecniche:

- occlusion culling
- Gameobject statici (Environment)
- GPU instancing dei Material ripetuti (porte, estintori, etc.)

2.2 Body Swap

Per quanto concerne la parte di Body Swap, sono stati utilizzati due approcci differenti, ognuno con vantaggi e svantaggi specifici. Se una prima implementazione prevede che ciascun utente viva l'esperienza di body swap impersonificando un avatar generato dalla scansione 3D dell'altro utente, la seconda implementazione prevede che i due utenti sperimentino il body swap attraverso la combinazione di visore e webcam ottenendo un cambio di punto di vista nel mondo reale. Per la prima esperienza la difficoltà sta appunto nel ricreare un avatar e una scena credibile, che possa massimizzare l'embodiment nell'avatar virtuale. Nel secondo caso invece, è richiesto che i due soggetti siano fortemente coordinati nel replicare gli stessi movimenti, in modo tale da minimizzare la motion sickness dovuta al disallineamento tra il movimento percepito e il movimento effettivo.

2.3 Soluzione multiplayer

In questa soluzione sono state usate le scansioni di due individui, un uomo e una donna, effettuate con lo scanner 3D Artec Eva. In primis i modelli sono stati elaborati tramite Blender nelle seguenti fasi:

1. decimazione dei poligoni (200.000 poligoni)
2. allineamento sugli assi
3. riempimento dei buchi
4. correzione delle normali tramite Flipping

Figure 3. Body Swap in VR collaborativo.

Per quanto riguarda il painting, nel caso del modello femminile è stato usato Adobe Photoshop per correggere la texture, ma nel caso del modello maschile la texture map era troppo complessa, e quindi è stato utilizzato il software 3DCoat per applicare un Vertex paint. In seguito è stata aggiunta l'armatura ai modelli attraverso Mixamo, che in seguito sono stati importati in Unity e animati attraverso la stessa procedura descritta nella sezione Avatar dell'Embodiment. Per il corretto funzionamento del vertex paint, è stato usato lo shader UltimateVertexColorShader/-Standard presente sull'Asset Store.

Per avere nella stessa scena dell'applicativo due utenti diversi, sono stati utilizzati due computer con due istanze diverse di SteamVR, e la comunicazione è stata implementata attraverso Photon Pun 2, un framework per giochi multiplayer real-time [14].

Attraverso lo script NetworkManager, viene gestita la connessione alla room del server Photon, e viene istanziato l'avatar nella scena (a partire dal relativo prefab in Resources). La scelta dell'avatar e la conseguente connessione avviene premendo uno dei due pulsanti nel canvas ("Connect as man" o "Connect as woman").

Per sincronizzare i dati della scena sulla rete, ai prefab sono state aggiunte le seguenti componenti:

- Photon Animator View e Photon Transform View agli avatar;
- Photon Transform View ai target delle mani e alle teste;
- Photon View agli avatar con tutte le componenti precedenti come componenti osservate.

La gestione delle collisioni avviene in modo simile alla modalità descritta nella sezione Embodiment, con l'accortezza di aggiornare le nuove posizioni e rotazioni nei GameObject Left Hand Target e Right Hand Target solo per il proprio avatar e per la propria replica, ma di applicare la Cinematica Inversa a tutti e quattro gli avatar.

2.4 Soluzione camera-based

Per quanto riguarda il secondo tipo di implementazione, prima di raggungere la soluzione finale, sono stati sperimentati più approcci differenti, in modo tale da trovare quello che raggiunge la latenza minore e la qualità visiva più elevata. In primis si era pensato un approccio mono-macchina, con due visori collegati a due schede video differenti. Questo avrebbe richiesto l'utilizzo di una macchina virtuale, agganciata ad una delle schede video, in modo tale da poter aprire indipendentemente due istanze Unity e avviare la simulazione. La problematica riscontrata in questo caso stava proprio nell'impostazione della scheda video sulla macchina virtuale, non resa possibile dalle soluzioni open source attualmente a disposizione. La seconda soluzione prevedeva l'utilizzo di due macchine differenti, connesse tramite rete, che reciprocamente si mandano i due flussi video. La problematica riscontrata qui era dovuta alla difficoltà di ottenere, tramite la comunicazione di rete, una qualità video molto elevata e una latenza pressoché nulla. Infatti, aumentando la qualità la latenza aumentava vertiginosamente, mentre per ridurre la latenza ci trovavamo a dover abbassare drasticamente la qualità dell'immagine, rendendo l'esperienza sgradevole e compromettendo il raggiungimento dello scopo. Questa problematica è stata riscontrata sia su rete cablata, sia su rete wireless (su rete wireless le prestazioni erano inferiori) e con più protocolli e standard di comunicazione differenti (UDP, SRTP, NDI).

Alla luce delle problematiche riscontrate, abbiamo optato per una soluzione più semplice, che risolvesse sia i problemi relativi alla latenza, sia migliorasse la qualità dell'immagine, mediante un'approccio hardware-oriented.

Essendo la risoluzione delle webcam integrate nei visori bassa, abbiamo deciso di utilizzare delle webcam esterne, connesse tramite USB. Abbiamo provato due approcci:

- webcam tradizionali (due per visore per consentire la visione stereoscopica);
- ZED Mini, delle camere per la Mixed Reality progettate appositamente per consentire la visione stereoscopica.

Utilizzando delle prolunghe per i cavi, è possibile collegare direttamente la webcam attaccata ad un visore alla macchina dell'altro utente, in modo tale da poter direzionare il flusso senza dover passare per la rete. Questo riduce drasticamente la latenza dovuta all'utilizzo della rete, e consente una qualità molto elevata dell'immagine.

Per quanto riguarda l'approccio che utilizza delle webcam tradizionali, la soluzione prevede l'utilizzo di 4 webcam, due per visore, collegate alla macchina dell'altro utente. Il programma in esecuzione, sviluppato tramite Unity 3D, prende il flusso video proveniente dalle webcam e lo riproduce sulla lente corrispondente (destra/sinistra) associando il Quad della lente con il flusso video corrispondente mediante un approccio a Layer. In questo caso si hanno quattro cavi differenti per le webcam. L'immagine in ingresso sul visore viene poi ingrandita in modo tale da riempire l'area visiva del visore, per evitare bordi nell'immagine visualizzata.

L'alternativa prevede l'utilizzo di Zed Mini, un sistema dotato di due camere e una sensoristica integrata che permette, tramite l'utilizzo della libreria correlata, di visualizzare sull'HMD un pannello che fornisce di default la visione stereoscopica (noto come ZED_Rig_Stereo). Collegando un'unico cavo e utilizzando il supporto fornito in dotazione, l'aggancio al visore e il suo utilizzo risulta semplice ed immediato. Le fotocamere, essendo integrate,

sono allineate alla perfezione e non vi è quindi il rischio di nausea dovuto al disaccoppiamento dell'immagine. Inoltre la libreria è studiata per rendere l'immagine adattiva alla posizione della testa, quindi il più confortevole possibile. È necessario un ridimensionamento del pannello anche in questo caso, attraverso la modifica della libreria di Zed Mini, perché la versione standard prevede un riquadro con bordi visibili, che non si adatta all'esperienza che si desidera fornire all'utente finale.

3 Conclusioni

3.1 Risultati

In conclusione, sono state sviluppate due esperienze VR, Embodiment e Body Swap, in grado di immergere l'utente in un corpo differente dal proprio. Queste esperienze sono adatte ad un contesto artistico, come una mostra museale, in quanto presentano una connotazione sociale.

3.2 Criticità

Tra le criticità della soluzione Embodiment, è stato notato un leggero lag presente durante le collisioni, dovuto al fatto che esse vengono gestite con un frame di ritardo per via dell'ordine di esecuzione di Unity delle funzioni evento (nello specifico, OnTrigger viene chiamata dopo OnAnimatorIK). L'animazione delle gambe, inoltre, non è precisa in quanto i movimenti delle gambe vengono stimati solo a partire dalla posizione del torso: per ottenere un'animazione migliore, sarebbe necessario l'utilizzo di sensori posti sulle caviglie e dell'applicazione della Cinematica Inversa anche agli arti inferiori. All'aumentare del numero di agenti, ci può essere un calo degli FPS, che può essere arginato combinando gli Skinned Mesh Renderer di ciascun agente.

Nella soluzione Body Swap invece, non sono state rilevate interferenze tra visori, ma una criticità è stata la condivisione dello spazio (soprattutto nell'approccio camera-based), con i rischi dovuti alle collisioni reali tra gli utenti ed i cavi. Una possibile soluzione per mitigare il problema potrebbe essere l'installazione di una struttura sul soffitto per il passaggio dei cavi e la presenza di un coordinatore.

Inoltre, una difficoltà specifica della soluzione che prevede l'utilizzo di camere tradizionali sta nella necessità di creare un supporto apposito per tenere le due webcam ben ancorate al visore e nella necessità di mantenere perfettamente allineate le due webcam per evitare la motion sickness.

Infine, per quanto riguarda l'approccio basato su Zed Mini, un fattore che va preso in considerazione è il costo, decisamente più elevato rispetto a quello delle webcam tradizionali.

4 Codice

Il codice è disponibile ai seguenti URL:

- https://github.com/emilypeek1/EmbodimentVR
- https://github.com/emilypeek1/MultiBodySwap
- https://github.com/emilypeek1/BodySwap

Riferimenti

[1] M.S. Domna Banakou, Parasuram D. Hanumanthu, *Virtual embodiment of white people in a black virtual body leads to a sustained reduction in their implicit racial bias* (2016), https://doi.org/10.3389/fnhum.2016.00601

[2] L.G.N.E.B.A.M.M. Jason Tham, Ann Hill Duin, *Understanding virtual reality: Presence, embodiment, and professional practice* (2018), https://ieeexplore.ieee.org/document/8316900

[3] Wikipedia, *Liminality*, https://en.wikipedia.org/wiki/Liminality

[4] Unity, *Unity probuilder*, https://unity.com/features/probuilder

[5] D.R.J. Johnson), *Magic mirror lite - reflection for unity*, https://assetstore.unity.com/packages/tools/particles-effects/magic-mirror-lite-reflection-for-unity-34824

[6] M. Community, *Makehuman* (2020), http://www.makehumancommunity.org/

[7] M. Community, *Unity rig* (2016), http://www.makehumancommunity.org/content/unity_rig.html

[8] B. Foundation, *Blender* (2021), https://www.blender.org/

[9] Thomas@MakeHuman, *Mhx2 documentation* (2018), https://thomasmakehuman.wordpress.com/mhx2-documentation/

[10] K. Inglesias, *Basic motions free* (2021), https://assetstore.unity.com/packages/3d/animations/basic-motions-free-154271

[11] POLYGONCRAFT, *Basic motions (free pack)* (2016), https://assetstore.unity.com/packages/3d/animations/basic-motions-free-pack-25900

[12] Unity, *Unity navmesh*, https://docs.unity3d.com/Manual/nav-BuildingNavMesh.html

[13] A.S. Incorporated, *Mixamo* (2021), http://www.mixamo.com

[14] E. Games, *Pun 2 - free* (2021), https://assetstore.unity.com/packages/tools/network/pun-2-free-119922